Manual del Desarrollador
V2.10.0-pre0-5438-g4a1d7cca75

Índice general

Capítulo 1

Introducción

This handbook is a work in progress. If you are able to help with writing, editing, or graphic preparation please contact any member of the writing team or join and send an email to emc-users@lists.sourceforge

Capítulo 2

HAL General Reference

2.1. HAL Entity Names

All HAL entities are accessible and manipulable by their names, so documenting the names of pins, signals, parameters, etc., is very important. Names in HAL have a maximum length of 41 characters (as defined by HAL_NAME_LEN in hal.h). Many names will be presented in the general form, with formatted text <*like-this*> representing fields of various values.

When pins, signals, or parameters are described for the first time, their name will be preceded by their type in parentheses (*float*) and followed by a brief description. Typical pins definitions look like these examples:

(bit) parport.<portnum>.pin-<pinnum>-in
: The HAL pin associated with the physical input pin <*pinnum*> of the db25 connector.

(float) pid.<loopnum>.output
: The PID loop output

Occasionally, an abbreviated version of the name may be used, for example the second pin above could be simply called with *.output* when it can be done without causing confusion.

2.2. HAL General Naming Conventions

Consistent naming conventions would make HAL much easier to use. For example, if every encoder driver provided the same set of pins and named them the same way, then it would be easy to change from one type of encoder driver to another. Unfortunately, like many open-source projects, HAL is a combination of things that were designed, and things that simply evolved. As a result, there are many inconsistencies. This section attempts to address that problem by defining some conventions, but it will probably be a while before all the modules are converted to follow them.

Halcmd and other low-level HAL utilities treat HAL names as single entities, with no internal structure. However, most modules do have some implicit structure. For example, a board provides several functional blocks, each block might have several channels, and each channel has one or more pins. This results in a structure that resembles a directory tree. Even though halcmd doesn't recognize the tree structure, proper choice of naming conventions will let it group related items together (since it sorts the names). In addition, higher level tools can be designed to recognize such structure, if the names provide the necessary information. To do that, all HAL components should follow these rules:

- Dots (".") separate levels of the hierarchy. This is analogous to the slash ("/") in a filename.

- Hyphens (“-”) separate words or fields in the same level of the hierarchy.
- Los componentes HAL no deben usar guiones bajos o ”mezcla de mayúsculas y minúsculas”. [1]
- Use only lowercase letters and numbers in names.

2.3. Hardware Driver Naming Conventions

nota
Most drivers do not follow these conventions in version 2.0. This chapter is really a guide for future developments.

2.3.1. Pins/Parameters names

Hardware drivers should use five fields (on three levels) to make up a pin or parameter name, as follows:

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

The individual fields are:

<nombre-del-dispositivo>
The device that the driver is intended to work with. This is most often an interface board of some type, but there are other possibilities.

<número-de-dispositivo>
It is possible to install more than one servo board, parallel port, or other hardware device in a computer. The device number identifies a specific device. Device numbers start at 0 and increment.

<tipo-es>
Most devices provide more than one type of I/O. Even the simple parallel port has both digital inputs and digital outputs. More complex boards can have digital inputs and outputs, encoder counters, pwm or step pulse generators, analog-to-digital converters, digital-to-analog converters, or other unique capabilities. The I/O type is used to identify the kind of I/O that a pin or parameter is associated with. Ideally, drivers that implement the same I/O type, even if for very different devices, should provide a consistent set of pins and parameters and identical behavior. For example, all digital inputs should behave the same when seen from inside the HAL, regardless of the device.

<chan-num>
Virtually every I/O device has multiple channels, and the channel number identifies one of them. Like device numbers, channel numbers start at zero and increment.[2] If more than one device is installed, the channel numbers on additional devices start over at zero. If it is possible to have a channel number greater than 9, then channel numbers should be two digits, with a leading zero on numbers less than 10 to preserve sort ordering. Some modules have pins and/or parameters that affect more than one channel. For example a PWM generator might have four channels with four independent ”duty-cycle” inputs, but one ”frequency” parameter that controls all four channels (due to hardware limitations). The frequency parameter should use ”0-3” as the channel number.

[1] Se han eliminado caracteres subrayados, pero quedan algunos casos de mezcla errónea, por ejemplo *pid.0.Pgain* en lugar de *pid.0.p-gain*.

[2] One exception to the ”channel numbers start at zero” rule is the parallel port. Its HAL pins are numbered with the corresponding pin number on the DB-25 connector. This is convenient for wiring, but inconsistent with other drivers. There is some debate over whether this is a bug or a feature.

<specific-name>
Un canal de E/S individual puede tener solo un pin HAL asociado con él, pero la mayoría tiene más de uno. Por ejemplo, una entrada digital tiene dos pines, uno es el estado del pin físico, el otro es la misma cosa pero invertida. Eso le permite al configurador elegir entre entradas activas bajas y activas altas. Para la mayoría de los tipos-io, hay un conjunto estándar de pines y parámetros, (denominada ”interfaz canónica”) que el driver debería implementar. Las interfaces canónicas se describen en el capítulo Interfaces canonicos de dispositivos.

Examples

motenc.0.encoder.2.position
The position output of the third encoder channel on the first Motenc board.

stg.0.din.03.in
The state of the fourth digital input on the first Servo-to-Go board.

ppmc.0.pwm.00-03.frequency
The carrier frequency used for PWM channels 0 through 3 on the first Pico Systems ppmc board.

2.3.2. Function Names

Hardware drivers usually only have two kinds of HAL functions, ones that read the hardware and update HAL pins, and ones that write to the hardware using data from HAL pins. They should be named as follows:

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

<nombre-de-dispositivo>
The same as used for pins and parameters.

<número-de-dispositivo>
The specific device that the function will access.

<tipo-es>
Optional. A function may access all of the I/O on a board, or it may access only a certain type. For example, there may be independent functions for reading encoder counters and reading digital I/O. If such independent functions exist, the <io-type> field identifies the type of I/O they access. If a single function reads all I/O provided by the board, <io-type> is not used. [3]

<chan-num-range>
Optional. Used only if the <io-type> I/O is broken into groups and accessed by different functions.

read|write
Indicates whether the function reads the hardware or writes to it.

Examples

motenc.0.encoder.read
Reads all encoders on the first motenc board.

generic8255.0.din.09-15.read
Reads the second 8 bit port on the first generic 8255 based digital I/O board.

ppmc.0.write
Writes all outputs (step generators, pwm, DACs, and digital) on the first Pico Systems ppmc board.

[3] Note to driver programmers: Do NOT implement separate functions for different I/O types unless they are interruptible and can work in independent threads. If interrupting an encoder read, reading digital inputs, and then resuming the encoder read will cause problems, then implement a single function that does everything.

Capítulo 3

Notas sobre el código

3.1. Audiencia Objetivo

Este documento es una colección de notas sobre los aspectos internos de LinuxCNC. Principalmente de interés para los desarrolladores. Sin embargo, gran parte de la información también puede ser de interés para integradores de sistemas u otros que estén simplemente interesados sobre cómo funciona LinuxCNC. Mucha de esta información está ahora desactualizada y nunca ha sido revisada su exactitud.

3.2. Organización

Habrá un capítulo para cada uno de los componentes principales de LinuxCNC, así como capítulos que cubren cómo esos componentes trabajan juntos. Este documento es por mucho un trabajo en progreso y su diseño puede cambiar en el futuro.

3.3. Términos y definiciones

- *EJE*: un eje es uno de los nueve grados de libertad que define la posición de una herramienta en el espacio cartesiano tridimensional. Los nueve ejes son referidos como X, Y, Z, A, B, C, U, V y W. Las coordenadas lineales ortogonales X, Y y Z determinan dónde está posicionada la punta de la herramienta. Las coordenadas angulares A, B y C determinan la orientación de la herramienta. Un segundo conjunto de coordenadas lineales ortogonales U, V y W permite el movimiento de la herramienta (generalmente para acciones de corte) en relación con los ejes previamente desplazados y rotados. Lamentablemente, ”eje” se usa a veces para significar un grado de libertad de la máquina en sí, como los carros longitudinal y transversal o el avance fino del husillo de una fresadora vertical. En estas maquinas, esto no causa confusión ya que, por ejemplo, el movimiento de la mesa corresponde directamente al movimiento a lo largo del eje X. Sin embargo, las articulaciones de hombro y codo de un brazo robótico y los actuadores lineales de un hexápodo no se corresponde al movimiento a lo largo de ningún eje cartesiano y en general es importante hacer la distinción entre el eje cartesiano y grados de libertad de la máquina. En este documento, esto último se llamarán *articulaciones*, no ejes. (Las GUI y algunas otras partes de el código no siempre sigue esta distinción, pero las partes internas de el controlador de movimiento si lo hacen.)
- *ARTICULACIÓN*: una articulación es cada una de las partes móviles de la máquina. Las articulaciones son distintas de los ejes, aunque los dos términos a veces se usan (incorrectamente) para significa lo mismo. En LinuxCNC, una articulación es un objeto físico que puede ser movido, no una

coordenada en el espacio. Por ejemplo, todos los carros, la palanca del husillo o un plato giratorio de una fresadora vertical son articulaciones. El hombro, el codo y la muñeca de un brazo robótico son articulaciones, al igual que los actuadores lineales de un hexápodo. Cada articulación tiene un motor o actuador de algún tipo asociado con ella. Las articulaciones no corresponden necesariamente a los ejes X, Y y Z, aunque para máquinas con cinemática trivial, puede ser el caso. Incluso en esas máquinas, la posición articular y la posición del eje son cosas inherentemente diferentes. En este documento, los términos *articulación* y *eje* se utilizan con cuidado para respetar sus distintos significados. Desafortunadamente, eso no es necesariamente cierto en ningún otro lado. En en particular, las GUI para máquinas con cinemática trivial pueden pasar por alto o oculta completamente la distinción entre articulaciones y ejes. Adicionalmente, el archivo INI usa el término *eje* para datos que serían más precisos describirse como datos de articulaciones, como las escalas de entrada y salida, etc.

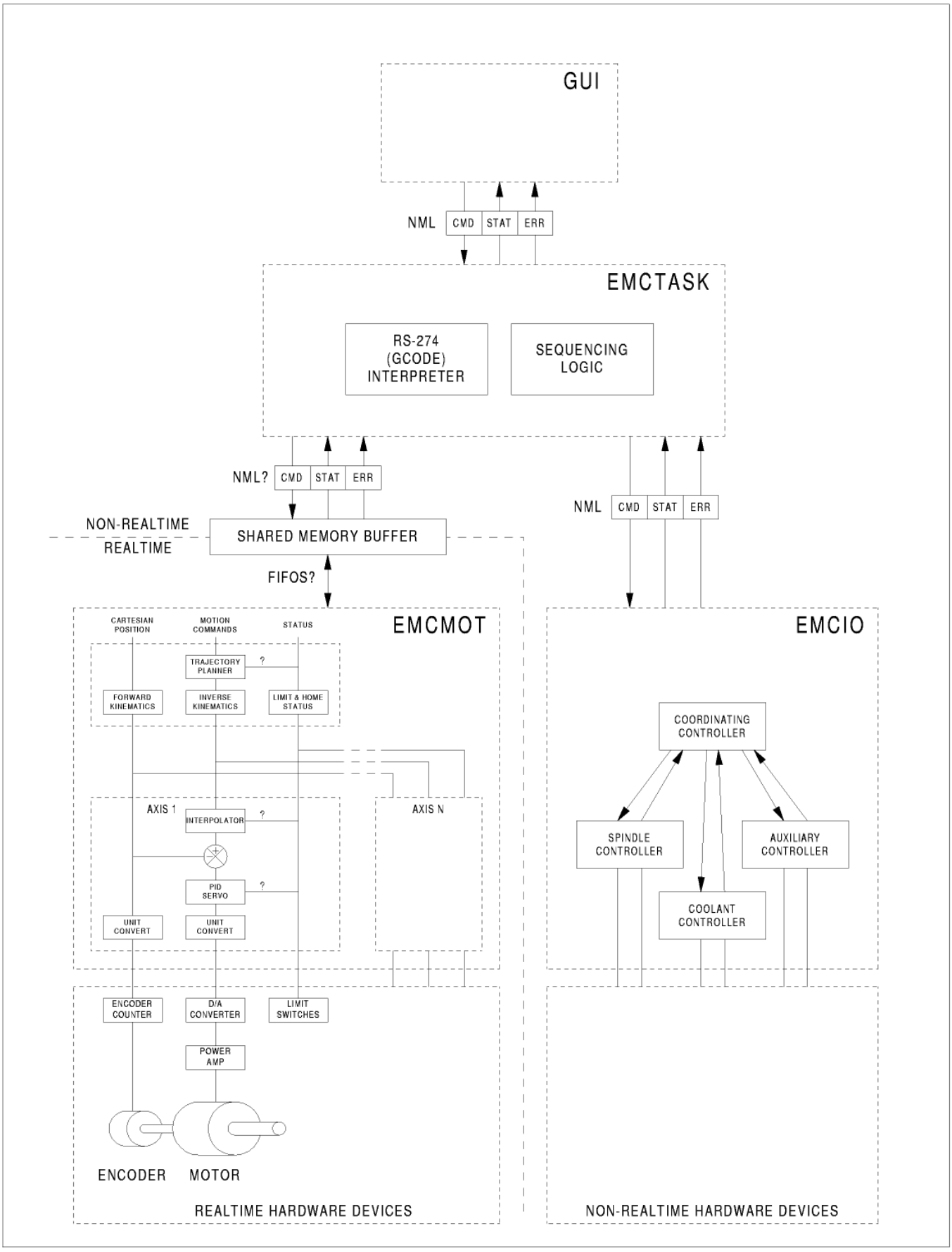
---

nota
This distinction was made in version 2.8 of LinuxCNC. The INI file got a new section [JOINT_<num>]. Many of the parameters that were previously proper to the [AXIS_<letter>] section are now in the new section. Other sections, such as [KINS], also take on new parameters to match this. An update script has been provided to transform old INI files to the new axes/joints configuration.

- *POSE*- una pose es una posición completamente especificada en un espacio cartesiano 3-D. En el controlador de movimiento LinuxCNC, cuando nos referimos a una pose nos referimos a una estructura EmcPose, que contiene seis coordenadas lineales (X, Y, Z, U, V y W) y tres angulares (A, B y C).
- *coord*, o modo coordinado, significa que todas las articulaciones están sincronizadas y se mueven juntas según lo ordenado por el código de nivel superior. Es el modo normal al mecanizar. En el modo coordinado, se supone que los comandos se dan en el marco de referencia cartesiano, y si la máquina no es cartesiana, los comandos son traducidos por la cinemática para impulsar cada articulación en el espacio articular según sea necesario.
- *free*, o modo libre, significa que los comandos se interpretan en el espacio articular. Se usa para mover manualmente (jog) articulaciones individuales, aunque no impide que se muevan múltiples articulaciones a la vez (creo). La detección del punto de inicio de los ejes (homing) también se realiza en modo libre; de hecho, a las máquinas con cinemática no trivial deben tener detectados sus puntos de inicio antes de que puedan pasar al modo coord o teleop.
- *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

Hay cuatro componentes contenidos en la Arquitectura LinuxCNC: un controlador de movimiento (EMCMOT), un controlador de E/S discreto (EMCIO), un ejecutor de tareas que los coordina (EMCTASK) y varios interfaces de usuario en modos texto y gráficos. Cada uno de ellos se describirá en el presente documento, tanto desde el punto de vista del diseño como del punto de vista de los desarrolladores (dónde encontrar los datos necesarios, cómo ampliar/modificar cosas fácilmente, 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. Arquitectura del software LinuxCNC

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.

