Manual del Desarrollador
V2.10.0-pre0-5905-g658d666a6f

Índice general

Capítulo 1

Introducción

Este manual es un trabajo en progreso. Si puede ayudar con redacción, edición o preparación gráfica, comuníquese con cualquier miembro del equipo de redacción o únase y envíe un correo electrónico a emc-users@lists.sourceforge.net.

Capítulo 2

Referencia general de HAL

2.1. Nombres de entidades HAL

Todas la entidades HAL son accesibles y manipulables por sus nombres, por lo que es muy importante documentar los nombres de los pines, señales, parámetros, etc. Los nombres en HAL tienen una longitud máxima de 41 caracteres (como se define en HAL_NAME_LEN en hal.h). Muchos nombres serán presentados en la forma general, con el texto fomateado <*como-este*> representando campos de varios valores.

Cuando se describan por primera vez pines, señales o parámetros, su nombre será precedido por su tipo en paréntesis (*float*) y seguido por una descripción breve. Las definiciones típicas de pines se ven como estos ejemplos:

(bit) parport.<número-de-puerto>.pin-<número-de-pin>-in
: El pin HAL asociado con el pin de entrada física <*número-de-pin*> de un conector db25.

(float) pid.<número-de-bucle>.output
: La salida de bucle PID

Ocasionalmente, se puede usar una versión abreviada del nombre, por ejemplo, el segundo pin de arriba pudo haber sido llamado simplemente con *.output* cuando se puede hacer sin causar confusión.

2.2. Convenciones generales de nombres en HAL

Unas convenciones de nombres consistentes harán que HAL sea mucho más fácil de usar. Por ejemplo, si cada controlador de codificador proporciona el mismo conjunto de pines y los nombra de la misma manera, sería fácil cambiar de un tipo de controlador de codificador a otro. Desafortunadamente, al igual que muchos proyectos de código abierto, HAL es una combinación de cosas que fueron diseñadas, y cosas que simplemente evolucionaron. Como resultado, hay muchas inconsistencias. Esta sección intenta abordar ese problema definiendo algunas convenciones, pero probablemente pasará un tiempo antes de que todos los módulos se conviertan para seguirlas.

Halcmd y otras utilidades HAL de bajo nivel tratan los nombres HAL como entidades simples, sin estructura interna. Sin embargo, la mayoría de los módulos sí tienen alguna estructura implícita. Por ejemplo, una tarjeta proporciona varios bloques funcionales, cada bloque puede tener varios canales, y cada canal tiene uno o más pines. Resulta asi en una estructura que se asemeja a un árbol de directorios. Aunque halcmd no reconoce estructuras de árbol, la elección adecuada de las convenciones de nomenclatura dejará agrupar elementos relacionados (ya que ordena los nombres). Además, se pueden diseñar herramientas de más alto nivel para reconocer dicha estructura, si los nombres

proporcionan la información necesaria. Para hacer eso, todos los componentes HAL deberían seguir estas reglas:

- Los puntos (”.”) separan niveles de la jerarquía. Esto es análogo a la barra inclinada (”/”) en un nombre de archivo.
- Los guiones (”-”) separan palabras o campos en el mismo nivel de la jerarquía.
- Los componentes HAL no deben usar guiones bajos o ”mezcla de mayúsculas y minúsculas”. [1]
- Usar solo letras minúsculas y números en los nombres.

2.3. Convenciones de nombres de controladores de hardware

nota
En la versión 2.0, la mayoría de los controladores no siguen estas convenciones. Este capítulo es en realidad una guía para desarrollos futuros.

2.3.1. Nombres de pines/parámetros

Los controladores de hardware deben usar cinco campos (en tres niveles) para formar un pin o nombre de parámetro, de la siguiente manera:

```
<nombre-de-dispositivo>.<número-de-dispositivo>.<tipo-e-s>.<número-de-canal>.<nombre- ↩
    específico>
```

Los campos individuales son:

<nombre-de-dispositivo>
El dispositivo con el que el controlador está diseñado trabajar. Esto es a menudo una placa de interfaz de algún tipo, pero hay otras posibilidades.

<número-de-dispositivo>
Es posible instalar más de un(a) placa servo, puerto paralelo, u otro dispositivo de hardware en una computadora. El número de dispositivo identifica a uno en específico. Los números de dispositivo comienzan en 0 e incrementan.

<tipo-e-s>
La mayoría de los dispositivos proporcionan más de un tipo de E/S. Incluso el simple puerto paralelo tiene entradas y salidas digitales. Placas más complejas pueden tener entradas y salidas digitales, codificadores contadores, generadores pwm o de impulsos de pasos, convertidores analógico a digital y/o digital a analógico u otras capacidades únicas. El tipo de E/S se usa para identificar el tipo de E/S al que está asociado un pin o parámetro. Idealmente, los controladores que implementan el mismo tipo de E/S, incluso si son para dispositivos muy diferentes, deberían proporcionar un conjunto consistente de pines y parámetros, y un comportamiento idéntico. Por ejemplo, todas las entradas digitales deberían comportarse de la misma manera desde el interior del HAL, independientemente del dispositivo.

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

<número-de-canal>
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.

<nombre-específico>
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 de E/S, hay un conjunto estándar de pines y parámetros (denominada ”interfaz canónica”), que el controlador debería implementar. Las interfaces canónicas se describen en el capítulo Interfaces canónicas de dispositivos.

Ejemplos

motenc.0.encoder.2.position
La salida de posición del tercer canal (2) del codificador de la primera (0) placa Motenc.

stg.0.din.03.in
El estado de la cuarta entrada digital (03) en la primera (0) placa Servo-to-Go.

ppmc.0.pwm.00-03.frequency
La frecuencia de portadora utilizada para los canales PWM 0 a 3 (cuatro canales) en la primera (0) placa ppmc de Pico Systems.

2.3.2. Nombres de funciones

Los controladores de hardware generalmente solo tienen dos tipos de funciones HAL, unas que leen el hardware y actualizan los pines HAL, y otras que escriben en el hardware utilizando datos de pines HAL. Deben nombrarse de la siguiente manera:

```
<nombre-de-dispositivo>-<número-de-dispositivo>.<tipo-e-s>-<rango-número-de-canal>.read| ↩
   write
```

<nombre-de-dispositivo>
El mismo que se usa para pines y parámetros.

<número-de-dispositivo>
El dispositivo específico al que accederá la función.

<tipo-e-s>
Opcional. Una función puede acceder a todas las E/S de una placa, o puede acceder solo a cierto tipo. Por ejemplo, puede haber distintas funciones para leer codificadores contadores y leer E/S digitales. Si tales funciones independientes existen, el campo <tipo-e-s> identifica el tipo de E/S a la que acceden. Si una sola función lee todas las E/S provistas por la placa no se utiliza <tipo-e-s>. [3]

[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.

[3] Nota para los programadores de controladores: NO implementar funciones por separado para diferentes tipos de E/S a menos que sean interrumpibles y puedan trabajar en hilos independientes. Si se interrumpe una lectura de codificador leyendo entradas digitales y luego la reanudación de la lectura causa problemas, entonces implementar una sola función que lo haga todo.

<rango-número-de-canal>
Opcional. Se usa solo si <tipo-e-s> E/S se divide en grupos y se acceden por funciones diferentes.

read|write
Indica si la función lee el hardware o escribe en él.

Ejemplos

motenc.0.encoder.read
Lee todos los codificadores en la primera placa motenc.

generic8255.0.din.09-15.read
Lee el segundo puerto de 8 bits en la primera placa de E/S digital genérica basada en 8255.

ppmc.0.write
Escribe todas las salidas (generadores de pasos, pwm, DAC y digitales) en la primera placa ppmc de Pico Systems.

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 simplemente estén interesados en el funcionamiento de 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.