Este documento asume que el lector tiene una comprensión básica de HAL, y comprende términos como pines HAL, señales HAL, etc., por lo que no se explican. Para obtener más información sobre HAL, consulte el Manual HAL. Otro capítulo de este documento entrará eventualmente en las interioridades del propio HAL, pero en este capítulo, solo usamos la API HAL, definida en src/hal/hal.h.

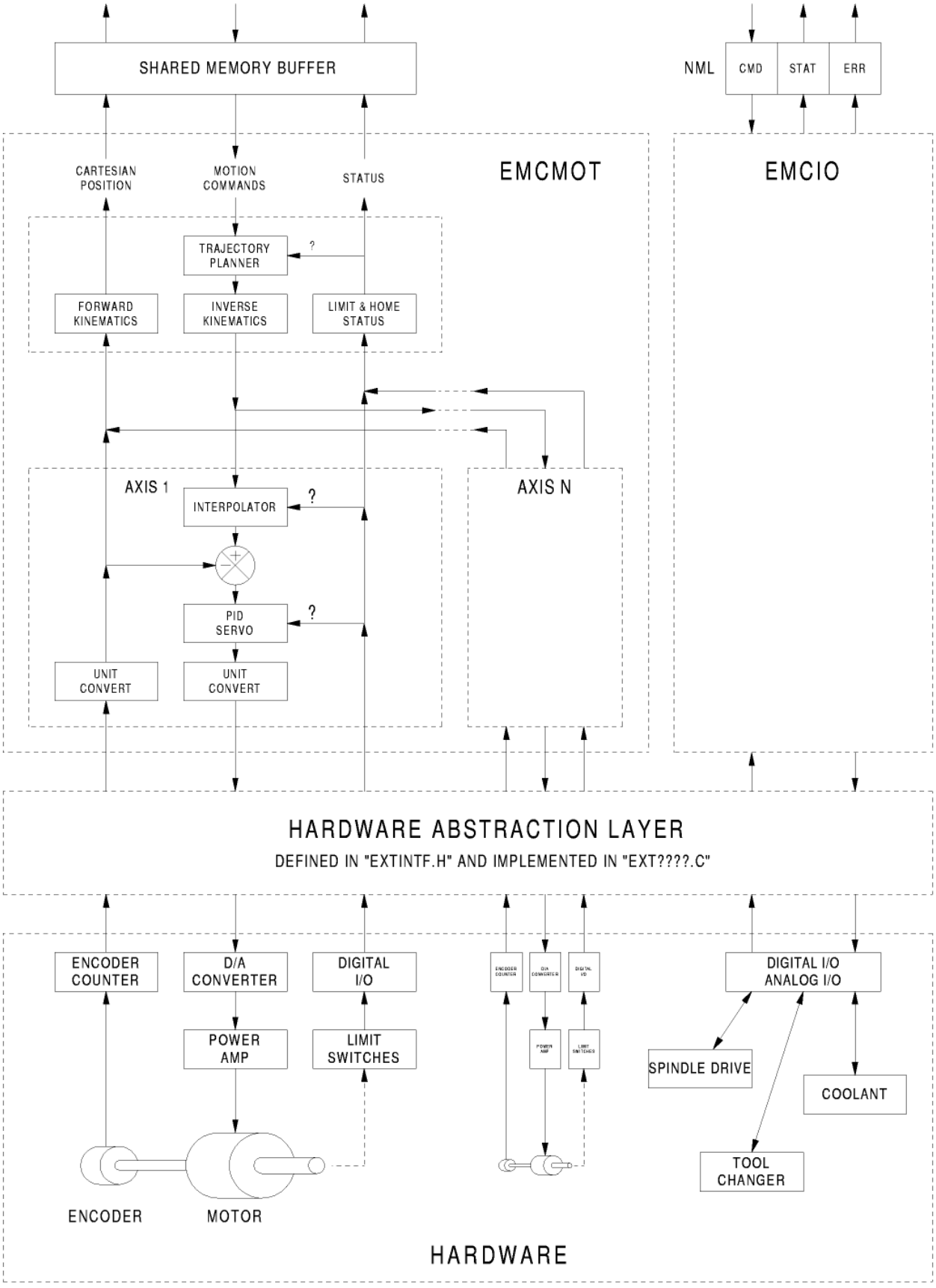
3.5.1. Motion Controller Modules

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

La siguiente figura es un diagrama de bloques de un controlador articular. Hay un controlador por articulación. Los controladores articulares funcionan a un nivel más bajo que la cinemática; un nivel donde todas las articulaciones son completamente independientes. Todos los datos para una articulación está en una sola estructura articular. Algunos miembros de esa estructura son visible en el diagrama de bloques, como coarse_pos, pos_cmd y motor_pos_fb.

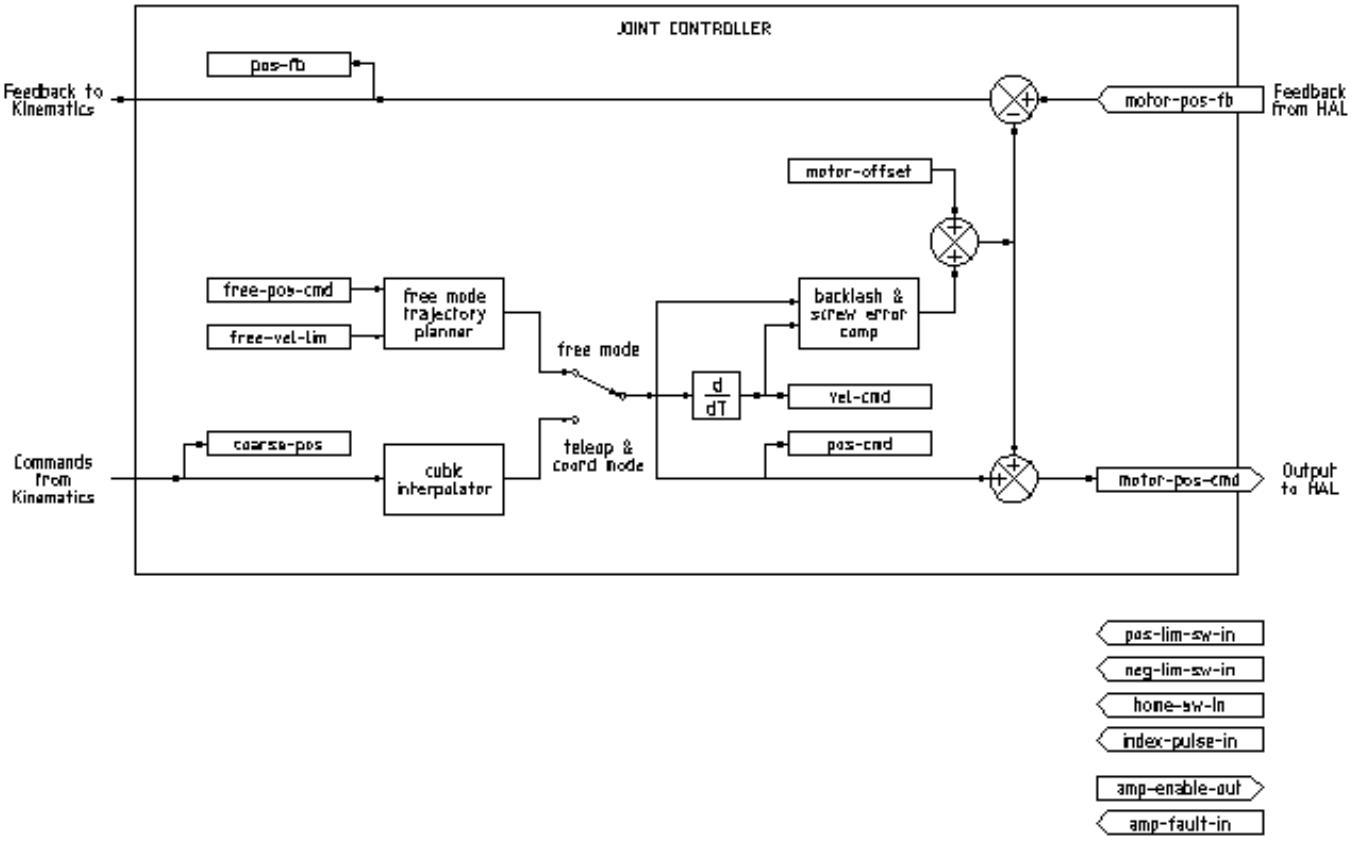


Figura 3.1: Joint Controller Block Diagram

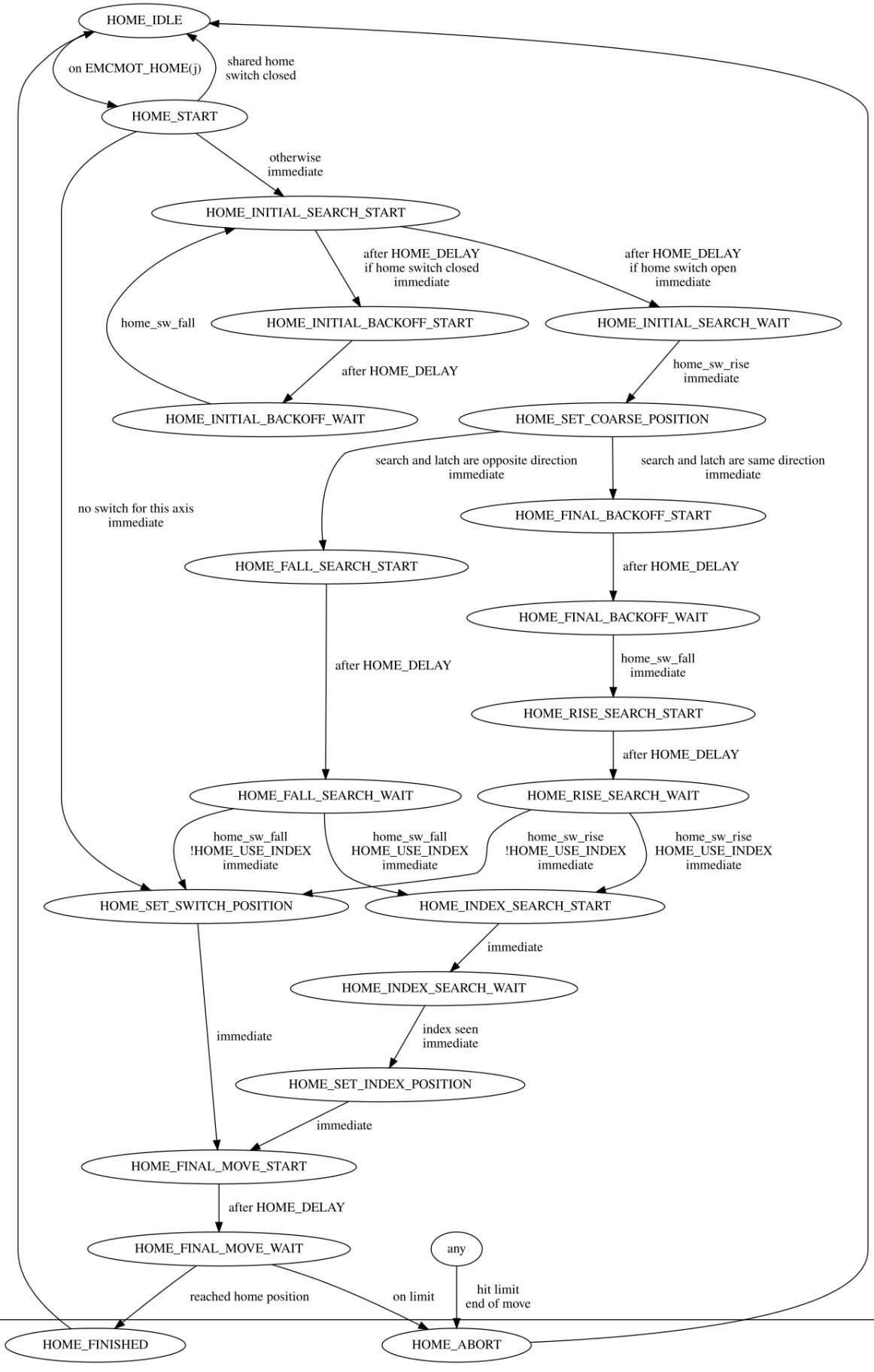
The above figure shows five of the seven sets of position information that form the main data flow through the motion controller. The seven forms of position data are as follows:

- *emcmotStatus->carte_pos_cmd* - Esta es la posición deseada, en coordenadas cartesianas. Se actualiza a tasa traj, no a tasa servo. En modo coord, se determina por el planificador traj. En modo teleop, está determinado por el planificador traj?. En modo libre, es copiado de actualPos, o generado mediante la aplicación de cinemática directa a (2) o (3).
- *emcmotStatus->joints[n].coarse_pos* - Esta es la posición deseada, en coordenadas articulares, pero antes de interpolación. Se actualiza a tasa traj, no a tasa servo. En modo coord, se genera aplicando cinematica inversa a (1). En modo teleop, se genera aplicando cinemática inversa a (1). En modo libre, creo que se copia de (3).

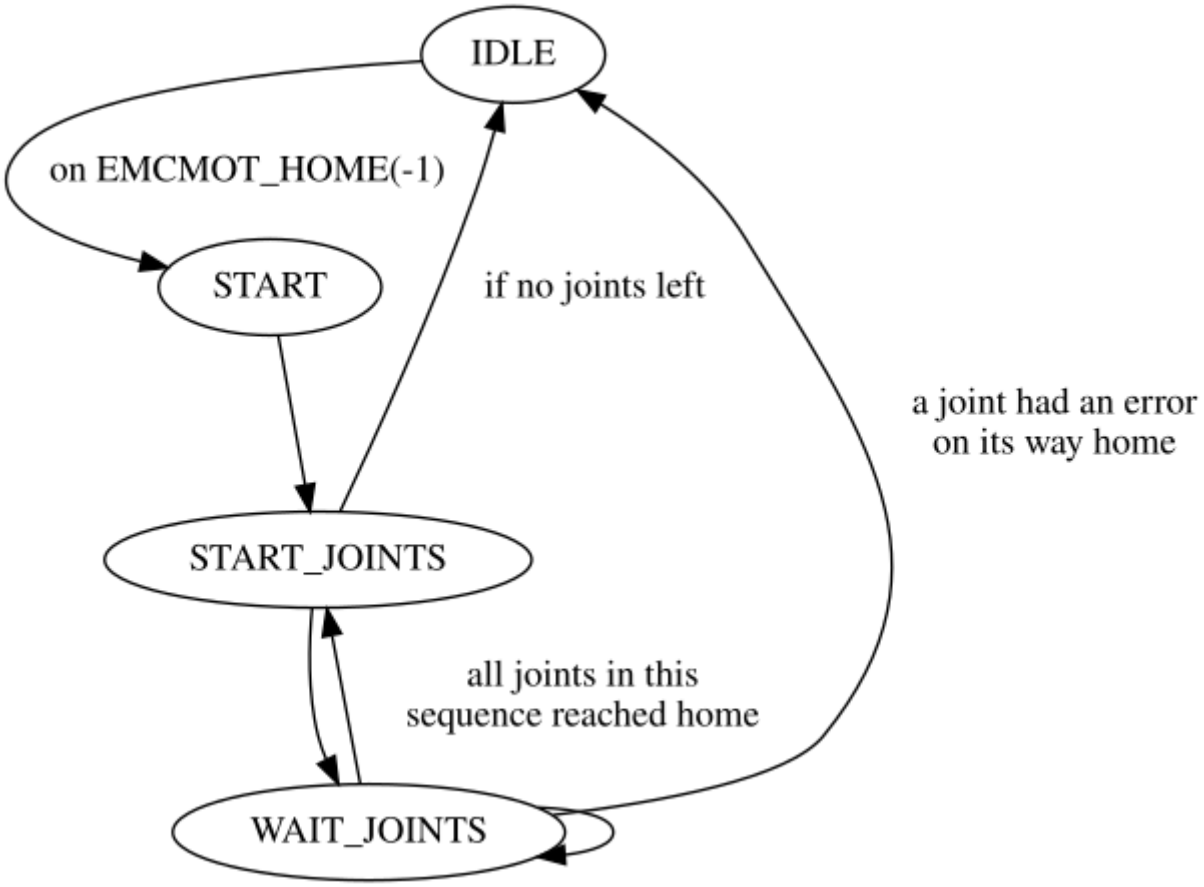
- ’emcmotStatus->joints[n].pos_cmd - This is the desired position, in joint coords, after interpolation. A new set of these coords is generated every servo period. In coord mode, it is generated from (2) by the interpolator. In teleop mode, it is generated from (2) by the interpolator. In free mode, 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* - Esta es la posición real, en coordenadas de motor. Es la entrada de codificadores u otro dispositivo de retroalimentación (o desde codificadores virtuales en máquinas de bucle abierto). Es ”generado” por la lectura del dispositivo de retroalimentación.
- *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.

---

nota
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

El comando ABORT simplemente detiene todo movimiento. Se puede emitir en cualquier momento y siempre será aceptado. No deshabilita el controlador de movimiento ni cambia ninguna información de estado; simplemente cancela cualquier movimiento que esté actualmente en progreso. Nota al pie:[Parece que el código de nivel superior (TASK y superior) también usa ABORT para borrar fallos. Siempre que haya un fallo persistente (como estar fuera de interruptores hardware de límite), el

código de nivel superior envía un flujo constante de ABORTs al controlador de movimiento en su intento de sobrepasar el fallo. Miles de ellos ... Eso significa que el controlador de movimiento debe evitar fallos persistentes. Esto necesita ser investigado.]

3.8.1.1. Requirements

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

3.8.1.2. Results

En modo libre, los planificadores de trayectoria de modo libre quedan deshabilitados. Esto da como resultado que cada articulación se detenga tan rápido como su límite de aceleración (desaceleración) permita. La parada no está coordinada. En modo teleop, la velocidad cartesiana comandada se establece a cero. No sé exactamente qué tipo de parada resulta (coordinada, descoordinada, etc.), pero lo resolveré finalmente. En modo coord, se le dice al planificador de trayectoria del modo coord que aborte el movimiento actual. De nuevo, no sé el resultado exacto de esto, pero lo documentaré cuando lo resuelva.

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

El controlador de comandos no aplica requisitos al comando FREE, siempre será aceptado. Sin embargo, si alguna articulación está en movimiento (GET_MOTION_INPOS_FLAG () == FALSE), entonces el comando será ignorado. Este comportamiento está controlado por un código que ahora se encuentra en la función *set_operating_mode()* en control.c, ese código debe limpiarse. Creo que el comando no debe ignorarse en silencio, sino que el controlador de comandos debe determinar si se puede ejecutar y devolver un error si no puede.

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

El comando TELEOP coloca la máquina en modo de teleoperación. En teleop modo, el movimiento de la máquina se basa en coordenadas cartesianas utilizando cinemática, en lugar de en articulaciones individuales como en modo libre. Sin embargo, el planificador de trayectoria per se no se usa, en cambio el movimiento es controlado por un vector de velocidad. El movimiento en modo teleop es muy parecido a trotar, excepto que se hace en espacio cartesiano en lugar de espacio de articulación. En una máquina con cinemática trivial, hay pequeña diferencia entre el modo teleop y el modo libre, y las GUI para esas máquinas podrían ni siquiera emitir este comando. Sin embargo, para máquinas no triviales como robots y hexápodos, el modo teleop se utiliza para la mayoría de los movimientos tipo jog ordenados por usuario.

3.8.3.1. Requirements

El controlador de comandos rechazará el comando TELEOP con un error mensaje si la cinemática no se puede activar porque una o más articulaciones no han sido dirigidas. Además, si alguna articulación está en movimiento (GET_MOTION_INPOS_FLAG () == FALSE), entonces el comando será ignorado (sin mensaje de error). Este comportamiento está controlado por un código que ahora esta ubicado en la función *set_operating_mode()* en control.c. Creo que el comando no debe ser ignorado en silencio, sino el controlador del comando debe determinar si se puede ejecutar y devolver un error si no puede.

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

El controlador de comandos rechazará el comando COORD con un error mensaje si la cinemática no se puede activar porque una o más articulaciones no han sido dirigidas. Además, si alguna articulación está en movimiento (GET_MOTION_INPOS_FLAG () == FALSE), entonces el comando será ignorado (sin mensaje de error). Este comportamiento está controlado por un código que ahora esta ubicado en la función *set_operating_mode()* en control.c. Creo que el comando no debe ser ignorado en silencio, sino que el controlador de comando debe determinar si se puede ejecutar y devolver un error si no puede.

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

El comando ACTIVATE_JOINT activa todos los cálculos asociados con una sola articulación, pero no cambia el pin de salida de habilitación del amplificador de la articulación.

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

El comando DEACTIVATE_JOINT desactiva todos los cálculos asociados con una sola articulación, pero no cambia la salida del pin de habilitación del amplificador de la articulación.

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

Actualmente nada. El antiguo watchdog era una cosa extraña que utilizaba una tarjeta de sonido específica. Es posible que en el futuro se diseñe una nueva interfaz de watchdog.

3.8.12. DISABLE_WATCHDOG

The DISABLE_WATCHDOG command disables a hardware based watchdog (if present).

3.8.12.1. Requirements

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

3.8.12.2. Results

Actualmente nada. El antiguo watchdog era una cosa extraña que utilizaba una tarjeta de sonido específica. Es posible que en el futuro se diseñe una nueva interfaz de watchdog.

3.8.13. PAUSE

The PAUSE command stops the trajectory planner. It has no effect in free or teleop mode. At this point I don't know if it pauses all motion immediately, or if it completes the current move and then pauses before pulling another move from the queue.

3.8.13.1. Requirements

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

3.8.13.2. Results

The trajectory planner pauses.

3.8.14. RESUME

The RESUME command restarts the trajectory planner if it is paused. It has no effect in free or teleop mode, or if the planner is not paused.

3.8.14.1. Requirements

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

3.8.14.2. Results

The trajectory planner resumes.

3.8.15. STEP

The STEP command restarts the trajectory planner if it is paused, and tells the planner to stop again when it reaches a specific point. It has no effect in free or teleop mode. At this point I don't know exactly how this works. I'll add more documentation here when I dig deeper into the trajectory planner.

3.8.15.1. Requirements

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

3.8.15.2. Results

The trajectory planner resumes, and later pauses when it reaches a specific point.

3.8.16. SCALE

The SCALE command scales all velocity limits and commands by a specified amount. It is used to implement feed rate override and other similar functions. The scaling works in free, teleop, and coord modes, and affects everything, including homing velocities, etc. However, individual joint velocity limits are unaffected.

3.8.16.1. Requirements

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

3.8.16.2. Results

All velocity commands are scaled by the specified constant.

3.8.17. OVERRIDE_LIMITS

El comando OVERRIDE_LIMITS evita que los límites se disparen hasta el fin del siguiente comando JOG. Normalmente se usa para permitir que una máquina salga de un interruptor de límite después de disparar. (El comando puede usarse en realidad para anular límites o para cancelar una anulación anterior.)

3.8.17.1. Requirements

Ninguna. El comando puede emitirse en cualquier momento y siempre será aceptado. (Creo que solo debería funcionar en modo libre.)

3.8.17.2. Results

Los límites en todas las articulaciones se anulan hasta el final del próximo comando JOG. (Esto está roto actualmente ... una vez que se recibe un comando OVERRIDE_LIMITS, los límites se ignoran hasta que otro comando OVERRIDE_LIMITS los vuelve a habilitar.)

3.8.18. HOME

The HOME command initiates a homing sequence on a specified joint. The actual homing sequence is determined by a number of configuration parameters, and can range from simply setting the current position to zero, to a multi-stage search for a home switch and index pulse, followed by a move to an arbitrary home location. For more information about the homing sequence, see the homing section of the Integrator Manual.

3.8.18.1. Requirements

The command will be ignored silently unless the machine is in free mode.

3.8.18.2. Results

Any jog or other joint motion is aborted, and the homing sequence starts.

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

The SET_CIRCLE command adds a circular move to the trajectory planner queue.

(More later)

3.8.24. SET_TELEOP_VECTOR

The SET_TELEOP_VECTOR command instructs the motion controller to move along a specific vector in Cartesian space.

(More later)

3.8.25. PROBE

The PROBE command instructs the motion controller to move toward a specific point in Cartesian space, stopping and recording its position if the probe input is triggered.

(More later)

3.8.26. CLEAR_PROBE_FLAG

The CLEAR_PROBE_FLAG command is used to reset the probe input in preparation for a PROBE command. (Question: why shouldn't the PROBE command automatically reset the input?)

(More later)

3.8.27. SET_xix

Hay aproximadamente 15 comandos SET_xxx, donde xxx es el nombre de algún parámetro de configuración. Se anticipa que habrá varios comandos SET mas a medida que se agregan más parámetros. Me gustaría encontrar una forma más limpia de establecer y leer los parámetros de configuración. Los métodos existentes requieren que se agreguen muchas líneas de código a múltiples archivos cada vez que se agrega un parámetro. Gran parte de ese código es idéntico o casi idéntico para cada parámetro.

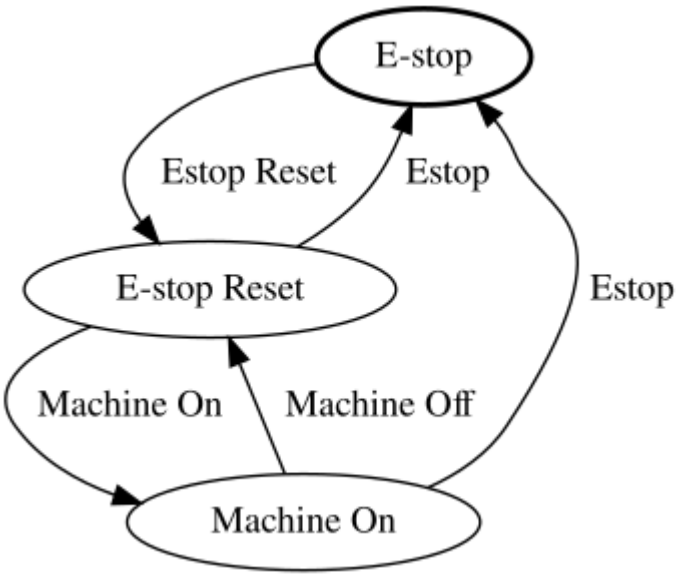
3.9. Backlash and Screw Error Compensation

FIXME Compensación de holguras y errores de tornillos

3.10. Task controller (EMCTASK)

3.10.1. State

Task has three possible internal states: **E-stop**, **E-stop Reset**, and **Machine On**.



3.11. IO controller (EMCIO)

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

Currently ESTOP/Enable, coolant, and tool changing are handled by iocontrol. These are relatively low speed events, high speed coordinated I/O is handled in motion.

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

iocontrol main loop process:

- checks to see it HAL inputs have changed
- checks if read_tool_inputs() indicates the tool change is finished and set emcioStatus.status

3.12. Interfaces de usuario

FIXME User Interfaces

3.13. libnml Introduction

libnml se deriva de rcslib del NIST sin todos los apoyos para otras plataformas. Muchos de los contenedores del código específico de plataformas han sido eliminados, junto con gran parte del código que no es requerido por LinuxCNC. Se espera que quede suficiente compatibilidad con rcslib para que las aplicaciones puedan implementarse en plataformas que no sean Linux y aún ser capaz de comunicarse con 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

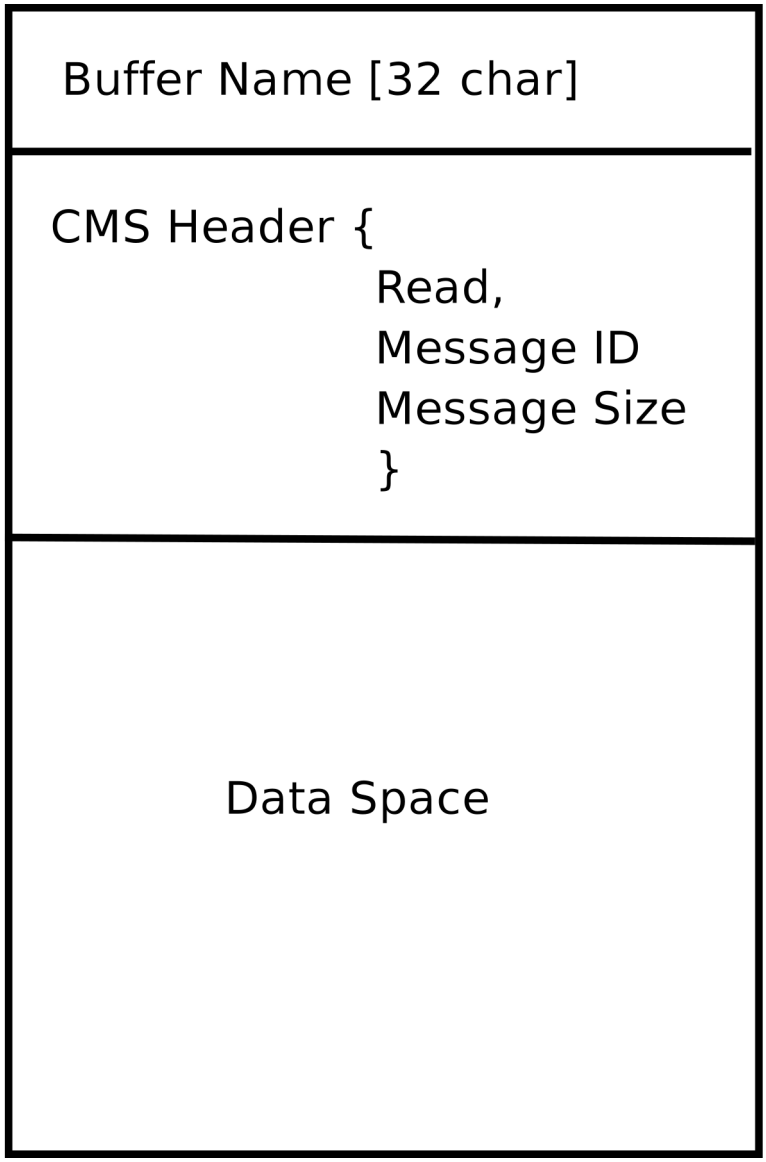
The Semaphore class provides a method of mutual exclusions for accessing a shared resource. The function to get a semaphore can either block until access is available, return after a timeout, or return immediately with or without gaining the semaphore. The constructor will create a semaphore or attach to an existing one if the ID is already in use.

The Semaphore::destroy() must be called by the last process only.

3.20. CMS

At the heart of libnml is the CMS class, it contains most of the functions used by libnml and ultimately NML. Many of the internal functions are overloaded to allow for specific hardware dependent methods of data passing. Ultimately, everything revolves around a central block of memory (referred to as the *message buffer* or just *buffer*). This buffer may exist as a shared memory block accessed by other CMS/NML processes, or a local and private buffer for data being transferred by network or serial interfaces.

The buffer is dynamically allocated at run time to allow for greater flexibility of the CMS/NML subsystem. The buffer size must be large enough to accommodate the largest message, a small amount for internal use and allow for the message to be encoded if this option is chosen (encoded data will be covered later). The following figure is an internal view of the buffer space.