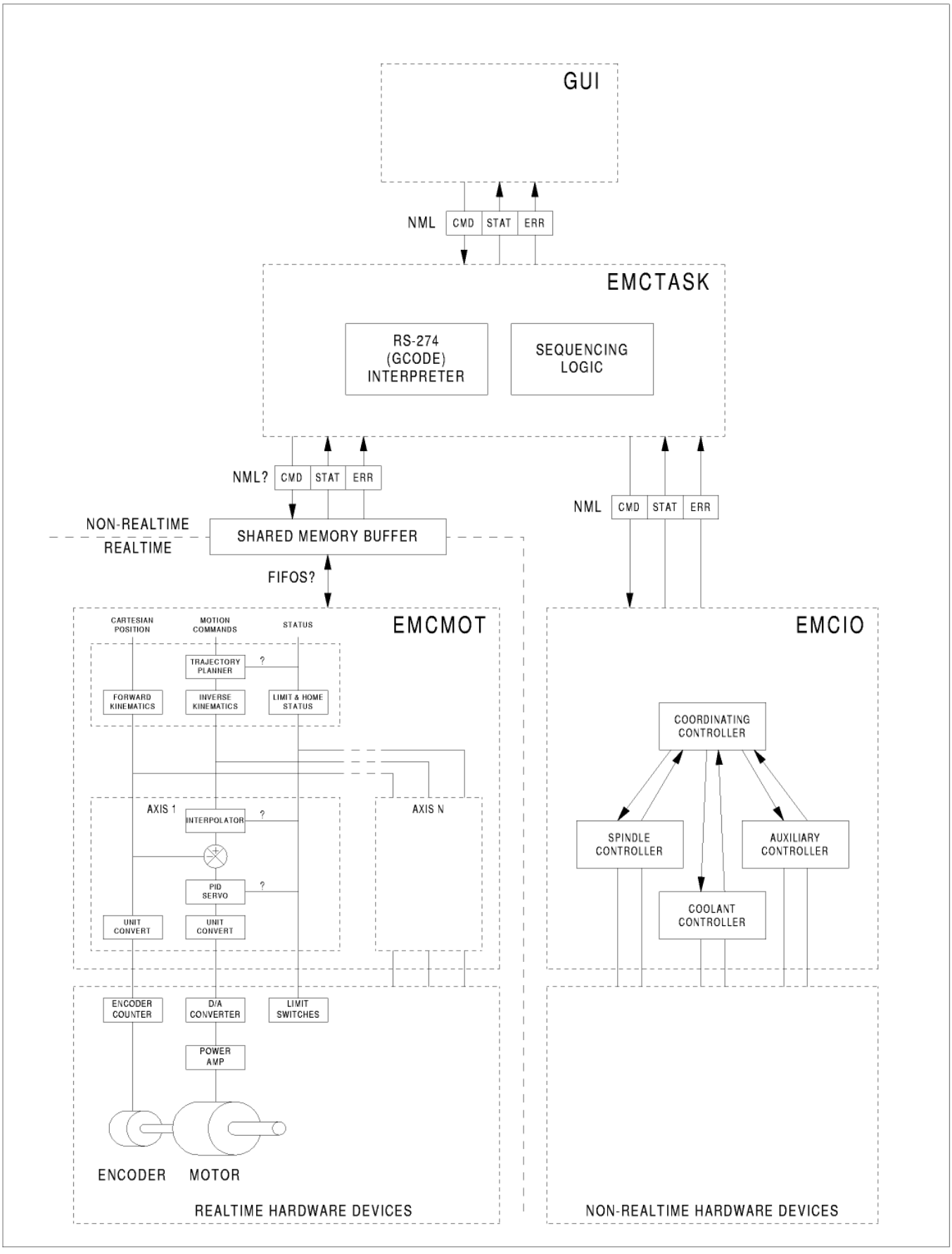
---

nota
En la versión 2.8 de LinuxCNC ya se hace esta distinción. El archivo INI cuenta con la nueva sección [JOINT_<núm>]. Muchos de los parámetros que antes eran propios de la sección [AXIS_<letra>] ahora están en la nueva sección. Otras secciones, como [KINS] también adquieren nuevos parámetros para ajustarse a esto. Se ha provisto una secuencia de comandos para transformar archivos INI antiguos a la nueva configuración ejes/articulaciones.

- *POSE* - Una pose es una posición completamente especificada en un espacio cartesiano 3D. 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* es el modo que probablemente se necesite si se hace trote con un hexápodo. Los comandos de trote implementados por el controlador de movimiento son trotes articulares, que funcionan en modo free. Pero si se desea mover un hexápodo o una máquina similar a lo largo de un eje cartesiano en particular, se debe operar más de una articulación. Para eso está *teleop*.

3.4. Descripción general de la arquitectura

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 varias interfaces de usuario en modo texto y gráfico. 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

Al nivel más general, LinuxCNC es un jerarquía de tres controladores: el manejador de comandos a nivel de tarea e intérprete de programa, el controlador de movimiento y el controlador de E/S discretas. El controlador de E/S discretas está implementado como una jerarquía de controladores, en este caso para husillo, refrigerante y subsistemas auxiliares (p. ej., Estop). El controlador de tareas coordina las acciones de los controladores de movimiento y de E/S discretas. Sus acciones están programadas en programas convencionales de control numérico ”código G y M”, que son interpretados por el controlador de tareas en mensajes NML y enviados al movimiento.

3.5. Introducción al controlador de movimiento

El controlador de movimiento es un componente de tiempo real. Recibe los comandos de control de movimiento desde las partes en tiempo no-real de LinuxCNC (p. ej. el intérprete de código G/Task, GUIs, etc.) y ejecuta esos comandos dentro de un contexto de tiempo real. La comunicación desde el contexto en tiempo no-real hacia el contexto en tiempo real ocurre mediante un mecanismo IPC pasa-mensajes que usa memoria compartida, y vía la Capa de abstracción de hardware (HAL).

El estado del controlador de movimiento esta disponible para el resto de LinuxCNC mediante el mismo IPC pasa-mensajes en memoria compartida y a través de HAL.

El controlador de movimiento interactúa con los controladores de motor y otro hardware en tiempo real y no-real usando 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.

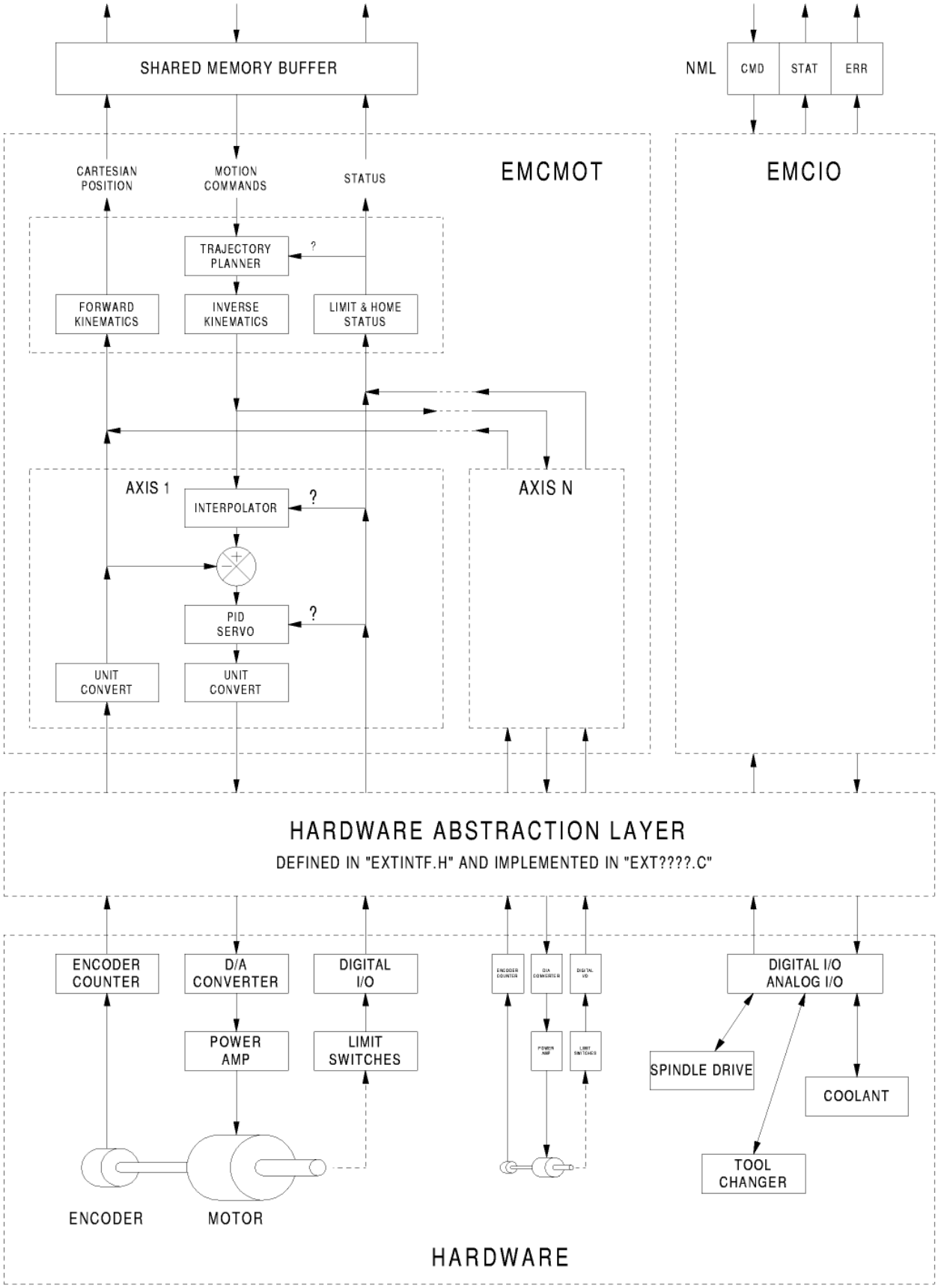
3.5.1. Módulos del controlador de movimiento