CMS buffer The CMS base class is primarily responsible for creating the communications pathways and interfacing to the operating system.

3.21. Configuration file format

NML configuration consists of two types of line formats. One for Buffers, and a second for Processes that connect to the buffers.

3.21.1. Buffer line

The original NIST format of the buffer line is:

- *B name type host size neut RPC# buffer# max_procs key [type specific configs]*
- *B* - identifies this line as a Buffer configuration.
- *name* - is the identifier of the buffer.

- *type* - describes the buffer type - SHMEM, LOCMEM, FILEMEM, PHANTOM, or GLOBMEM.
- *host* - is either an IP address or host name for the NML server
- *size* - is the size of the buffer
- *neut* - a boolean to indicate if the data in the buffer is encoded in a machine independent format, or raw.
- *RPC#* - Obsolete - Place holder retained for backward compatibility only.
- *buffer#* - A unique ID number used if a server controls multiple buffers.
- *max_procs* - is the maximum processes allowed to connect to this buffer.
- *key* - is a numerical identifier for a shared memory buffer

3.21.2. Type specific configs

The buffer type implies additional configuration options whilst the host operating system precludes certain combinations. In an attempt to distill published documentation in to a coherent format, only the **SHMEM** buffer type will be covered.

- *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* - defines whether this process is local or remote relative to the buffer.
- *host* - specifies where on the network this process is running.
- *ops* - gives the process read only, write only, or read/write access to the buffer.
- *server* - specifies if this process will running a server for this buffer.
- *timeout* - sets the timeout characteristics for accesses to the buffer.
- *master* - indicates if this process is responsible for creating and destroying the buffer.
- *c_num* - an integer between zero and (max_procs -1)

3.21.4. Configuration Comments

Some of the configuration combinations are invalid, whilst others imply certain constraints. On a Linux system, GLOBMEM is obsolete, whilst PHANTOM is only really useful in the testing stage of an application, likewise for FILEMEM. LOCMEM is of little use for a multi-process application, and only offers limited performance advantages over SHMEM. This leaves SHMEM as the only buffer type to use with LinuxCNC.

The neut option is only of use in a multi-processor system where different (and incompatible) architectures are sharing a block of memory. The likelihood of seeing a system of this type outside of a museum or research establishment is remote and is only relevant to GLOBMEM buffers.

The RPC number is documented as being obsolete and is retained only for compatibility reasons.

With a unique buffer name, having a numerical identity seems to be pointless. Need to review the code to identify the logic. Likewise, the key field at first appears to be redundant, and it could be derived from the buffer name.

The purpose of limiting the number of processes allowed to connect to any one buffer is unclear from existing documentation and from the original source code. Allowing unspecified multiple processes to connect to a buffer is no more difficult to implement.

The mutex types boil down to one of two, the default ”os_sem” or ”mao split”. Most of the NML messages are relatively short and can be copied to or from the buffer with minimal delays, so split reads are not essential.

La codificación de datos solo es relevante cuando se transmite a un proceso remoto - Usar TCP o UDP implica codificación XDR. La codificación ASCII puede tener algún uso en diagnósticos o para pasar datos a un sistema integrado que no implementa NML.

UDP protocols have fewer checks on data and allows a percentage of packets to be dropped. TCP is more reliable, but is marginally slower.

If LinuxCNC is to be connected to a network, one would hope that it is local and behind a firewall. About the only reason to allow access to LinuxCNC via the Internet would be for remote diagnostics - This can be achieved far more securely using other means, perhaps by a web interface.

El comportamiento exacto cuando timeout se establece en cero o un valor negativo no está claro desde los documentos del NIST. Solo se mencionan valores INF y positivos. Sin embargo, enterrado en el código fuente de rcslib, parece aplicar lo siguiente:

timeout > 0
Blocking access until the timeout interval is reached or access to the buffer is available.

timeout = 0
Access to the buffer is only possible if no other process is reading or writing at the time.

timeout < 0 or INF
Access is blocked until the buffer is available.

3.22. NML base class

Expand on the lists and the relationship between NML, NMLmsg, and the lower level cms classes.

Not to be confused with NMLmsg, RCS_STAT_MSG, or RCS_CMD_MSG.

NML es responsable de analizar el archivo de configuración, configurar los búfers cms y el mecanismo para enrutar mensajes al(los) búfer(s) correcto(s). Para hacer esto, NML crea varias listas para:

- cms buffers created or connected to.
- processes and the buffers they connect to
- a long list of format functions for each message type

This last item is probably the nub of much of the malignment of libnml/rcslib and NML in general. Each message that is passed via NML requires a certain amount of information to be attached in addition to the actual data. To do this, several formatting functions are called in sequence to assemble fragments of the overall message. The format functions will include NML_TYPE, MSG_TYPE, in addition to the data declared in derived NMLmsg classes. Changes to the order in which the formatting functions are called and also the variables passed will break compatibility with rcslib if messed with - There are reasons for maintaining rcslib compatibility, and good reasons for messing with the code. The question is, which set of reasons are overriding?

3.22.1. NML internals

3.22.1.1. NML constructor

NML::NML() parses the config file and stores it in a linked list to be passed to cms constructors in single lines. It is the function of the NML constructor to call the relevant cms constructor for each buffer and maintain a list of the cms objects and the processes associated with each buffer.

It is from the pointers stored in the lists that NML can interact with cms and why Doxygen fails to show the real relationships involved.

---

nota
The config is stored in memory before passing a pointer to a specific line to the cms constructor. The cms constructor then parses the line again to extract a couple of variables... It would make more sense to do ALL the parsing and save the variables in a struct that is passed to the cms constructor - This would eliminate string handling and reduce duplicate code in cms...

3.22.1.2. NML read/write

Calls to NML::read and NML::write both perform similar tasks in so much as processing the message - The only real variation is in the direction of data flow.

Una llamada a la función de lectura primero obtiene datos del búfer y luego llama a format_output(), mientras que una función de escritura llamaría a format_input() antes de pasar los datos al búfer. El trabajo de construir o deconstruir el mensaje está dentro de format_xxx(). Una lista de funciones variadas se llama a su vez para colocar varias partes del encabezado NML (que no debe confundirse con el encabezado cms) en el orden correcto - La última función llamada es emcFormat() en emc.cc.

3.22.1.3. NMLmsg and NML relationships

NMLmsg is the base class from which all message classes are derived. Each message class must have a unique ID defined (and passed to the constructor) and also an update(*cms) function. The update() will be called by the NML read/write functions when the NML formatter is called — the pointer to the formatter will have been declared in the NML constructor at some point. By virtue of the linked lists NML creates, it is able to select cms pointer that is passed to the formatter and therefore which buffer is to be used.

3.23. Adding custom NML commands

LinuxCNC is pretty awesome, but some parts need some tweaking. As you know communication is done through NML channels, the data sent through such a channel is one of the classes defined in emc.hh (implemented in emc.cc). If somebody needs a message type that doesn't exist, he should follow these steps to add a new one. (The Message I added in the example is called EMC_IO_GENERIC (inherits EMC_IO_CMD_MSG (inherits RCS_CMD_MSG)))

1. add the definition of the EMC_IO_GENERIC class to emc2/src/emc/nml_intf/emc.hh
2. add the type define: #define EMC_IO_GENERIC_TYPE \((NMLTYPE) 1605)

    a. (I chose 1605, because it was available) to emc2/src/emc/nml_intf/emc.hh
3. add case EMC_IO_GENERIC_TYPE to emcFormat in emc2/src/emc/nml_intf/emc.cc
4. add case EMC_IO_GENERIC_TYPE to emc_symbol_lookup in emc2/src/emc/nml_intf/emc.cc
5. add EMC_IO_GENERIC::update function to emc2/src/emc/nml_intf/emc.cc

Recompile, and the new message should be there. The next part is to send such messages from somewhere, and receive them in another place, and do some stuff with it.

3.24. The Tool Table and Toolchanger

LinuxCNC interactúa con el hardware del cambiador de herramientas y tiene una abstracción interna del mismo. LinuxCNC gestiona la información de la herramienta en un archivo de tabla de herramientas.

3.24.1. Toolchanger abstraction in LinuxCNC

LinuxCNC supports two kinds of toolchanger hardware, called *nonrandom* and *random*. The INI setting [EMCIO]RANDOM_TOOLCHANGER controls which of these kinds of hardware LinuxCNC thinks it is connected to.

3.24.1.1. Nonrandom Toolchangers

Nonrandom toolchanger hardware puts each tool back in the pocket it was originally loaded from.

Examples of nonrandom toolchanger hardware are the ”manual” toolchanger, lathe tool turrents, and rack toolchangers.

When configured for a nonrandom toolchanger, LinuxCNC does not change the pocket number in the tool table file as tools are loaded and unloaded. Internal to LinuxCNC, on tool change the tool information is **copied** from the tool table’s source pocket to pocket 0 (which represents the spindle), replacing whatever tool information was previously there.

---

nota
In LinuxCNC configured for nonrandom toolchanger, tool 0 (T0) has special meaning: ”no tool”. T0 may not appear in the tool table file, and changing to T0 will result in LinuxCNC thinking it has got an empty spindle.

3.24.1.2. Random Toolchangers

Random toolchanger hardware swaps the tool in the spindle (if any) with the requested tool on tool change. Thus the pocket that a tool resides in changes as it is swapped in and out of the spindle.

An example of random toolchanger hardware is a carousel toolchanger.

When configured for a random toolchanger, LinuxCNC swaps the pocket number of the old and the new tool in the tool table file when tools are loaded. Internal to LinuxCNC, on tool change, the tool information is **swapped** between the tool table’s source pocket and pocket 0 (which represents the spindle). So after a tool change, pocket 0 in the tool table has the tool information for the new tool, and the pocket that the new tool came from has the tool information for the old tool (the tool that was in the spindle before the tool change), if any.

---

nota
If LinuxCNC is configured for random toolchanger, tool 0 (T0) has **no** special meaning. It is treated exactly like any other tool in the tool table. It is customary to use T0 to represent ”no tool” (i.e., a tool with zero TLO), so that the spindle can be conveniently emptied when needed.

3.24.2. The Tool Table

LinuxCNC keeps track of tools in a file called the tool table. The tool table records the following information for each tool:

tool number
An integer that uniquely identifies this tool. Tool numbers are handled differently by LinuxCNC when configured for random and nonrandom toolchangers:

- When LinuxCNC is configured for a nonrandom toolchanger this number must be positive. T0 gets special handling and is not allowed to appear in the tool table.

- When LinuxCNC is configured for a random toolchanger this number must be non-negative. T0 is allowed in the tool table, and is usually used to represent ”no tool”, i.e. the empty pocket.

pocket number
An integer that identifies the pocket or slot in the toolchanger hardware where the tool resides. Pocket numbers are handled differently by LinuxCNC when configured for random and nonrandom toolchangers:

- When LinuxCNC is configured for a nonrandom toolchanger, the pocket number in the tool file can be any positive integer (pocket 0 is not allowed). LinuxCNC silently compactifies the pocket numbers when it loads the tool file, so there may be a difference between the pocket numbers in the tool file and the internal pocket numbers used by LinuxCNC-with-nonrandom-toolchanger.
- When LinuxCNC is configured for a random toolchanger, the pocket numbers in the tool file must be between 0 and 1000, inclusive. Pockets 1-1000 are in the toolchanger, pocket 0 is the spindle.

diameter
Diameter of the tool, in machine units.

tool length offset
Tool length offset (also called TLO), in up to 9 axes, in machine units. Axes that don’t have a specified TLO get 0.

3.24.3. G-codes affecting tools

The G-codes that use or affect tool information are:

3.24.3.1. Txxx

Tells the toolchanger hardware to prepare to switch to a specified tool `xxx`.

Handled by `Interp::convert_tool_select()`.

1. The machine is asked to prepare to switch to the selected tool by calling the Canon function `SELECT_TOOL()` with the tool number of the requested tool.
   a. (saicanon) No-op.
   b. (emccanon) Builds an `EMC_TOOL_PREPARE` message with the requested pocket number and sends it to Task, which sends it on to IO. IO gets the message and asks HAL to prepare the pocket by setting `iocontrol.0.tool-prep-pocket`, `iocontrol.0.tool-prep-number`, and `iocontrol.0.tool-prepare`. IO then repeatedly calls `read_tool_inputs()` to poll the HAL pin `iocontrol.0.tool-prepared`, which signals from the toolchanger hardware, via HAL, to IO that the requested tool prep is complete. When that pin goes True, IO sets `emcioStatus.tool.pocketPrepped` to the requested tool’s pocket number.
2. Back in interp, `settings->selected_pocket` is assigned the tooldata index of the requested tool *xxx*.

---

nota
The legacy names **selected_pocket** and **current_pocket** actually reference a sequential tooldata index for tool items loaded from a tool table ([EMCIO]TOOL_TABLE) or via a tooldata database ([EMCIO]DB_PROGRAM).

3.24.3.2. M6

Tells the toolchanger to switch to the currently selected tool (selected by the previous Txxx command).

Handled by `Interp::convert_tool_change()`.

1. The machine is asked to change to the selected tool by calling the Canon function `CHANGE_TOOL()` with `settings->selected_pocket` (a tooldata index).
   a. (saicanon) Sets sai's `_active_slot` to the passed-in pocket number. Tool information is copied from the selected pocket of of the tool table (ie, from sai's `_tools[_active_slot]`) to the spindle (aka sai's `_tools[0]`).
   b. (emccanon) Sends an `EMC_TOOL_LOAD` message to Task, which sends it to IO. IO sets `emcioStatus.t` to the tool number of the tool in the pocket identified by `emcioStatus.tool.pocketPrepped` (set by `Txxx` aka `SELECT_TOOL()`). It then requests that the toolchanger hardware perform a tool change, by setting the HAL pin `iocontrol.0.tool-change` to True. Later, IO's `read_tool_inpu` will sense that the HAL pin `iocontrol.0.tool_changed` has been set to True, indicating the toolchanger has completed the tool change. When this happens, it calls `load_tool()` to update the machine state.
      i. `load_tool()` with a nonrandom toolchanger config copies the tool information from the selected pocket to the spindle (pocket 0).
      ii. `load_tool()` with a random toolchanger config swaps tool information between pocket 0 (the spindle) and the selected pocket, then saves the tool table.
2. Back in interp, `settings->current_pocket` is assigned the new tooldata index from `settings->select` (set by `Txxx`). The relevant numbered parameters (#5400-#5413) are updated with the new tool information from pocket 0 (spindle).

3.24.3.3. G43/G43.1/G49

Apply tool length offset. G43 uses the TLO of the currently loaded tool, or of a specified tool if the H-word is given in the block. G43.1 gets TLO from axis-words in the block. G49 cancels the TLO (it uses 0 for the offset for all axes).

Handled by `Interp::convert_tool_length_offset()`.

1. It starts by building an `EmcPose` containing the 9-axis offsets to use. For `G43.1`, these tool offsets come from axis words in the current block. For `G43` these offsets come from the current tool (the tool in pocket 0), or from the tool specified by the H-word in the block. For G49, the offsets are all 0.
2. The offsets are passed to Canon's `USE_TOOL_LENGTH_OFFSET()` function.
   a. (saicanon) Records the TLO in `_tool_offset`.
   b. (emccanon) Crea un mensaje `EMC_TRAJ_SET_OFFSET` que contiene los offsets y lo envía a Task, que copia las compensaciones en `emcStatus->task.toolOffset` y los envía a Motion a través de un comando `EMCMOT_SET_OFFSET`. Motion copia las compensaciones a `emcmotStatus->tool_off` donde se usa para compensar movimientos futuros.
3. Back in interp, the offsets are recorded in `settings->tool_offset`. The effective pocket is recorded in `settings->tool_offset_index`, though this value is never used.