La funciones de tiempo real del controlador de movimiento están implementadas con módulos de tiempo real — objetos compartidos en espacio de usuario para sistemas Preempt-RT o módulos de kernel para algunas implementaciones de tiempo real en modo kernel como RTAI:

- *tpmod* - planificación de trayectoria
- *homemod* - funciones homing
- *motmod* - procesa comandos NML y controla hardware vía HAL
- *kinematics module* - realiza cinemáticas directas (articulaciones->coordenadas) e inversas (coordenadas->articulaciones)

LinuxCNC se inicia con una secuencia de comandos **linuxcnc** que lee un archivo de configuración INI e inicia todos los procesos necesarios. Para control de movimiento en tiempo real, la secuencia de comandos primero carga los módulos predeterminados tpmod y homemod y luego carga los módulos de cinemáticas y movimiento de acuerdo con las configuraciones de los archivos HAL especificados en el archivo INI.

En vez de usar los módulos predeterminados, se pueden usar módulos personalizados (compilados por el usuario) de homing o planificación de trayectoria mediante configuraciones en archivo INI u opciones de línea de comandos. Los módulos personalizados deben implementar todas las funciones usadas por los módulos predeterminados. La utilería halcompile puede crear módulos personalizados.


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
DIGITAL I/O ANALOG I/O
POWER AMP
LIMIT SWITCHES
SPINDLE DRIVE
COOLANT
TOOL CHANGER
ENCODER
MOTOR
HARDWARE

3.6. Diagrama de bloques y flujo de datos

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.

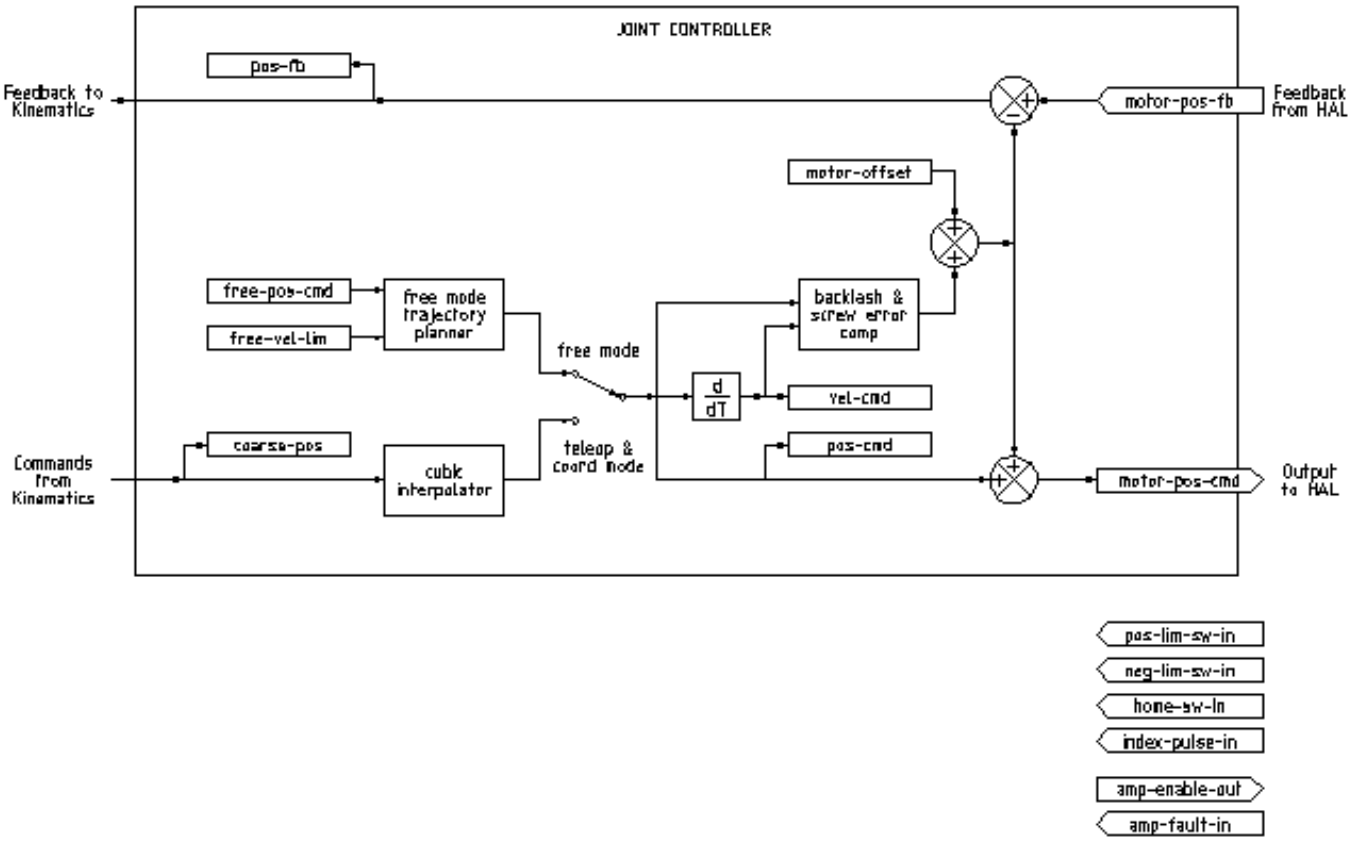


Figura 3.1: Diagrama de bloques del controlador articular

La figura anterior muestra cinco de los siete conjuntos de información de posición que forman el flujo principal de datos a través del controlador de movimiento. Las siete formas de datos de posición son las siguientes:

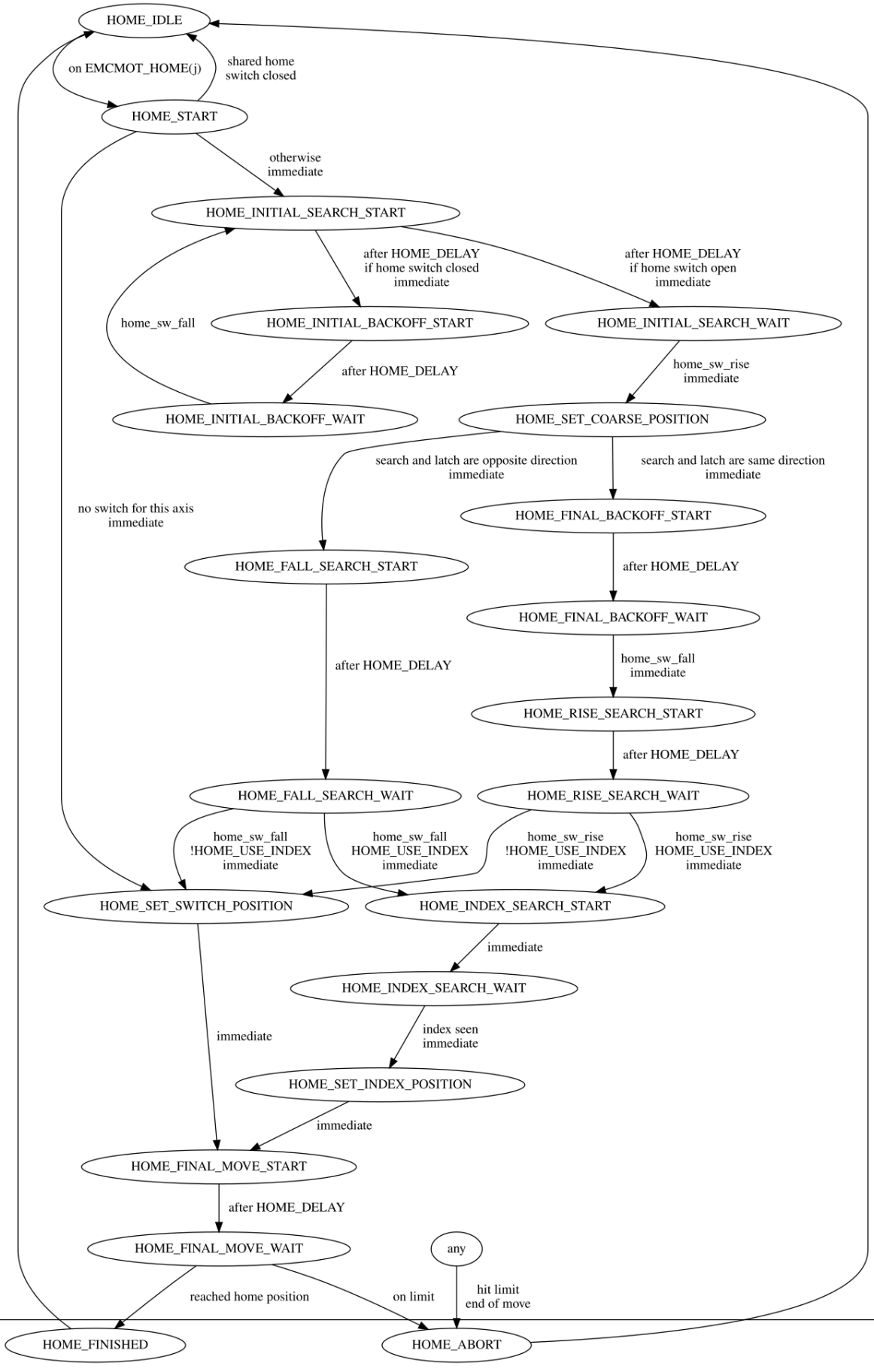
- *emcmotStatus->carte_pos_cmd* - Esta es la posición deseada, en coordenadas cartesianas. Se actualiza a tasa traj, no a tasa servo. En modo coord, se determina por el planificador traj. En modo teleop, está determinado por el planificador traj?. En modo libre, es copiado de actualPos, o generado mediante 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).

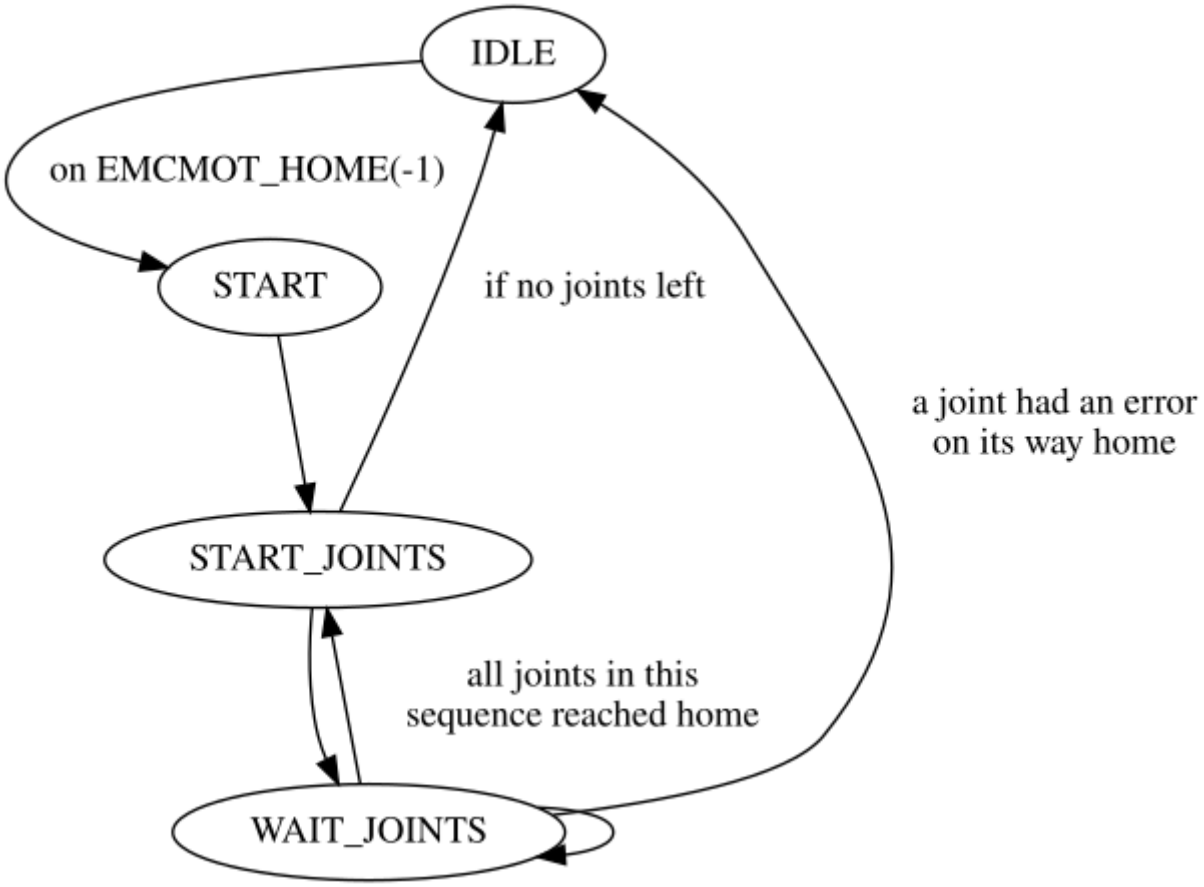
- *emcmotStatus->joints[n].pos_cmd* - Esta es la posición deseada, en coordenadas articulares, después de interpolación. En cada período servo se genera un nuevo conjunto de estas coordenadas. En modo coord se genera a partir de (2) por el interpolador. En modo teleop, se genera a partir de (2) por el interpolador. En modo free es generado por el planificador de trayectoria de modo libre.
- *emcmotStatus->joints[n].motor_pos_cmd* - Esta es la posición deseada, en coordenadas de motor. Las coordenadas del motor se generan agregando compensación de holgura mecánica, compensación de error del tornillo de avance y offset (para homing) a (3). Se genera de la misma manera independientemente del modo, y es la salida al bucle PID u otro bucle de posición.
- *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* - Esta es la posición real en coordenadas articulares. Se genera restando offsets, compensación de error del tornillo de avance y compensación de holgura mecánica de (5). Se genera de la misma manera, independientemente del modo operativo.
- *emcmotStatus->carte_pos_fb* - Esta es la posición real, en coordenadas cartesianas. Se actualiza a tasa traj, no a tasa servo. Idealmente, actualPos siempre se calcularía aplicando cinemática directa a (6). Sin embargo, la cinemática directa puede no estar disponible, o pueden ser inutilizable porque uno o más ejes no están en casa. En ese caso, las opciones son: A) fingirla, copiando (1), o B) admitir que realmente no se conocen las coordenadas cartesianas, y simplemente no actualizar actualPos. Cualquiera que sea el enfoque utilizado, no veo ninguna razón para no hacerlo de la misma manera, independientemente del modo de operación. Yo propondría lo siguiente: si hay cinemática directa, usarla, a menos que no funcionen debido a ejes no-en casa u otros problemas, en cuyo caso hacer (B). Si no hay cinemática directa, hacer (A), ya que de lo contrario actualPos *nunca* estará actualizada.