3.24.3.4. G10 L1/L10/L11

Modifies the tool table.

Handled by `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. Con una configuración de cambiador de herramientas aleatorio, si la herramienta está actualmente cargada utiliza la ranura 0 (ranura 0 significa "el husillo"), y si la herramienta no está cargada, usa el número de ranura en el cambiador de herramientas. (Esta diferencia es importante.)
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.

Handled by `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-n` (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` es de tipo `settings`, definida como `struct setup_struct` en `src/emc/rs274ngc/interp_interna`

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.
- Sent to Motion via the `EMCMOT_SET_OFFSET` message. All motion does with the offsets is export them to the HAL pins `motion.0.tooloffset.[xyzabcuvw]`. FIXME: export these from some-place closer to the tool table (io or interp, probably) and remove the EMCMOT_SET_OFFSET message.

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

The Motion controller realtime component first gets the number of joints from the num_joints load-time parameter. This determines how many joints worth of HAL pins are created at startup.

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

The Motion controller always operates on EMCMOT_MAX_AXIS axes. It always creates nine sets of axis.*.* pins.

Capítulo 4

NML Messages

Lista de mensajes NML.
Para mas detalles, vea `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. Others

```
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
```

Capítulo 5

Coding Style

This chapter describes the source code style preferred by the LinuxCNC team.

5.1. Do no harm

When making small edits to code in a style different than the one described below, observe the local coding style. Rapid changes from one coding style to another decrease code readability.

Never check in code after running ”indent” on it. The whitespace changes introduced by indent make it more difficult to follow the revision history of the file.

No use un editor que realice cambios innecesarios en los espacios en blanco (por ejemplo, que reemplace 8 espacios con un tabulador en una línea no modificada, o ajuste de texto en líneas que no han sido modificadas).

5.2. Tab Stops

A tab stop always corresponds to 8 spaces. Do not write code that displays correctly only with a differing tab stop setting.

5.3. Indentation

Use 4 spaces per level of indentation. Combining 8 spaces into one tab is acceptable but not required.

5.4. Placing Braces

Put the opening brace last on the line, and put the closing brace first:

```
if (x) {
    // do something appropriate
}
```

The closing brace is on a line of its own, except in the cases where it is followed by a continuation of the same statement, i.e. a *while* in a do-statement or an *else* in an if-statement, like this:

```
do {
    // something important
} while (x > 0);
```

and

```
if (x == y) {
    // do one thing
} else if (x < y) {
    // do another thing
} else {
    // do a third thing
}
```

This brace-placement also minimizes the number of empty (or almost empty) lines, which allows a greater amount of code or comments to be visible at once in a terminal of a fixed size.

5.5. Naming

C is a Spartan language, and so should your naming be. Unlike Modula-2 and Pascal programmers, C programmers do not use cute names like ThisVariableIsATemporaryCounter. A C programmer would call that variable *tmp*, which is much easier to write, and not the least more difficult to understand.

However, descriptive names for global variables are a must. To call a global function *foo* is a shooting offense.

GLOBAL variables (to be used only if you **really** need them) need to have descriptive names, as do global functions. If you have a function that counts the number of active users, you should call that *count_active_users()* or similar, you should **not** call it *cntusr()*.

Codificar el tipo de una función en el nombre (llamada notación húngara) no tiene sentido; el compilador conoce los tipos de todos modos y puede verificarlos, y solo confunde al programador. No es de extrañar que Microsoft haga programas con errores.

LOCAL variable names should be short, and to the point. If you have some random integer loop counter, it should probably be called *i*. Calling it *loop_counter* is non-productive, if there is no chance of it being misunderstood. Similarly, *tmp* can be just about any type of variable that is used to hold a temporary value.

Si tiene miedo de mezclar sus nombres de variables locales, tiene otro problema, que se llama síndrome de desequilibrio de la hormona del crecimiento funcional. Ver el siguiente capítulo.

5.6. Functions

Las funciones deben ser cortas y fáciles, y hacer una sola cosa. Deben caber en una o dos pantallas de texto (el tamaño de pantalla ISO/ANSI es 80x24, como todos sabemos), hacer una cosa y hacerla bien.

The maximum length of a function is inversely proportional to the complexity and indentation level of that function. So, if you have a conceptually simple function that is just one long (but simple) case-statement, where you have to do lots of small things for a lot of different cases, it's OK to have a longer function.

However, if you have a complex function, and you suspect that a less-than-gifted first-year high-school student might not even understand what the function is all about, you should adhere to the maximum limits all the more closely. Use helper functions with descriptive names (you can ask the compiler to

in-line them if you think it's performance-critical, and it will probably do a better job of it that you would have done).

Otra medida de la función es el número de variables locales. No deben exceder de 5-10, o algo estás haciendo mal. Repensar la función, y dividirla en pedazos más pequeños. Un cerebro humano generalmente puede realizar un seguimiento de aproximadamente 7 cosas diferentes, cualquier cosa más y se confunde. Sabes que eres brillante, pero tal vez te gustaría entender lo que hiciste dentro de 2 semanas.

5.7. Commenting

Comments are good, but there is also a danger of over-commenting. NEVER try to explain HOW your code works in a comment: it's much better to write the code so that the **working** is obvious, and it's a waste of time to explain badly written code.

Generally, you want your comments to tell WHAT your code does, not HOW. A boxed comment describing the function, return value, and who calls it placed above the body is good. Also, try to avoid putting comments inside a function body: if the function is so complex that you need to separately comment parts of it, you should probably re-read the Functions section again. You can make small comments to note or warn about something particularly clever (or ugly), but try to avoid excess. Instead, put the comments at the head of the function, telling people what it does, and possibly WHY it does it.

If comments along the lines of /* Fix me */ are used, please, please, say why something needs fixing. When a change has been made to the affected portion of code, either remove the comment, or amend it to indicate a change has been made and needs testing.

5.8. Shell Scripts & Makefiles

Not everyone has the same tools and packages installed. Some people use vi, others emacs - A few even avoid having either package installed, preferring a lightweight text editor such as nano or the one built in to Midnight Commander.

gawk versus mawk - Again, not everyone will have gawk installed, mawk is nearly a tenth of the size and yet conforms to the POSIX AWK standard. If some obscure gawk specific command is needed that mawk does not provide, than the script will break for some users. The same would apply to mawk. In short, use the generic awk invocation in preference to gawk or mawk.

5.9. C++ Conventions

Los estilos de codificación de C++ siempre terminan en acalorados debates (un poco como los argumentos de emacs versus vi). Una cosa es cierta, sin embargo; el estilo común utilizado por todos los que trabajan en un proyecto conduce a la uniformidad y a código legible.

Convenciones de nomenclatura: las constantes #define o enumeraciones debe estar en mayúscula. Justificación: hace que sea más fácil detectar constantes de tiempo de compilación en el código fuente, p.ej. EMC_MESSAGE_TYPE.

Las clases y los espacios de nombres deben poner en mayúscula la primera letra de cada palabra y evitar guiones bajos. Justificación: identifica clases, constructores y destructores, p.ej. GtkWidget.

Los métodos (o nombres de funciones) deben seguir las recomendaciones C anteriores y no debe incluir el nombre de la clase. Justificación: mantiene un estilo común en fuentes C y C++, p.ej. get_foo_bar().

Sin embargo, los métodos booleanos son más fáciles de leer si se evitan los guiones bajos y usan un prefijo *is* (no debe confundirse con los métodos que manipulan un booleano). Justificación: identifica el valor de retorno como VERDADERO o FALSO y nada más, p.ej. isOpen, isHomed.

Do NOT use *Not* in a boolean name, it leads only leads to confusion when doing logical tests, e.g., isNotOnLimit or is_not_on_limit are BAD.

Los nombres de las variables deben evitar el uso de mayúsculas y guiones bajos a excepción de nombres locales o privados. El uso de variables globales debería evitarse tanto como sea posible. Justificación: aclara cuáles son variables y cuales son métodos. Ej. Público: axislimit. Ej. Privado: maxvelocity_ .

5.9.1. Specific method naming conventions

Los términos get y set deben usarse donde se accede a un atributo directamente. Justificación: indica el propósito de la función o método, p.ej. get_foo set_bar.

For methods involving boolean attributes, set & reset is preferred. Rationale: As above. e.g. set_amp_enable reset_amp_fault

Math intensive methods should use compute as a prefix. Rationale: Shows that it is computationally intensive and will hog the CPU. e.g. compute_PID

Abbreviations in names should be avoided where possible - The exception is for local variable names. Rationale: Clarity of code. e.g. pointer is preferred over ptr compute is preferred over cmp compare is again preferred over cmp.

Las enumeraciones y otras constantes pueden tener como prefijo un nombre de tipo común p.ej. `enum COLOR{COLOR_RED, COLOR_BLUE};`.

Se debe evitar el uso excesivo de macros y defines. Se prefieren métodos o funciones. Justificación: mejora el proceso de depuración.

Include Statements Header files must be included at the top of a source file and not scattered throughout the body. They should be sorted and grouped by their hierarchical position within the system with the low level files included first. Include file paths should NEVER be absolute - Use the compiler -I flag instead to extend the search path. Rationale: Headers may not be in the same place on all systems.

Los punteros y las referencias deben tener su símbolo de referencia junto al nombre de la variable y no junto al nombre del tipo. Justificación: Reduce la confusión, p.ej. `float *x` o `int &i`.

Las pruebas implícitas para cero no deben usarse excepto para variables booleanas, p.ej. `if (spindle_speed != 0)` NO `if (spindle_speed)`.

Solo las declaraciones de control de bucle deben incluirse en una construcción for(). p.ej. `sum = 0; for (i = 0; i < 10; i++) { sum += value[i]; }`
NO: `for (i=0,sum=0; i<10; i++) sum += value[I];`.

Likewise, executable statements in conditionals must be avoided, e.g., `if (fd = open(file_name))` is bad.

Complex conditional statements should be avoided - Introduce temporary boolean variables instead.

Los paréntesis deben usarse en abundancia en las expresiones matemáticas. - No confíe en la precedencia del operador cuando un paréntesis adicional aclararía las cosas.

File names: C++ sources and headers use .cc and .hh extension. The use of .c and .h are reserved for plain C. Headers are for class, method, and structure declarations, not code (unless the functions are declared inline).

5.10. Python coding standards

Utilice el estilo PEP 8 para Código de Python.

5.11. Comp coding standards

In the declaration portion of a .comp file, begin each declaration at the first column. Insert extra blank lines when they help group related items.

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

Capítulo 6

GUI Development Reference

This document attempts to be a *best practices* reference for general use screen development.
While it is possible to program just about anything to work with LinuxCNC, using a common frame work, language and configuration requirements allows easier transition between screens and more developers to maintain them.
That said, nothing in this document is written in stone.

6.1. Language

Python is currently the preferred language of LinuxCNC's screen code.
Python has a low entry bar for new users to modify the screens to suit them.
Python has a rich array of documentation, tutorials and libraries to pull from.
It is already used and integrated into LinuxCNC's system requirements.
While C or C++ could be used, it severely limits who can maintain and develop them.
It would be better to extend Python with C/C++ modules for whatever function that requires it.

6.2. Localization of float numbers in GUIs

Different locales use different decimal separators and thousands separators. Locale-specific string-to-float functions should be avoided as they may give unexpected results. (For example the text string "1.58" in de_DE will be converted to 158 by atof()). The following guidelines (based on avoiding ambiguity rather than on "correctness" in any specific locale) are suggested if parsing float to string and vice-versa:

- In the case of input allow either comma (,) or point(.) as a decimal separator, but reject any input that has more than one of either. Space should be accepted but not required as a thousands separator.
- In the case of display either use point (.) consistently or use the current localisation format consistently. The emphasis here being on "consistently".

6.3. Basic Configuration

Currently, most screens use a combination of INI file and preference file entries to configure their functions.
INI text files are usually used for the common machine controller settings, while text based preference

files are used for more GUI related properties (such as sounds, size, colors).
There can be other files used for translations, stylizing and function customization. These are highly dependent on the underlying widget toolkit.

6.3.1. INI [DISPLAY]

The **[DISPLAY]** section of the INI is for specifying screen related settings.

6.3.1.1. Display

The most important of is specifying the name of the screen that the LinuxCNC script will use to load.
The screen program usually recognizes switches such as to set full screen.
Title is for the window title and icon is used for iconizing the window.

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

6.3.1.2. Cycle Time

If settable, this is how to set the cycle time of the display GUI.
This is often the update rate rather then sleep time between updates.
A value of 100 ms (0.1 s) is a common setting though a range of 50 - 200 ms is not unheard of.

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

6.3.1.3. File Paths

If these functions are available in the screen here is how to specify the path to use.
These should reference from the current INI file, or allow ~ for the home folder, or allow use of absolute paths.

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

6.3.1.4. Jog Increments

Radio buttons or a combobox are generally used for increments selection.
The linear increments can be a mix of inches or millimeters.
Angular increments are specified in degrees.
The word *continuous* is used to specify continuous jogging and probably should be added even if left out of the INI line.

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

6.3.1.5. Machine Type Hint

The screen often needs to be adjusted based on machine type. Lathes have different controls and display DROs differently. Foam machine display the plot differently.
The old way to do this was adding switches LATHE = 1, FOAM = 1 etc

```
MACHINE_TYPE_HINT = LATHE
```

6.3.1.6. Ajuste manual

Overrides allows the user to adjust feed rate or spindle speed on the fly. Usually a slider or dial is used.
These settings are in percent.

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

6.3.1.7. Jog Rate

Most screens have slider controls to adjust the linear and angular jog speed rate,
These settings should be specified in machine units per minute for linear and degrees per minute for angular
Default refers to the starting rate when the screen is first loaded.

```
DEFAULT_LINEAR_VELOCITY =
MIN_LINEAR_VELOCITY =
MAX_LINEAR_VELOCITY =

DEFAULT_ANGULAR_VELOCITY =
MIN_ANGULAR_VELOCITY =
MAX_ANGULAR_VELOCITY =
```

6.3.1.8. Spindle Manual Controls

Manual controls for spindle control could be, (or a combinations of) buttons, sliders or dials
You can set limits that are less then the what the machine controller can utilize by setting these entries.
If your screen is capable of running multiple spindles, then should accept entries higher then the shown 0.

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

6.3.2. INI [MDI_COMMAND]

Some screens use buttons to run *Macro* NGC commands.
They can be specified like these compact examples.
NGC commands separated by colons are run to completion before the next.
The optional comma separates text for the button from the NGC code.

```
[MDI_COMMAND_LIST]
MDI_COMMAND_MACRO0 = G0 Z25;X0 Y0;Z0, Goto\nUser\nZero
MDI_COMMAND_MACRO1 = G53 G0 Z0;G53 G0 X0 Y0,Goto\nMachn\nZero
```

6.3.3. INI [FILTER]

This section allows setting of what files are shown in the file chooser and what filter programs will preprocess its output before sending it to LinuxCNC.
The extensions follow this pattern:
PROGRAM_EXTENSION = .extension,.extension2[space]Description of extensions
The filter program definitions are such:
filter extension = program to run

```
[FILTER]
# Controls what programs are shown in the file manager:
PROGRAM_EXTENSION = .ngc,.nc,.tap G-Code File (*.ngc,*.nc,*.tap)
PROGRAM_EXTENSION = .png,.gif,.jpg Greyscale Depth Image
PROGRAM_EXTENSION = .py Python Script

# Mapea datos/extensiones de archivo de código fuente a un programa 'filtro' especial para ↩
    la visualización/ejecución:
png = image-to-gcode
gif = image-to-gcode
jpg = image-to-gcode
py = python3
```

6.3.4. INI [HAL]

Most screens will need some HAL pins. They need to be connected after the screen creates them.