3.7. Homing

3.7.1. Diagrama de estado de homing



3.7.2. Otro diagrama de homing



3.8. Comandos

Los comandos se implementan mediante una gran instrucción switch en la función emcmotCommandHandler(), que se llama a la tasa servo. Más sobre esa función más adelante.

Hay aproximadamente 44 comandos - esta lista todavía está en construcción.

---

nota
La enumeración cmd_code_t, en motion.h, contiene 73 comandos, pero la instrucción switch en command.c contempla sólo 70 comandos (hasta 5/jun/2020). Los comandos ENABLE_WATCHDOG / DISABLE_WATCHDOG están en motion-logger.c. Quizás estén obsoletos. El comando SET_TELEOP_VECTOR solo aparece en motion-logger.c, sin más efecto que su propio 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. Requisitos

Ninguno. El comando siempre se acepta y actúa inmediatamente.

3.8.1.2. Resultados

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

El comando FREE pone el controlador de movimiento en modo libre. Modo libre significa que cada articulación es independiente de todas las demás articulaciones. Coordenadas cartesianas, poses y cinemáticas se ignoran cuando está en modo libre. En esencia, cada articulación tiene su propio planificador de trayectoria simple, y cada articulación ignora por completo las otras articulaciones. Algunos comandos (como JOG y HOME de articulación) solo funcionan en modo libre. Otros comandos, incluso cualquier cosa que trate con coordenadas cartesianas, no funciona en absoluto en modo libre.

3.8.2.1. Requisitos

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. Resultados

Si la máquina ya está en modo libre, nada. De lo contrario, la máquina se coloca en modo libre. Cada planificador de trayectoria en modo libre de articulación se inicializa con la ubicación actual de la articulación, pero los planificadores no están habilitados y las articulaciones están estacionadas.

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. Requisitos

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. Resultados

Si la máquina ya está en modo teleop, nada. De lo contrario la máquina se coloca en modo teleop. El código cinemático está activado, los interpoladores son drenados y enjuagados, y los comandos de velocidad cartesiana se ponen a cero.

3.8.4. COORD

El comando COORD coloca la máquina en modo coordinado. En modo coord, el movimiento de la máquina se basa en coordenadas cartesianas utilizando cinemáticas, en lugar de articulaciones individuales como en modo libre. Además, el planificador de trayectoria principal se utiliza para generar movimiento, con base en comandos LINE, CIRCLE y/o PROBE en cola. El modo coord es el modo que se usa al ejecutar un programa de código G.

3.8.4.1. Requisitos

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. Resultados

Si la máquina ya está en modo coord, nada. De lo contrario, la máquina se coloca en modo coord. El código cinemático está activado, los interpoladores son drenados y enjuagados, y las colas de planificador de trayectoria son vaciadas. El planificador de trayectoria está activo y en espera de un comando LINE, CIRCLE o PROBE.

3.8.5. ENABLE

El comando ENABLE habilita el controlador de movimiento.

3.8.5.1. Requisitos

Ninguna. El comando puede emitirse en cualquier momento y siempre será aceptado.

3.8.5.2. Resultados

Si el controlador ya está habilitado, nada. Si no, el controlador es habilitado. Las colas y los interpoladores se drenan. Cualquier movimiento u operación de homing se finaliza. Las salidas amp-enable asociadas con articulaciones activas se encienden. Si no hay cinemática directa disponible, la máquina se cambia al modo libre.

3.8.6. DISABLE

El comando DISABLE deshabilita el controlador de movimiento.

3.8.6.1. Requisitos

Ninguna. El comando puede emitirse en cualquier momento y siempre será aceptado.

3.8.6.2. Resultados

Si el controlador ya está deshabilitado, nada. Si no, el controlador está desactivado. Las colas y los interpoladores se drenan. Cualquier movimiento u operación de homing se finaliza. Las salidas amp-enable asociadas con las articulaciones activas son apagadas. Si no hay cinemática directa disponible, la máquina se cambia al modo libre.

3.8.7. ENABLE_AMPLIFIER

El comando ENABLE_AMPLIFIER activa la salida de habilitación del amplificador para un amplificador de salida única, sin cambiar nada más. Puede ser usado para habilitar un controlador de velocidad del husillo.

3.8.7.1. Requisitos

Ninguna. El comando puede emitirse en cualquier momento y siempre será aceptado.

3.8.7.2. Resultados

Actualmente nada. (Una llamada a la antigua función extAmpEnable esta actualmente comentada). Eventualmente establecerá el pin HAL de habilitación del amplificador a TRUE.

3.8.8. DISABLE_AMPLIFIER

El comando DISABLE_AMPLIFIER apaga la salida de habilitación del amplificador para un solo amplificador, sin cambiar nada más. De nuevo, útil para controladores de velocidad del husillo.

3.8.8.1. Requisitos

Ninguna. El comando puede emitirse en cualquier momento y siempre será aceptado.

3.8.8.2. Resultados

Actualmente nada. (Una llamada a la antigua función extAmpEnable está actualmente comentada). Eventualmente establecerá el pin HAL de habilitación del amplificador en 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. Requisitos

Ninguna. El comando puede emitirse en cualquier momento y siempre será aceptado.

3.8.9.2. Resultados

Los cálculos para la articulación especificada están habilitados. El pin de habilitación del amplificador no se cambia, sin embargo, cualquier comando ENABLE o DISABLE posterior modificará el pin de habilitación del amplificador de la articulación.

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. Requisitos

Ninguna. El comando puede emitirse en cualquier momento y siempre será aceptado.

3.8.10.2. Resultados

Los cálculos para la articulación especificada están habilitados. El pin de habilitación del amplificador no se cambia, y los comandos ENABLE o DISABLE subsecuentes no modificarán el pin de habilitación del amplificador de la articulación.

3.8.11. ENABLE_WATCHDOG

El comando ENABLE_WATCHDOG habilita un perro guardián basado en hardware (si está presente).

3.8.11.1. Requisitos

Ninguna. El comando puede emitirse en cualquier momento y siempre será aceptado.

3.8.11.2. Resultados

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

El comando DISABLE_WATCHDOG deshabilita un perro guardián basado en hardware (si está presente).

3.8.12.1. Requisitos

Ninguna. El comando puede emitirse en cualquier momento y siempre será aceptado.

3.8.12.2. Resultados

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

El comando PAUSE detiene el planificador de trayectoria. No tiene efecto en modo libre o teleop. En este punto no sé si detiene todo el movimiento inmediatamente, o si completa el movimiento actual y luego se detiene antes de jalar otro movimiento de la cola.

3.8.13.1. Requisitos

Ninguna. El comando puede emitirse en cualquier momento y siempre será aceptado.

3.8.13.2. Resultados

El planificador de trayectoria hace una pausa.

3.8.14. RESUME

El comando RESUME reinicia el planificador de trayectoria si está en pausa. No tiene efecto en modo libre o teleop, o si el planificador no está en pausa.

3.8.14.1. Requisitos

Ninguna. El comando puede emitirse en cualquier momento y siempre será aceptado.

3.8.14.2. Resultados

Se reanuda el planificador de trayectoria.

3.8.15. STEP

El comando STEP reinicia el planificador de trayectoria si está en pausa, y le dice al planificador que se detenga nuevamente cuando llegue a un punto específico. No tiene efecto en modo libre o teleop. En este punto no se exactamente cómo funciona esto. Agregaré más documentación aquí cuando excave más profundo en el planificador de trayectoria.

3.8.15.1. Requisitos

Ninguna. El comando puede emitirse en cualquier momento y siempre será aceptado.

3.8.15.2. Resultados

El planificador de trayectoria se reanuda y luego se detiene cuando llega a un punto específico.

3.8.16. SCALE

El comando SCALE escala todos los límites de velocidad y comandos por una cantidad especificada. Se utiliza para implementar la anulación de velocidad de alimentación y otros funciones similares. El escalado funciona en modo libre, teleop y coord, y afecta todo, incluidas las velocidades de homing, etc. Sin embargo, los límites individuales de velocidad de articulación no se ven afectados.

3.8.16.1. Requisitos

Ninguna. El comando puede emitirse en cualquier momento y siempre será aceptado.

3.8.16.2. Resultados

Todos los comandos de velocidad son escalados por la constante especificada.

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. Requisitos

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

3.8.17.2. Resultados

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

El comando HOME inicia una secuencia de homing en una articulación especificada. La secuencia homing real está determinada por unos parámetros de configuración, y puede variar desde simplemente establecer la posición actual a cero, hasta una búsqueda en varias etapas de un interruptor de casa y pulso de índice, seguido de un movimiento a una ubicación de casa arbitraria. Para más información sobre la secuencia de homing consultar el Manual del integrador.

3.8.18.1. Requisitos

El comando se ignorará en silencio a menos que la máquina esté en modo libre.

3.8.18.2. Resultados

Se anula cualquier trote o movimiento de articulación y se inicia la secuencia de homing.

3.8.19. JOG_CONT

El comando JOG_CONT inicia un trote continuo en una sola articulación. Se genera un trote continuo al establecer la posición objetivo del planificador de trayectoria en modo libre a un punto más allá del final del rango de carrera de la articulación. Esto asegura que el planificador moverá constantemente hasta que sea detenido por los límites de articulación o por un comando ABORT. Normalmente, una GUI envía un comando JOG_CONT cuando el usuario presiona un botón de trote, y un ABORT al soltar el botón.

3.8.19.1. Requisitos

El controlador de comandos rechazará el comando JOG_CONT con un mensaje de error si la máquina no está en modo libre, o si alguna articulación está en movimiento (GET_MOTION_INPOS_FLAG () == FALSE), o si el movimiento no está habilitado. También ignorará silenciosamente el comando si la articulación ya está en (o más allá de) su límite y el trote ordenado lo empeoraría.

3.8.19.2. Resultados

El planificador de trayectoria de modo libre para la articulación identificada por emcmotCommand->eje es activado, con una posición destino más allá del final de carrera de la articulación, y un límite de velocidad de emcmotCommand->vel. Este comienza el movimiento de la articulación, y el movimiento continuará hasta ser detenido por un comando ABORT o al alcanzar un límite. El planificador de modo libre acelera al límite de aceleración de articulación al comienzo del movimiento, y desacelerará al límite de aceleración de articulación cuando se detenga.

3.8.20. JOG_INCR

El comando JOG_INCR inicia un trote gradual en una sola articulación. Los trotes incrementales son acumulativos, en otras palabras, emitir dos comandos JOG_INCR pidiendo cada uno 0.100 pulgadas de movimiento resultarán en 0.200 pulgadas de recorrido, incluso si el segundo comando se emite antes que termine el primero. Normalmente, los trotes incrementales se detienen cuando ya recorrieron la distancia deseada, sin embargo, también se detienen cuando topan con un límite, o con un comando ABORT.

3.8.20.1. Requisitos

El controlador de comandos rechazará silenciosamente el comando JOG_INCR si la máquina no está en modo libre, o si alguna articulación está en movimiento (GET_MOTION_INPOS_FLAG () == FALSE), o si el movimiento no está habilitado. También ignorará silenciosamente el comando si la articulación ya está en (o más allá de) su límite y el trote ordenado lo empeoraría.

3.8.20.2. Resultados

El planificador de trayectoria de modo libre para la articulación identificada por emcmotCommand->eje está activado, la posición destino es incrementada/decrementada por emcmotCommand->offset, y el límite de velocidad se establece en emcmotCommand->vel. El planificador de trayectoria de modo libre generará un movimiento trapezoidal suave desde la posición actual hasta la posición destino. El planificador puede manejar correctamente los cambios en posición objetivo que ocurran mientras el movimiento este en progreso, por lo que se pueden emitir múltiples comandos JOG_INCR en sucesión rápida. El planificador de modo libre acelera al límite de aceleración de articulación al comienzo del movimiento, y desacelerará al límite de aceleración de articulación para detenerse en el posición destino.

3.8.21. JOG_ABS

El comando JOG_ABS inicia un trote absoluto en una sola articulación. Un trote absoluto es un simple movimiento a una ubicación específica en coordenadas articulares. Normalmente los trotes absolutos se detienen cuando alcanzan la ubicación deseada, sin embargo, también se detienen cuando alcanzan un límite, o con un comando ABORT.

3.8.21.1. Requisitos

El controlador de comandos rechazará silenciosamente el comando JOG_ABS si la máquina no está en modo libre, o si alguna articulación está en movimiento (GET_MOTION_INPOS_FLAG () == FALSE), o si el movimiento no está habilitado. También ignorará silenciosamente el comando si la articulación ya está en (o más allá de) su límite y el trote ordenado lo empeoraría.

3.8.21.2. Resultados

El planificador de trayectoria de modo libre para la articulación identificada por emcmotCommand->axis está activado, la posición destino se establece a emcmotCommand->offset, y el límite de velocidad se establece a emcmotCommand->vel. El planificador de trayectoria de modo libre generará un movimiento trapezoidal suave desde la posición actual hasta la posición destino. El planificador puede manejar correctamente los cambios a la posición destino que sucedan mientras el movimiento está en progreso. Si varios comandos JOG_ABS se emiten en sucesión rápida, cada nuevo comando cambia la posición destino y la máquina pasa a la posición final ordenada. El planificador de modo libre acelera al límite de aceleración de articulación en el comienzo del movimiento, y desacelerará al límite de aceleración de articulación para detenerse en la posición destino.

3.8.22. SET_LINE

El comando SET_LINE agrega una línea recta a la cola del planificador de trayectoria.

(Más tarde)

3.8.23. SET_CIRCLE

El comando SET_CIRCLE agrega un movimiento circular a la cola del planificador de trayectoria.

(Más tarde)

3.8.24. SET_TELEOP_VECTOR

El comando SET_TELEOP_VECTOR indica al controlador de movimiento que se mueva a lo largo de un vector específico en el espacio cartesiano.

(Más tarde)

3.8.25. PROBE

El comando PROBE indica al controlador de movimiento que se mueva hacia un punto específico en el espacio cartesiano, deteniendo y grabando su posición si se dispara la entrada de la sonda.

(Más tarde)

3.8.26. CLEAR_PROBE_FLAG

El comando CLEAR_PROBE_FLAG se usa para restablecer la entrada de la sonda en preparación para un comando PROBE. (Pregunta: ¿por qué el comando PROBE no debería restablecer automáticamente la entrada?)

(Más tarde)

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 más 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.