6.3.4.1. Postgui Halfile

These files should be run one after another in order, after all the GUI HAL pins have been made.

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

6.3.4.2. Postgui Halcmd

These files should be run one after another in order, after all the POSTGUI files have been run.

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

6.4. Extended Configuration

6.4.1. Embedding GUI Elements

Allowing users to build small panels independently, that can be embedded into the main screen is a common and very useful customization. Some screens allow embedding of 3rd party foreign programs, others only the native widget toolkit based panels.
Usually these are embedded in tabs or side panel widgets.
This is how to describe the optional title, loading command and location widget name:

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

6.4.2. User Message Dialogs

User dialogs are used for popping up import information (usually errors), that the user deems important.
Some stay up till the problem is fixed, some require acknowledgement, others a yes/no choice.
A HAL I/O pin would pop up the dialog, the dialog would reset the I/O pin and set any response output pins.

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

This style gives multiple messages defined by a number.
This example shows 3 possible messages based around a VFD error number.

```
[DISPLAY]
MULTIMESSAGE_ID = VFD

MULTIMESSAGE_VFD_NUMBER = 1
MULTIMESSAGE_VFD_TYPE = okdialog status
MULTIMESSAGE_VFD_TITLE = VFD Error: 1
MULTIMESSAGE_VFD_TEXT = This is the longer text FOR MESSAGE NUMBER 1
MULTIMESSAGE_VFD_DETAILS = DETAILS for VFD error 1
MULTIMESSAGE_VFD_ICON = WARNING

MULTIMESSAGE_VFD_NUMBER = 2
MULTIMESSAGE_VFD_TYPE = nonedialog status
MULTIMESSAGE_VFD_TITLE = VFD Error: 2
MULTIMESSAGE_VFD_TEXT = This is the longer text FOR MESSAGE NUMBER 2
MULTIMESSAGE_VFD_DETAILS = DETAILS for VFD error 2
MULTIMESSAGE_VFD_ICON = INFO

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

Capítulo 7

Construyendo LinuxCNC

7.1. Introducción

Este documento describe cómo construir el software LinuxCNC y la documentación desde las fuentes. Esto es útil principalmente si eres un desarrollador que está modificando LinuxCNC. También puede ser útil si eres un usuario que está probando ramas de desarrollo, aunque también se tiene la opción de instalar paquetes únicos Debian desde buildboot: (http://buildbot.linuxcnc.org) o como un paquete normal de tu distribución Linux (https://tracker.debian.org/pkg/linuxcnc). Ciertamente, esta sección también existe por que LinuxCNC es un esfuerzo comunitario y te alentamos a contribuir al desarrollo de LinuxCNC. Generalmente, querrás compilar LinuxCNC por ti mismo para un acceso funcional inmediato

- para un nuevo desarrollo de LinuxCNC o
- un desarrollo nuevo que quizás quieras contribuir a LinuxCNC o ayudar a otros a completarlo.

Por lo tanto, podrías portar LinuxCNC a una nueva distribución Linux, o como es común, un desarrollador reacciona a un problema que has reportado y cuya solución quieres probar. Cualquier cambio como tal no tendrá buildbot de ayuda, o esa ayuda estará desfasada, dependiendo de la revisión de alguien mas que no quieras esperar o eres el único individuo con un hardware en particular para probar el código.

Además de los programas que controlan tu máquina que están construidos desde el árbol fuente, también puedes construir los mismos archivos PDF y/o HTML que seguramente has encontrado en línea en https://linuxcnc.org/documents/.

Si deseas contribuir a LinuxCNC pero no estas seguro de dónde empezar, por favor considera seriamente en contribuir a la documentación. Cada quien encuentra siempre algo que mejorar, y si solo dejas en el texto un ”ARREGLAME: con un comentario” como referencia para ti y otros de revisar esa sección después. Además, hay traducciones a otros lenguajes distintos al inglés en https://hosted.weblate.org/-projects/linuxcnc que muy probablemente se beneficiarán de tu escrutinio.

7.2. Descargando el árbol fuente

El repositorio git del proyecto LinuxCNC está en https://github.com/LinuxCNC/linuxcnc. Github es un popular servicio de alojamiento git y un sitio web para compartir código.

Para obtener el árbol fuente tienes dos opciones:

Descargar tarball
En la página del proyecto LinuxCNC en Github encontrarás una referencia a ”releases” o ”tags”, haz clic en esa liga y descarga el archivo .tar mas reciente. Notarás que el archivo esta comprimido como .tar.xz o tar.gz. Este archivo comúnmente conocido como un ”tarball” es un archivo muy similar a un .zip. Tu escritorio de Linux sabrá como tratar ese archivo cuando hagas doble clic sobre él.

Prepara una copia local del repositorio de LinuxCNC
Primero instalarías en tu máquina la herramienta ”git” si no esta disponible (`sudo apt Install git`). Luego prepara una instancia local del árbol fuente como a continuación: .

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

. El primer argumento al comando git se explica por si mismo: Esto es lo que se llama un ”clon” del repositorio de LinuxCNC. La ventaja es que este clon local mantiene la comunicación sobre los cambios que hayas decidido hacer en el árbol fuente.

GitHub es una infraestructura por sí solo y se explica a profundidad en otros lados. Sólo para motivarte, si no lo sabes, ofrece crear un clon para ti y hacer esa instancia disponible al público. GitHub se refiere a tal instancia adicional de otro repositorio como un ”fork” (bifurcación). Puedes crear fácilmente (sin ningún costo) una bifurcación del repositorio git de LinuxCNC en GitHub, y usarla para dar seguimiento y publicar tus cambios. Después de crear tu propia bifurcación de LinuxCNC en GitHub, clónala en tu computadora de desarrollo y procede a hackear como de costumbre.

Nosotros, el proyecto LinuxCNC, esperamos que comparta sus cambios, para que la comunidad pueda beneficiarse de su trabajo. GitHub hace que compartir sea muy fácil: Después de pulir sus cambios y añadirlos a su bifurcacion github, envíenos una solicitud de extracción.

7.2.1. Inicio rápido

Si usted es un impaciente, intente esto:

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

¡Eso probablemente fallará!. No es culpa suya; solo significa que debe leer todo este documento para averiguar cómo solucionar tus problemas. Especialmente la sección sobre Satisfacer Dependencias de Construcción.

Si estás manejando un sistema con capacidad de tiempo real (como una instalación desde la imagen de LinuxCNC Live/Install, ve la sección Tiempo Real más abajo); se necesita un paso de compilación adicional:

```
$ sudo make setuid
```

Después de haber compilado con éxito LinuxCNC, es hora de ejecutar las pruebas:

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

¡Esto también podría fallar! Lea todo este documento, pero especialmente la sección en Configuración del entorno de prueba.

7.3. Plataformas compatibles

The LinuxCNC project targets modern Debian-based distributions, including Debian, Ubuntu, and Mint. We continuously test on the platforms listed at http://buildbot.linuxcnc.org.

LinuxCNC builds on most other Linux distributions, though dependency management will be more manual and less automatic. Patches to improve portability to new platforms are always welcome.

7.3.1. En tiempo real

LinuxCNC es un controlador de máquina herramienta, y requiere una plataforma en tiempo real para hacer este trabajo. Esta versión de LinuxCNC soporta las plataformas siguientes. Las primeras tres en la lista son sistemas operativos de tiempo real:

RTAI
: De https://www.rtai.org. Del archivo de Debian en https://linuxcnc.org hay disponible un kernel de Linux con el parche RTAI. Vea Obtener LinuxCNC para las instrucciones de instalación.

Xenomai
: De https://xenomai.org. Tendrá que compilar u obtener un kernel Xenomai por usted mismo.

Preempt-RT
: From https://rt.wiki.kernel.org. A Linux kernel with the Preempt-RT patch is occasionally available from the Debian archive at https://www.debian.org, and from the wayback machine at https://snapshot.debian.org.

Tiempo diferido
: LinuxCNC can also be built and run on non-realtime platforms, such as a regular install of Debian or Ubuntu without any special realtime kernel.
In this mode LinuxCNC is not useful for controlling machine tools, but it is useful for simulating the execution of G-code and for testing the non-realtime parts of the system (such as the user interfaces, and some kinds of components and device drivers).
To make use of the realtime capabilities of LinuxCNC, certain parts of LinuxCNC need to run with root privileges. To enable root for these parts, run this extra command after the `make` that builds LinuxCNC:

```
$ sudo make setuid
```

7.4. Build modes

There are two ways to build LinuxCNC: The developer-friendly ”run in place” mode and the user-friendly Debian packaging mode.

7.4.1. Building for Run In Place

In a Run-In-Place build, the LinuxCNC programs are compiled from source and then run directly from within the build directory. Nothing is installed outside the build directory. This is quick and easy, and suitable for rapid iteration of changes. The LinuxCNC test suite runs only in a Run-In-Place build. Most LinuxCNC developers primarily build using this mode.

Para una compilacion Run-In-Place, sigua los pasos en la sección Inicio rápido en la parte superior de este documento, posiblemente con diferentes argumentos para `src/configure` y` make`.

7.4.1.1. Argumentos `src/configure`

El script `src/configure` configura cómo será compilado el código fuente. Admite muchos argumentos opcionales; para enlistarlos todos ejecuta:

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

Los argumentos más utilizados son:

`--with-realtime=uspace`
: Build for any realtime platform, or for non-realtime. The resulting LinuxCNC executables will run on both a Linux kernel with Preempt-RT patches (providing realtime machine control) and on a vanilla (un-patched) Linux kernel (providing G-code simulation but no realtime machine control).

  If development files are installed for Xenomai (typically from package libxenomai-dev) or RTAI (typically from a package with a name starting ”rtai-modules”), support for these real-time kernels will also be enabled.

`--with-realtime=/usr/realtime-$VERSION`
: Build for the RTAI realtime platform using the older ”kernel realtime” model. This requires that you have an RTAI kernel and the RTAI modules installed in `/usr/realtime-$VERSION`. The resulting LinuxCNC executables will only run on the specified RTAI kernel. As of LinuxCNC 2.7, this produces the best realtime performance.

`--enable-build-documentation`
: Build the documentation, in addition to the executables. This option adds significantly to the time required for compilation, as building the docs is quite time consuming. If you are not actively working on the documentation you may want to omit this argument.

`--disable-build-documentation-translation`
: Disable building the translated documentation for all available languages. The building of the translated documentation takes a huge amount of time, so it is recommend to skip that if not really needed.

7.4.1.2. `make` arguments

The `make` command takes two useful optional arguments.

Parallel compilation
: `make` takes an optional argument `-j` *N* (where *N* is a number). This enables parallel compilation with *N* simultaneous processes, which can significantly speed up your build.

  A useful value for *N* is the number of CPUs in your build system.

You can discover the number of CPUs by running `nproc`.

Building just a specific target
: If you want to build just a specific part of LinuxCNC, you can name the thing you want to build on the `make` command line. For example, if you are working on a component named `froboz`, you can build its executable by running:

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

7.4.2. Construyendo paquetes Debian

When building Debian packages, the LinuxCNC programs are compiled from source and then stored in a Debian package, complete with dependency information. This process by default also includes the building of the documentation, which takes its time because of all the I/O for many languages, but that can be skipped. LinuxCNC is then installed as part of those packages on the same machines or on whatever machine of the same architecture that the .deb files are copied to. LinuxCNC cannot be run until the Debian packages are installed on a target machine and then the executables are available in /usr/bin and /usr/lib just like other regular software of the system.

This build mode is primarily useful when packaging the software for delivery to end users, and when building the software for a machine that does not have the build environment installed, or that does not have internet access.

Si usted es un impaciente, intente esto:

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

Building Debian packages is performed with the `dpkg-buildpackage` tool that is provided by the `dpkg-dev` package. Its execution comes with a series of prerequisites that are detailed below: * general build infrastructure shall be installed, i.e. compilers, etc. * build-time dependencies are to be installed, i.e. header files for external code libraries used, as described in the section Satisfying Build Dependencies. * file in debian folder need to be complete that describe the package

Build tools have been gathered as a virtual package named `build-essential`. To install it, run:

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

Once those prerequisites are met, building the Debian packages consists of two steps.

The first step is generating the Debian package scripts and meta-data from the git repo by running this:

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

---

nota
The `debian/configure` script is different from the `src/configure` script!
The `debian/configure` accepts arguments depending on the platform you are building on/for, see the debian/configure arguments section. It defaults to LinuxCNC running in user space (”uspace”), expecting the preempt_rt kernel to minimize latencies.

Once the Debian package scripts and meta-data are configured, build the package by running `dpkg-buildpac`

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

---

nota
`dpkg-buildpackage` needs to run from the root of the source tree, which you mave have named `linuxcnc-source-dir`, **not** from within `linuxcnc-source-dir/debian`.
`dpkg-buildpackage` takes an optional argument ``-j`N` (where *N* is a number). This enables to run multiple jobs simultaneously.

7.4.2.1. LinuxCNC's `debian/configure` arguments

The LinuxCNC source tree has a debian directory with all the info about how the Debian package shall be built, but some key files within are only distributed as templates. The `debian/configure` script readies those build instructions for the regular Debian packaging utilities and must thus be run prior to `dpkg-checkbuilddeps` or `dpkg-buildpackage`.

The `debian/configure` script takes a single argument which specifies the underlying realtime or non-realtime platform to build for. The regular values for this argument are:

`no-docs`
: Saltar construcción de documentación.

`uspace`
: Configure the Debian package for Preempt-RT realtime or for non-realtime (these two are compatible).

`noauto` , `rtai` , `xenomai`
: Normally, the lists of RTOSes for uspace realtime to support is detected automatically. However, if you wish, you may specify one or more of these after `uspace` to enable support for these RTOSes. Or, to disable autodetection, specify `noauto`.

  If you want just the traditional RTAI "kernel module" realtime, use `-r` or `$KERNEL_VERSION` instead.

`rtai=<package name>`
: If the development package for RTAI, lxrt, does not start with "rtai-modules", or if the first such package listed by apt-cache search is not the desired one, then explicitly specify the package name.

`-r`
: Configure the Debian package for the currently running RTAI kernel. You must be running an RTAI kernel on your build machine for this to work!

`$KERNEL_VERSION`
: Configure the Debian package for the specified RTAI kernel version (for example "3.4.9-rtai-686-pae"). The matching kernel headers Debian package must be installed on your build machine, e.g. "linux-headers-3.4.9-rtai-686-pae". Note that you can *build* LinuxCNC in this configuration, but if you are not running the matching RTAI kernel you will not be able to *run* LinuxCNC, including the test suite.

7.4.2.2. Satisfying Build Dependencies

En las plataformas basadas en Debian, proporcionamos metadatos de empaquetado que saben qué paquetes de software externos deben instalarse para construir LinuxCNC. Son referidas como *buid dependencies* de LinuxCNC, ej, aquellos paquetes que deben estar disponibles como

- the build succeeds and
- the build can be built reproducibly.

You can use this meta-data to easily list the required packages missing from your build system. First, go to the source tree of LinuxCNC and initiate its default self-configuration, if not already performed:

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

This will prepare the file debian/control that contains lists of Debian packages to create with the runtime dependencies for those packages and for our cause also the build-dependencies for those to-be-created packages.

The most straightforward way to get all build-dependencies installed is to just execute (from the same directory):

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

which will install all the dependencies required, not yet installed, but available. The . is part of the command line, i.e. an instruction to retrieve the dependencies for the source tree at hand, not for dependencies of another package. This completes the installation of build-dependencies.

The remainder of this section describes a semi-manual approach. The list of dependencies in debian/control is long and it is tedious to compare the current state of packages already installed with it. Debian systems provide a program called `dpkg-checkbuilddeps` that parses the package meta-data and compares the packages listed as build dependencies against the list of installed packages, and tells you what's missing.

First, install the `dpkg-checkbuilddeps` program by running:

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

This generates the file `debian/control` in a user-readable yaml-format which lists the build-dependencies close to the top. You can use this meta-data to easily list the required packages missing from your build system. You may decide to manually inspecting those files if you have a good understanding what is already installed.

Alternatively, Debian systems provide a program called `dpkg-checkbuilddeps` that parses the package meta-data and compares the packages listed as build dependencies against the list of installed packages, and tells you what's missing. Also, `dpkg-buildpackage` would inform you about what is missing, and it should be fine. However, it reports missing build-deps only after patches in the directory `debian/patches` have been automatically applied (if any). If you are new to Linux and git version management, a clean start may be preferable to avoid complications.

The `dpkg-checkbuilddeps` (also from the dpkg-dev package that is installed as part of the build-essential dependencies) program can be asked to do its job (note that it needs to run from the `linuxcnc-source-dir` directory, **not** from `linuxcnc-source-dir/debian`):