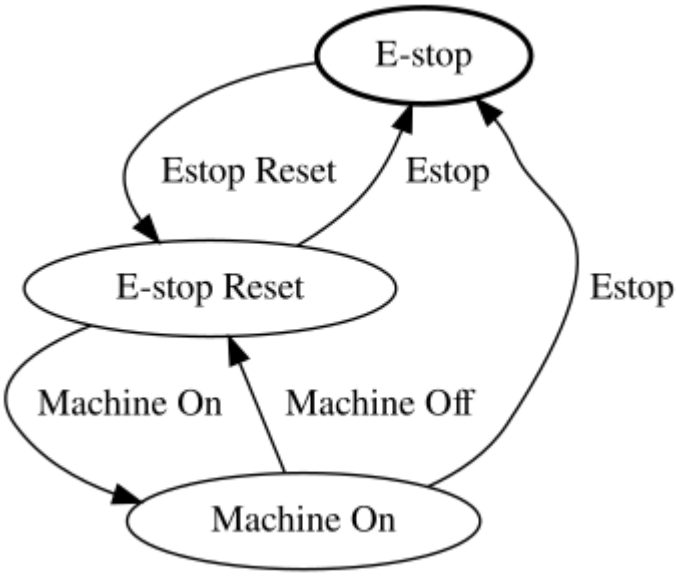
3.9. Compensación de error de tornillo y holgura mecánica

FIXME Compensación de holguras y errores de tornillos

3.10. Controlador de tareas (EMCTASK)

3.10.1. Estado

Task tiene tres estados internos posibles: **E-stop**, **E-stop Reset**, y **Machine on**.



3.11. Controlador E/S (EMCIO)

El controlador de E/S es parte de TASK. Interactúa con E/S externas usando pines HAL.

Actualmente ESTOP/Enable, el refrigerante, y el cambio de herramienta se manejan con iocontrol. Estos son eventos de velocidad relativamente baja; las E/S coordinadas de alta velocidad se manejan en motion.

emctaskmain.cc envía comandos de E/S mediante taskclass.cc.

Proceso del bucle principal de iocontrol:

- comprueba si las entradas HAL han cambiado
- comprueba si read_tool_inputs() indica que el cambio de herramienta ha finalizado y establece emcioStatus.status

3.12. Interfaces de usuario

Interfaces de usuario FIXME

3.13. Introducción a libnml

libnml se deriva de rcslib del NIST sin todo el soporte multi-plataformas. Muchas de las envolturas de código específico de plataforma han sido eliminadas, 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.

Este capítulo no pretende ser una guía definitiva para usar libnml (o rcslib); en cambio proporcionará eventualmente una visión general de cada clase C++ y sus funciones miembro. Inicialmente, la mayoría de estas notas se agregarán como comentarios aleatorios a medida que el código se analice y modifique.

3.14. LinkedList

Clase base para mantener una lista enlazada. Este es uno de los principales bloques de construcción utilizados para pasar mensajes NML y una variedad de estructuras de datos internas.

3.15. LinkedListNode

Clase base para producir una lista enlazada. Su propósito es mantener punteros a los nodos anteriores y siguientes, puntero a los datos y el tamaño de los datos.

No asigna memoria para el almacenamiento de datos.

3.16. SharedMemory

Proporciona un bloque de memoria compartida junto con un semáforo (heredado de la clase Semaphore). La creación y destrucción del semáforo es manejado por el constructor y destructor SharedMemory.

3.17. ShmBuffer

Clase para pasar mensajes NML entre procesos locales mediante memoria intermedia de uso compartido. Gran parte del funcionamiento interno se hereda de la clase CMS.

3.18. Timer

La clase Timer proporciona un temporizador periódico limitado solo por la resolución del reloj del sistema. Si, por ejemplo, un proceso necesita ser ejecutado cada 5 segundos, independientemente del tiempo que lleve ejecutar el proceso, el siguiente fragmento de código muestra cómo hacerlo:

```
main()
{
    timer = new Timer(5.0);    /* Inicializa un temporizador con un ciclo de 5 segundos */
    while(0) {
        /* Hacer algún proceso */
        timer.wait();    /* Espera hasta el siguiente intervalo de 5 segundos */
    }
    delete timer;
}
```

3.19. Semaphore

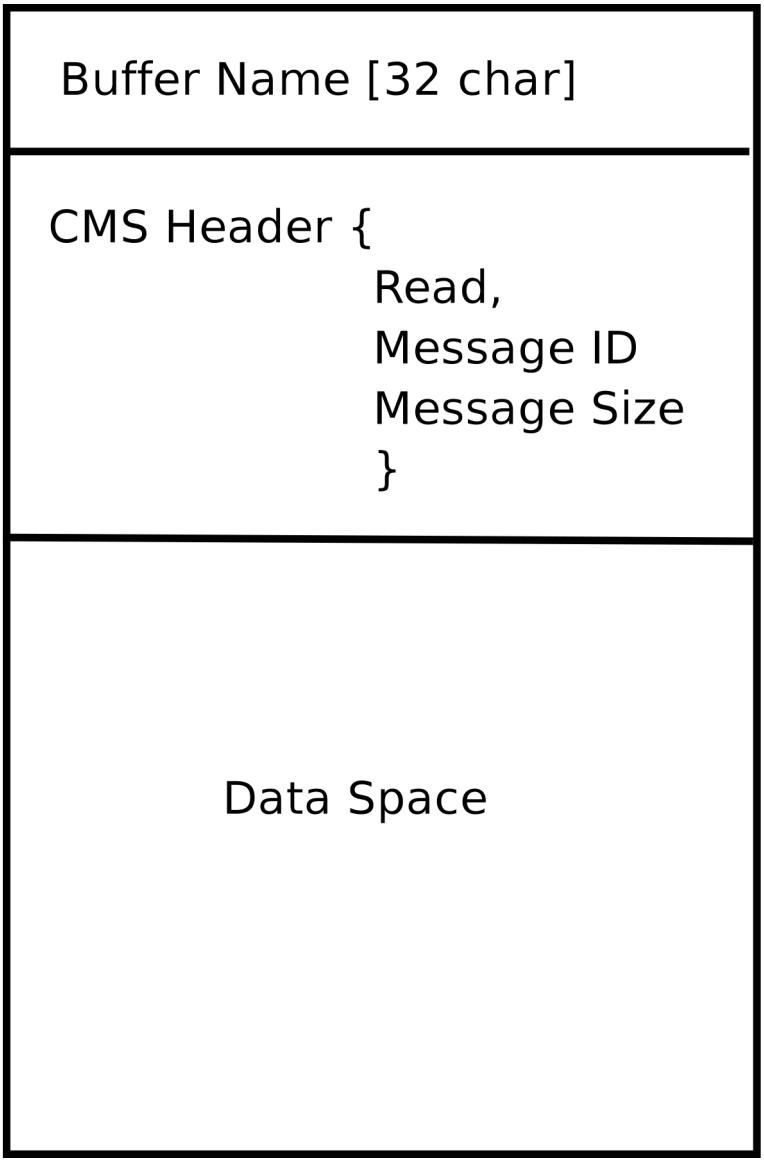
La clase Semaphore proporciona un método de exclusiones mutuas para acceder a un recurso compartido. La función para obtener un semáforo puede bloquear hasta que el acceso esté disponible, regresar después de un tiempo de espera o regresar inmediatamente con o sin obtener el semáforo. El constructor creará un semáforo o adjuntará a uno existente si la ID ya está en uso.

Semaphore::destroy() debe ser invocado solo por el último proceso.

3.20. CMS

En el corazón de libnml está la clase CMS. Contiene la mayor parte de las funciones utilizadas por libnml y finalmente NML. Muchas de las funciones internas se sobrecargan para permitir métodos de paso de datos dependientes de hardware específico. Al final, todo gira en torno a un bloque central de memoria (denominado ”búfer de mensajes” o simplemente *búfer*). Este búfer puede existir como un bloque de memoria compartida accedido por otros procesos CMS/NML, o un búfer local y privado para la transmisión de datos por red o interfaces seriales.

El búfer se asigna dinámicamente en tiempo de ejecución para permitir una mayor flexibilidad del subsistema CMS/NML. El tamaño del búfer debe ser suficientemente grande para acomodar el mensaje más grande, una pequeña cantidad para uso interno y permitir que el mensaje se codifique si se elige esta opción (los datos codificados se cubrirán más adelante). La figura siguiente es una vista interna del espacio del búfer.