```
$ dpkg-checkbuilddeps
```

Esto emitirá una lista de paquetes necesarios para construir LinuxCNC en su sistema, pero que aún no están instalados. Ahora puede instalar dependencias de construcción faltantes

manually
: Install them all with `sudo apt-get install`, followed by the package names. You can rerun `dpkg-checkbuilddeps` any time you want, to list any missing packages, which has no effect on the source tree.

automated
: Ejecute `sudo apt build-dep .`.

If in doubt about what a particular package of a build-dep may be providing, check out the package's description with ``apt-cache show`` *packagename*.

7.4.2.3. Opciones para `dpkg-buildpackage`

For a typical Debian package to build, you would run dpkg-buildpackage without any arguments. As introduced above, the command has two extra options passed to it. Like for all good Linux tools, the man page has all the details with `man dpkg-buildpackage`.

-uc
: Do not digitally sign the resulting binaries. You would want to sign your packages with a GPG key of yours only if you would wanted to distribute them to others. Having that option not set and then failing to sign the package would not affect the .deb file.

-b
: Only compiles the architecture-dependent packages (like the `linuxcnc` binaries and GUIs). This is very helpful to avoid compiling what is hardware-independent. For LinuxCNC this is the documentation, which is available online anyway.

If you happen to run into difficulties while compiling, check the LinuxCNC forum online.

Currently emerging is the support for the DEB_BUILD_OPTIONS environment variable. Set it to

nodocs
: to skip building the documentation, preferably instead use the `-B` flag to dpkg-buildpackage.

nocheck
: to skip self-tests of the LinuxCNC build process. This saves some time and reduces the demand for a few software packages that may not be available for your system, i.e. the xvfb in particular. You should not set this option to gain some extra confidence in your build to perform as expected unless you are running into mere technical difficulties with the test-specific software dependencies.

An environment variable can be set together with the execution of the command, e.g.

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

combinará todas las opciones presentadas en esta sección.

7.4.2.4. Installing self-built Debian packages

A Debian package can be recognised by its .deb extension. The tool installing it, `dpkg` is part of every Debian installation. The .deb files created by `dpkg-buildpackage` are found in the directory above the linuxcnc-source-dir, i.e. in `...` To see what files are provided in a package, run

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

The version of LinuxCNC will be part of the file name, which is meant to be matched by the asterisk. There may be too many files listed to fit on your screen. If you cannot scroll up in your terminal then add `| more` to that command to have its output passed through a so-called ”pager”. Quit with ”q”.

Para instalar los paquetes ejecute

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

7.5. Setting up the environment

This section describes the special steps needed to set up a machine to run the LinuxCNC programs, including the tests.

7.5.1. Increase the locked memory limit

LinuxCNC tries to improve its realtime latency by locking the memory it uses into RAM. It does this in order to prevent the operating system from swapping LinuxCNC out to disk, which would have bad effects on latency. Normally, locking memory into RAM is frowned upon, and the operating system places a strict limit on how much memory a user is allowed to have locked.

When using the Preempt-RT realtime platform LinuxCNC runs with enough privilege to raise its memory lock limit itself. When using the RTAI realtime platform it does not have enough privilege, and the user must raise the memory lock limit.

If LinuxCNC displays the following message on startup, the problem is your system's configured limit on locked memory:

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

To fix this problem, add a file named `/etc/security/limits.d/linuxcnc.conf` (as root) with your favorite text editor (e.g., `sudo gedit /etc/security/limits.d/linuxcnc.conf`). The file should contain the following line:

```
* - memlock 20480
```

Log out and log back in to make the changes take effect. Verify that the memory lock limit is raised using the following command:

```
$ ulimit -l
```

7.6. Building on Gentoo

Building on Gentoo is possible, but not supported. Be sure you are running a desktop profile. This project uses the Tk Widget Set, asciidoc, and has some other dependencies. They should be installed as 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
```

You can switch back to being a normal user for most of the rest of the install. As that user, create a virtual environment for pip, then install the pip packages:

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

Then you can contrinue as normally:

```
(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
```

There is no need to run "make suid", just make sure your user is in the "dialout" group. To start linuxcnc, you must be in the Python Virtual Environment, and set up the linuxcnc environment:

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

7.7. Options for checking out the git repo

The Quick Start instructions at the top of this document clone our git repo at https://github.com/-LinuxCNC/linuxcnc.git. This is the quickest, easiest way to get started. However, there are other options to consider.

7.7.1. Fork us on GitHub

The LinuxCNC project git repo is at https://github.com/LinuxCNC/linuxcnc. GitHub is a popular git hosting service and code sharing website. You can easily (and at no costs) create a fork (a second instance holding a copy that you control) of the LinuxCNC git repository at GitHub. You can then use that fork of yours to track and publish your changes, receive comments to your changes and accept patches from the community. .

After creating your own GitHub fork of LinuxCNC, clone it to your development machine and proceed with your hacking as usual.

We of the LinuxCNC project hope that you will share your changes with us, so that the community can benefit from your work. GitHub makes this sharing very easy: after you polish your changes and push them to your GitHub fork, send us a Pull Request.

Capítulo 8

Agregar elementos al selector de configuraciones

Se pueden agregar configuraciones de ejemplo al Selector de configuración por dos métodos:

- Aplicaciones auxiliares — Aplicaciones instaladas independientemente con un paquete deb que pueden colocar subdirectorios de configuración en un directorio de sistema en específico. El nombre del directorio se especifica utilizando el script de shell linuxcnc_var:

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

- Configuración de tiempo de ejecución — el selector de configuración también puede ofrecer subdirectorios de configuración especificados en tiempo de ejecución utilizando una variable de entorno exportada (LINUXCNC_AUX_CONFIGS). Esta variable debe ser una lista de rutas de uno o más directorios de configuraciones separados por (:). Normalmente, esta variable se establecería en un shell de inicio de `linuxcnc` o en un script de inicio de usuario `~/.profile`. Ejemplo:

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

Capítulo 9

Contributing to LinuxCNC

9.1. Introducción

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.

9.2. 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

9.3. The LinuxCNC Source Forge project

Utilizamos Source Forge para las listas de correo.

9.4. The Git Revision Control System

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

9.4.1. 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

9.4.2. 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

We have a development branch called `master`, and one or more stable branches with names like `2.6` and `2.7` indicating the version number of the releases we make from it.

Bugfixes go in the oldest applicable stable branch, and that branch gets merged into the next newer stable branch, and so on up to `master`. The committer of the bugfix may do the merges themselves, or they may leave the merges for someone else.

New features generally go in the `master` branch, but some kinds of features (specifically well isolated device drivers and documentation) may (at the discretion of the stable branch release managers) go into a stable branch and get merged up just like bugfixes do.

9.4.3. git tutorials

There are many excellent, free git tutorials on the internet.

The first place to look is probably the "gittutorial" manpage. This manpage is accessible by running "man gittutorial" in a terminal (if you have the git manpages installed). The gittutorial and its follow-on documentation are also available online here:

- 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
- Everyday git with 20 commands or so: https://www.kernel.org/pub/software/scm/git/docs/giteveryday.html
- Git User's Manual: https://www.kernel.org/pub/software/scm/git/docs/user-manual.html

Para una documentación más completa de git, vea el libro "Pro Git": https://git-scm.com/book

Another online tutorial that has been recommended is "Git for the Lazy": https://wiki.spheredev.org/-index.php/Git_for_the_lazy

9.5. Overview of the process

The high-level overview of how to contribute changes to the source goes like this:

- Communicate with the project developers and let us know what you're hacking on. Explain what you are doing, and why.
- Clonar el repositorio git.
- Make your changes in a local branch.

- Adding documentation and writing tests is an important part of adding a new feature. Otherwise, others won't know how to use your feature, and if other changes break your feature it can go unnoticed without a test.
- Share your changes with the other project developers in one of these ways:
  - Haga push de su rama a github y cree una solicitud de extracción pull de github a https://github.com/-linuxcnc/linuxcnc (esto requiere una cuenta github), o
  - 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
  - Envíe sus confirmaciones por correo electrónico a la lista de correo de LinuxCNC-developers (<emc-developers@lists.sourceforge.net>) (use `git format-patch` para crear parches).
- Defienda su parche:
  - Explain what problem it addresses and why it should be included in LinuxCNC.
  - Be receptive to questions and feedback from the developer community.
  - It is not uncommon for a patch to go through several revisions before it is accepted.

9.6. git configuration

In order to be considered for inclusion in the LinuxCNC source, commits must have correct Author fields identifying the author of the commit. A good way to ensure this is to set your global git config:

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

Use your real name (not a handle), and use an unobfuscated e-mail address.

9.7. Effective use of git

9.7.1. Commit contents

Keep your commits small and to the point. Each commit should accomplish one logical change to the repo.

9.7.2. Write good commit messages

Keep commit messages around 72 columns wide (so that in a default-size terminal window, they don't wrap when shown by `git log`).

Use the first line as a summary of the intent of the change (almost like the subject line of an e-mail). Follow it with a blank line, then a longer message explaining the change. Example:

9.7.3. Commit to the proper branch

Bugfixes should go on the oldest applicable branch. New features should go in the master branch. If you're not sure where a change belongs, ask on irc or on the mailing list.

9.7.4. Use multiple commits to organize changes

When appropriate, organize your changes into a branch (a series of commits) where each commit is a logical step towards your ultimate goal. For example, first factor out some complex code into a new function. Then, in a second commit, fix an underlying bug. Then, in the third commit, add a new feature which is made easier by the refactoring and which would not have worked without fixing that bug.

This is helpful to reviewers, because it is easier to see that the ”factor out code into new function” step was right when there aren’t other edits mixed in; it’s easier to see that the bug is fixed when the change that fixes it is separate from the new feature; and so on.

9.7.5. Follow the style of the surrounding code

Make an effort to follow the prevailing indentation style of surrounding code. In particular, changes to whitespace make it harder for other developers to track changes over time. When reformatting code must be done, do it as a commit separate from any semantic changes.

9.7.6. Deshacerse de RTAPI_SUCCESS, usar 0 en su lugar

La prueba ”retval < 0” debería ser familiar; es el mismo tipo de prueba que se utiliza en el espacio de usuario (devuelve -1 para error) y en el espacio de kernel (devuelve -ERRNO para error).

9.7.7. Simplify complicated history before sharing with fellow developers

With git, it’s possible to record every edit and false start as a separate commit. This is very convenient as a way to create checkpoints during development, but often you don’t want to share these false starts with others.

Git provides two main ways to clean history, both of which can be done freely before you share the change:

`git commit --amend` lets you make additional changes to the last thing you committed, optionally modifying the commit message as well. Use this if you realized right away that you left something out of the commit, or if you typo’d the commit message.

`git rebase --interactive` upstream-branch lets you go back through each commit made since you forked your feature branch from the upstream branch, possibly editing commits, dropping commits, or squashing (combining) commits with others. Rebase can also be used to split individual commits into multiple new commits.

9.7.8. Make sure every commit builds

If your change consists of several patches, `git rebase -i` may be used to reorder these patches into a sequence of commits which more clearly lays out the steps of your work. A potential consequence of reordering patches is that one might get dependencies wrong - for instance, introducing a use of a variable, and the declaration of that variable only follows in a later patch.

While the branch HEAD will build, not every commit might build in such a case. That breaks `git bisect` - something somebody else might use later on to find the commit which introduced a bug. So beyond making sure your branch builds, it is important to assure every single commit builds as well.

There’s an automatic way to check a branch for each commit being buildable - see https://dustin.sallings.org/2010/03/28/git-test-sequence.html and the code at https://github.com/dustin/bindir/blob/master/git-test-sequence. Use as follows (in this case testing every commit from origin/master to HEAD, including running regression tests):

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

This will either report *All is well* or *Broke on <commit>*

9.7.9. Renaming files

Please use the ability to rename files very cautiously. Like running indent on single files, renames still make it more difficult to follow changes over time. At a minimum, you should seek consensus on irc or the mailing list that the rename is an improvement.

9.7.10. Prefer "rebase"

Use `git pull --rebase` instead of bare `git pull` in order to keep a nice linear history. When you rebase, you always retain your work as revisions that are ahead of origin/master, so you can do things like `git format-patch` them to share with others without pushing to the central repository.

9.8. Translations

The LinuxCNC project uses `gettext` to translate the software into many languages. We welcome contributions and help in this area! Improving and extending the translations is easy: you don't need to know any programming, and you don't need to install any special translation programs or other software.

La forma más fácil de ayudar con traducciones es usando Weblate, un servicio web de código abierto. Nuestro proyecto de traducción esta aquí:

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

Documentation on how to use Weblate is here: https://docs.weblate.org/en/latest/user/basic.html

9.9. Other ways to contribute

There are many ways to contribute to LinuxCNC, that are not addressed by this document. These ways include:

- Answering questions on the forum, mailing lists, and in IRC
- Reporting bugs on the bug tracker, forum, mailing lists, or in IRC
- Helping test experimental features

Capítulo 10

Glosario

Una lista de términos y su significado. Algunos términos tienen un significado general y varios significados adicionales para usuarios, instaladores y desarrolladores.

Tornillo Acme
: Un tipo de tornillo de avance que tiene el roscado en forma Acme. Las roscas Acme tienen un poco menos de fricción y desgaste que los roscados triangulares, pero los husillos de bolas tienen aún menos. La mayoría de las herramientas de máquina manuales tienen tornillos de avance Acme.

Eje
: Una de las partes movibles controladas por computadora de la máquina. Para una típica fresadora vertical, la mesa es el eje X, el carro transversal es el eje Y, y la rodilla o caña es el eje Z. Los ejes angulares como en mesas giratorias son referidos como A, B y C. Los ejes lineales adicionales con relación a la herramienta se denominan U, V y W respectivamente.

AXIS (GUI)
: Una de las interfaces gráficas de usuario disponibles para los usuarios de LinuxCNC. Cuenta con uso de menús y botones de ratón a la vez que automatiza y oculta algunos de los controles más tradicionales de LinuxCNC. Es la única interfaz de código abierto que muestra completamente la ruta de la herramienta tan pronto como se abre un archivo.

GMOCCAPY (GUI)
: Una interfaz gráfica de usuario disponible para los usuarios de LinuxCNC. Cuenta con el aspecto y sensación de un control industrial y puede usarse con pantalla táctil, teclado y mouse. Admite pestañas incrustadas y mensajes al usuario controlados mediante HAL, ofrece bastantes estados HAL para controlarse con el hardware. GMOCCAPY es muy personalizable.

Holgura mecánica
: La cantidad de ”juego” o movimiento perdido que ocurre en un cambio de dirección en un tornillo de avance, o en otros sistemas de conducción de movimiento mecánico. Puede ser el resultado de tuercas aflojadas en tornillos de avance, deslizamiento de bandas, fatiga de cable, desgaste en acoplamientos giratorios, y otras piezas donde el sistema mecánico no está ”apretado”. La holgura mecánica provocará movimientos imprecisos, o en el caso de movimiento causado por fuerzas externas (piensa en una herramienta de corte jalando la pieza de trabajo) resultará en rotura de herramientas de corte debido al incremento repentino de viruta en el cortador cuando jala la pieza de trabajo a lo largo de la distancia de holgura.

Compensación de holgura mecánica
: Cualquier técnica que intente reducir el efecto de holgura mecánica sin eliminarla de hecho del sistema mecánico. Esto se hace típicamente con software en el controlador. Esto puede corregir el lugar final de descanso de la parte en movimiento, pero no servirá para resolver problemas relacionados con cambios de dirección al estar en movimiento (piensa en interpolación circular),

ni para resolver movimiento provocado por fuerzas externas (piensa en una herramienta de corte jalando la pieza de trabajo).

Tornillo de bolas
Un tipo de tornillo de avance que usa pequeñas bolas de acero endurecido entre la tuerca y el tornillo para reducir la fricción. Los tornillos de bolas tienen fricción y holgura mecánica muy bajos, pero son normalmente muy costosos.

Tuerca de bolas
Una tuerca especial diseñada para usarse con tornillo de bolas. Contiene un pasaje interno para recircular la bolas de un lado a otro del tornillo.

CNC
Control numérico por computadora. El término general se usa para referirse al control computarizado de maquinaria. En lugar de que un operador humano dé vueltas a una manivela para mover una herramienta de corte, CNC usa una computadora y motores para mover la herramienta, con base en un programa de parte.

Halcompile
Una herramienta para construir, compilar e instalar componentes HAL de LinuxCNC.

Configuración (sustantivo)
Un directorio que contiene un conjunto de archivos de configuración. Las configuraciones personalizadas se guardan normalmente en el directorio del usuario home/linuxcnc/configs. Los archivos incluyen a los tradicionales de LinuxCNC INI y HAL. Una configuración también puede contener varios archivos generales que describan herramientas, parámetros y conexiones NML.

Configuración (verbo)
La acción de configurar LinuxCNC, de tal manera que coincida con el hardware en una máquina herramienta.

Máquina de medición por coordenadas
Una máquina de medición por coordenadas se usa para realizar varias mediciones precisas en partes. Estas máquinas pueden usarse para crear datos CAD de partes de las que no se encuentran sus diagramas, o para digitalizar para moldeo un prototipo hecho a mano, o para verificar la precisión de partes maquinadas o moldeadas.

Unidades de visualización
Las unidades lineales y angulares usadas para mostrarse en pantalla.

DRO
Un lector digital (Digital Read Out) es un sistema de dispositivos de medición de posición anexos a las guías de una máquina herramienta, el cual esta conectado a una pantalla numérica que muestra la posición actual de la herramienta con respecto a un punto de referencia. Los DROs son muy populares en máquinas herramientas operadas a mano porque miden la verdadera posición de la herramienta sin holgura mecánica, incluso si la máquina tiene tornillos Acme aflojados. Algunos DROs usan codificadores lineales de cuadratura para recolectar información de posición desde la máquina, y algunos usan métodos similares a un solucionador, los cuáles aún siguen en funcionamiento.

EDM
Es un método para quitar material de metales duros o difíciles de maquinar, o donde las herramientas rotatorias no puedan crear la forma deseada con una buena relación costo-beneficio. Un excelente ejemplo son las matrices rectangulares, donde son deseables las esquinas interiores afiladas, que no se pueden lograr con fresado por el límite del diámetro de la herramienta. Una máquina EDM de *hilo* puede crear esquinas interiores con un radio tan solo un poco más grande que el radio del alambre. Una EDM *penetradora* puede hacer esquinas interiores con un radio poco mas grande que el radio de la esquina del electrodo inmerso.

EMC
El Controlador de Máquina Mejorado. Inicialmente un proyecto de NIST. Renombrado a LinuxCNC en 2012.

EMCIO
El módulo dentro de LinuxCNC que maneja las E/S de propósito general, no se relaciona con el movimiento real de los ejes.

EMCMOT
El módulo dentro de LinuxCNC que maneja el movimiento real de la herramienta de corte. Se ejecuta como un programa en tiempo real y controla directamente los motores.

Codificador
Un dispositivo para medir posición. Normalmente un dispositivo opto-mecánico, del cual sale una señal en cuadratura. La señal puede contarse con hardware especial, o directamente con el puerto paralelo con LinuxCNC.

Alimentación
Movimiento controlado y relativamente lento de la herramienta en uso al hacer un corte.

Velocidad de alimentación
La velocidad a la que ocurre el movimiento de corte. En modo automático o MDI, la velocidad de alimentación es comandada con una palabra F. F10 significa diez unidades de máquina por minuto.

Retroalimentación
Un método (p. ej. señales de un codificador en cuadratura) por el cual LinuxCNC recibe información sobre la posición de los motores.

Porcentaje de alimentación
Un cambio manual controlado por el operador en la velocidad a la que se mueve la herramienta durante el corte. A menudo usado para permitirle al operador ajustar herramientas que están algo aburridas, o cualquier otra que requiera que se ”ajuste” la velocidad de alimentación.

Número de punto flotante
Un número que tiene un punto decimal. (12.300) En HAL se le conoce como float.

Códigos G
El término general usado para referirse a la parte más común de lenguaje de programación. Existen diversos dialectos de código G, LinuxCNC usa RS274/NGC.

GUI
Interfaz gráfica de usuario.

En general
Un tipo de interfaz que permite comunicaciones entre una computadora y un humano (en la mayoría de los casos) mediante la manipulación de iconos y otros elementos (widgets) en una pantalla de computadora.

LinuxCNC
Una aplicación que presenta una pantalla gráfica al operador de una máquina permitiendo la manipulación de la máquina y del correspondiente programa controlador.

HAL
Capa de Abstracción de Hardware. Al más alto nivel, es simplemente una manera de permitir que una serie de bloques de construcción sean cargados e interconectados para ensamblar un sistema complejo. Muchos de los bloques de construcción son controladores de dispositivos de hardware. Sin embargo, HAL puede hacer más que solo configurar controladores de hardware.

Casa
Una ubicación específica en el espacio de trabajo de la máquina que se usa para asegurar que tanto la computadora como la máquina real estén de acuerdo en la posición de la herramienta.

Archivo INI
Un archivo de texto que contiene la mayoría de la información que configura LinuxCNC para una máquina en particular.

Instancia
Uno puede tener una instancia de una clase o de un objeto particular. La instancia es el objeto real creado en tiempo de ejecución. En la jerga de programación, el objeto ”Lassie” es una instancia de la clase ”Perro”.

Coordenadas de articulación
Especifican los ángulos entre la articulaciones individuales de la máquina. Ver también Cinemática

Trote
Movimiento manual de un eje de la máquina. Trotar mueve el eje ya sea una cantidad fija por cada pulsación de tecla, o a una velocidad constante mientras se mantenga presionada la tecla. En modo manual, la velocidad de trote puede especificarse desde la interfaz gráfica.

espacio de kernel
Código ejecutándose dentro del kernel, lo opuesto a código ejecutándose en el espacio de usuario. Algunos sistemas en tiempo real (como RTAI) corren código en tiempo real en el kernel y código en tiempo no-real en el espacio de usuario, mientras que otros sistemas (como Preempt-RT) corren códigos tanto en tiempo real como en tiempo no-real en el espacio de usuario.

Cinemática
La relación de posición entre las coordenadas mundiales y las coordenadas de la articulación de una máquina. Hay dos tipos de cinemáticas. Las cinemáticas directas se usan para calcular las coordenadas mundiales desde las coordenadas de la articulación. La cinemáticas inversas se usan exactamente para el propósito opuesto. Considere que las cinemáticas no toman en cuenta las fuerzas, momentos, etc. en la máquina. Sólo es para posicionamiento.

Tornillo de avance
Un tornillo que es girado por un motor para mover una mesa u otra parte de una máquina. Los tornillos de avance son comúnmente tornillos de bolas o tornillos Acme; aunque también se pueden usar tornillos convencionales de cuerda triangular cuando la precisión y su tiempo de vida no son tan importantes como el bajo costo.

Unidades de máquina
Las unidades lineares y angulares usadas para la configuración de la máquina. Estas unidades se especifican y usan en el archivo INI. Los pines HAL y parámetros también son generalmente en unidades de máquina.

MDI
Entrada de Datos Manual. Es un modo de operación donde el controlador ejecuta líneas individuales de código G al ser tecleadas por el operador.

NIST
Instituto Nacional de Estándares y Tecnología. Una agencia del Departamento de Comercio en los Estados Unidos.

NML
El Lenguaje de Mensaje Neutral proporciona un mecanismo para manejar distintos tipos de mensajes en el mismo búfer, así como una simplificación de la interfaz para codificar y decodificar búfers en un formato neutral y el mecanismo de configuración.

Offsets
Una cantidad arbitraria agregada al valor de algo para igualarlo a un valor deseado. Por ejemplo, los programas en código G a menudo se escriben alrededor de un punto conveniente, como X0, Y0. Se pueden usar offsets de fijación para alterar el punto de ejecución actual de ese programa en código G para ajustar la verdadera ubicación de la prensa y mordaza. Los offsets de herramienta pueden usarse para cambiar la longitud ”incorrecta” de una herramienta para igualarla con su longitud real.

Programa de parte
Una descripción de una parte, en un lenguaje que el controlador pueda entender. Para LinuxCNC ese lenguaje es RS274/NGC, comúnmente conocido como código G.

Unidades de programa
Las unidades lineares y angulares usadas en un programa de parte. Las unidades de programa lineares no necesariamente tienen que ser las mismas que las unidades lineares de máquina. Ver G20 y G21 para más información. Las unidades angulares de programa siempre se expresan en grados.

Python
Lenguaje de programación de propósito general de muy alto nivel. Usado en LinuxCNC para la GUI Axis, para la herramienta de configuración StepConf y para varios scripts de programación en código G.

Movimiento rápido
Movimiento de la herramienta rápido y posiblemente menos preciso. Usado comúnmente para movimientos entre cortes. Si la herramienta topa con la pieza de trabajo o la fijación durante un movimiento rápido, probablemente ¡será algo malo!

Velocidad rápida
La velocidad a la cual ocurre un movimiento rápido. En modo automático o MDI, la velocidad rápida es normalmente la velocidad máxima de la máquina. A menudo es deseable limitar la velocidad rápida cuando se prueba por primera vez un programa en código G.

Tiempo real
Software destinado a alcanzar tiempos límite sumamente estrictos. En Linux, con tal de cumplir con esa exigencia, es necesario instalar un kernel en tiempo real como RTAI o Preempt-RT, y compilar el software de LinuxCNC para ejecutarse en ese ambiente especial de tiempo real. Un software en tiempo real puede ejecutarse en el kernel o en el espacio de usuario, dependiendo de las facilidades que ofrezca el sistema.

RTAI
Interfaz de Aplicación en Tiempo Real, ver https://www.rtai.org/, las extensiones de tiempo real para Linux que puede usar LinuxCNC para alcanzar desempeño de tiempo real.

RTLINUX
Ver https://es.wikipedia.org/wiki/RTLinux, una extensión antigua de tiempo real para Linux que LinuxCNC solía usar para alcanzar desempeño de tiempo real. Obsoleta, reemplazada por RTAI.

RTAPI
Una interfaz portable para sistemas operativos en tiempo real incluyendo pthreads de RTAI y POSIX con extensiones de tiempo real.

RS-274/NGC
El nombre formal para el lenguaje usado en programas de parte de LinuxCNC.

Servo motor
En general, cualquier motor que se use con retroalimentación de detección de errores para corregir la posición de un actuador. También, un motor especialmente diseñado para brindar desempeño mejorado en tales aplicaciones.

Bucle de servo
Un bucle de control usado para control de posición o velocidad de un motor equipado con un dispositivo de retroalimentación.

Número entero con signo
Un número entero que puede tener signo positivo o negativo. En HAL es normalmente un s32, pero también podría ser s64.

Husillo
La parte de una máquina herramienta que gira para hacer el corte. En una fresadora, el husillo sostiene la herramienta de corte. En un torno, el husillo sostiene la pieza de trabajo.

Porcentaje de velocidad del husillo
Un cambio manual controlado por el operador en la velocidad a la que gira la herramienta durante el corte. A menudo usado para permitir al operador ajustar la vibración causada por los dientes del cortador. El porcentaje de velocidad manual del husillo al que se ha configurado LinuxCNC para controlar la velocidad del husillo.

StepConf
Un asistente de configuración de LinuxCNC. Es capaz de manejar diversas máquinas basadas en comandos de movimiento por paso y dirección. Escribe una configuración completa después de que el usuario contesta algunas preguntas acerca de la computadora y la máquina donde correrá LinuxCNC.

Motor a pasos
Un tipo de motor que gira a pasos fijos. Contando los pasos, es posible determinar cuánto ha girado el motor. Si la carga excede la capacidad de torque del motor, se saltarán uno o más pasos, provocando errores de posición.

TASK
El módulo dentro de LinuxCNC que coordina la ejecución en general e interpreta el programa de parte.

Tcl/Tk
Un lenguaje de secuencias de comandos y conjunto de herramientas de widgets gráficos con el que se han escrito diversos GUIs y asistentes de selección de LinuxCNC.

Atravesar
Un movimiento en línea recta desde el punto inicial hasta el punto final.

Unidades
Ver ”Unidades de máquina”, ”Unidades de visualización” o ”Unidades de programa”.

Entero sin signo
Un número entero que no tiene signo. En HAL es normalmente un u32 pero también podría ser un u64.

Coordenadas mundiales
Es el marco de referencia absoluto. Da coordenadas en términos de un marco de referencia fijo ligado a un punto (generalmente la base) de una máquina herramienta.

Capítulo 11

Legal Section

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

11.1. Copyright Terms

11.2. GNU Free Documentation License

Licencia de Documentación Libre GNU Versión 1.1, Marzo 2000

0. PREÁMBULO

The purpose of this License is to make a manual, textbook, or other written document ”free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

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. APLICABILIDAD Y DEFINICIONES

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.

A ”Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

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.

A ”Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not ”Transparent” is called ”Opaque”.

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.COPIA TEXTUAL

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. COPIA EN CANTIDAD

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.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICACIONES

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. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission. B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five). C. State on the Title page the name of the publisher of the Modified Version, as the publisher. D. Preserve all the copyright notices of the Document. E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices. F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below. G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document’s license notice. H. Include an unaltered copy of this License. I. Preserve the section entitled ”History”, and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled ”History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence. J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the ”History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission. K. In any section entitled ”Acknowledgements” or ”Dedications”, preserve the section’s title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein. L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles. M. Delete any section entitled ”Endorsements”. Such a section may not be included in the Modified Version. N. Do not retitle any existing section as ”Endorsements” or to conflict in title with any Invariant Section.

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. COMBINANDO DOCUMENTOS

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. COLECCIONES DE DOCUMENTOS

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. AGREGACIÓN CON TRABAJOS INDEPENDIENTES

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. TRADUCCIÓN

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. TERMINACIÓN

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. REVISIONES FUTURAS DE ESTA LICENCIA

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: Cómo usar esta Licencia para tus documentos

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.