Búfer de CMS La clase base de CMS es la principal responsable de crear las vías de comunicación e interfazar con el sistema operativo.

3.21. Formato del archivo de configuración

La configuración NML consta de dos tipos de formatos de línea. Uno para búfers y un segundo para procesos que se conectan a los búfers.

3.21.1. Línea de búfer

El formato NIST original de la línea de búfer es:

- *B nombre tipo host tamaño neut RPC# búfer# máx_procs clave [configuraciones específicas por tipo]*
- *B*- identifica la línea como una configuración de búfer.
- *nombre*- es el identificador del búfer.
- *tipo*- describe el tipo de búfer: SHMEM, LOCMEM, FILEMEM, PHANTOM o GLOBMEM.
- *host*- es una dirección IP o un nombre de host para el servidor NML
- *tamaño*- es el tamaño del búfer
- *neut*- un booleano para indicar si los datos en el búfer están codificados en un formato independiente de la máquina, o sin formato.
- *RPC#*- obsoleto - marcador de posición retenido solo para compatibilidad con versiones anteriores.
- *búfer#*- un número identificador único que se usa si un servidor controla varios búfers.
- *máx_procs*- número máximo de procesos permitidos para conectarse a este búfer.
- *clave* - es un identificador numérico para un búfer de memoria compartida

3.21.2. Configuraciones específicas por tipo

El tipo de búfer implica opciones de configuración adicionales mientras que el sistema operativo anfitrión impide ciertas combinaciones. En una tentativa de concretar la documentación publicada en un formato coherente, solo será cubierto el tipo de buffer **SHMEM**.

- *mutex=os_sem*- modo predeterminado para proporcionar el semáforo de bloqueo de la memoria intermedia.
- *mutex=none*- no utilizado
- *mutex=no_interrupts* - no aplicable en un sistema Linux
- *mutex=no_switching* - no aplicable en un sistema Linux
- *mutex=mao split*- divide el búfer en la mitad (o más) y permite que un proceso acceda a una parte del búfer mientras que un segundo proceso está escribiendo en la otra parte.
- *TCP=(número de puerto)*- especifica qué puerto de red utilizar.
- *UDP =(número de puerto)* - ídem
- *STCP =(número de puerto)* - ídem
- *serialPortDevName=(puerto serie)* - sin documentar.

- *passwd=file_name.pwd*- agrega una capa de seguridad al búfer requiriendo que cada proceso proporcione una contraseña.
- *bsem*- la documentación del NIST implica una clave para un semáforo de bloqueo, y si bsem=-1, se evitan los bloqueos de lectura.
- *queue*- permite pasar mensajes en cola.
- *ascii* - Codifica mensajes en formato de texto plano
- *disp*- codifica los mensajes en un formato adecuado para mostrarlos (???)
- *xdr*- codifica mensajes en Representación de Datos Externos. (Ver rpc/xdr.h para más detalles).
- *diag*- habilita almacenado de diagnósticos en el búfer (¿temporizaciones y recuentos de bytes?)

3.21.3. Línea de proceso

El formato NIST original de la línea de proceso es:

P nombre búfer tipo host ops servidor tiempo master c_num [configuraciones específicas por tipo]

- *P*- identifica esta línea como una configuración de proceso.
- *nombre*- es el identificador del proceso.
- *búfer* - es uno de los búfers definidos en otra parte del archivo de configuración.
- *tipo*- define si este proceso es local o remoto en relación con el búfer.
- *host*- especifica en qué parte de la red se está ejecutando este proceso.
- *ops*- proporciona al proceso acceso de solo lectura, solo escritura o de lectura/escritura al búfer.
- *servidor*- especifica si este proceso ejecutará un servidor para este búfer.
- *tiempo* - establece las características de tiempo de espera para los accesos al búfer.
- *master* - indica si este proceso es responsable de crear y destruir el búfer.
- *c_num* - un número entero entre cero y (máx_procs -1)

3.21.4. Comentarios de configuración

Algunas de las combinaciones de configuración no son válidas, mientras que otras implican ciertas restricciones. En un sistema Linux GLOBMEM es obsoleto, mientras que PHANTOM solo es realmente útil en la etapa de prueba de una aplicación; igualmente para FILEMEM. LOCMEM es de poca utilidad para una aplicación multiproceso, y solo ofrece ventajas limitadas de rendimiento sobre SHMEM. Esto deja a SHMEM como el único tipo de búfer para usar con LinuxCNC.

La opción neut solo se usa en un sistema multiprocesador donde arquitecturas diferentes (e incompatibles) comparten un bloque de memoria. La probabilidad de ver un sistema de este tipo fuera de un museo o lugar de investigación es remoto y solo es relevante para buffers GLOBMEM.

El número RPC está documentado como obsoleto y solo se conserva por razones de compatibilidad.

Con un nombre de búfer único, tener una identidad numérica parece ser inútil. Es necesario revisar el código para identificar la lógica. Asimismo, el campo clave parece ser redundante, y podría derivarse del nombre del búfer.

El propósito de limitar el número de procesos permitidos para conectarse a cualquier búfer no está claro a partir de la documentación existente y del código fuente original. Permitir un número no especificado de procesos para conectarse a un búfer no es más difícil de implementar.

Los tipos mutex se reducen a uno de estos dos; el predeterminado ”os_sem” o ”mao split”. La mayoría de los mensajes NML son relativamente cortos y se pueden copiar hacia o desde el búfer con retrasos mínimos, por lo que las lecturas divididas no son esenciales.

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.

Los protocolos UDP tienen menos verificaciones en los datos y permiten descartar un porcentaje de paquetes. TCP es más confiable, pero es relativamente más lento.

Si LinuxCNC se va a conectar a una red, se esperaría que sea local y detrás de un cortafuegos. La única razón para permitir el acceso a LinuxCNC a través de internet sería para diagnósticos remotos. Esto puede lograrse de manera mucho más segura utilizando otros medios, tal vez por una interfaz web.

El comportamiento exacto cuando el tiempo de espera 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:

tiempo > 0
: Bloqueo de acceso hasta que se alcanza el intervalo de tiempo de espera o el acceso al búfer esté disponible.

tiempo = 0
: El acceso al búfer solo es posible si no hay otro proceso que esté leyendo o escribiendo en ese momento.

tiempo < 0 o INF
: El acceso está bloqueado hasta que el búfer esté disponible.

3.22. Clase base NML

Expande las listas y la relación entre NML, NMLmsg y las clases cms de nivel inferior.

No debe confundirse con NMLmsg, RCS_STAT_MSG o RCS_CMD_MSG.

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

- búferes cms creados o conectados.
- procesos y búferes a las que se conectan
- una larga lista de funciones de formato para cada tipo de mensaje

Este último elemento es probablemente el núcleo de gran parte de la desalineación de libnml/rcslib y NML en general. Cada mensaje que se pasa a través de NML requiere que se adjunte una cierta cantidad de información además de los datos reales. Para hacer esto, se invocan en secuencia varias funciones de formato para ensamblar fragmentos del mensaje general. Las funciones de formato incluirán NML_TYPE, MSG_TYPE, además de los datos declarados en clases NMLmsg derivadas. Los cambios en el orden en que se llaman las funciones de formato y también las variables pasadas pueden romper la compatibilidad con rcslib si se hacen mal. Hay razones para mantener la compatibilidad con rcslib y buenas razones para alterar el código. La pregunta es ¿qué conjunto de razones son primordiales?

3.22.1. Interioridades de NML

3.22.1.1. Constructor NML

NML::NML() analiza el archivo de configuración y lo almacena en una lista enlazada para ser pasada a constructores cms en líneas simples. Es la función del constructor NML para llamar al constructor cms relevante para cada búfer y mantener una lista de los objetos cms y los procesos asociados con cada búfer.

NML puede interactuar con cms desde los punteros almacenados en las listas y es por esto que Doxygen no muestra las relaciones reales involucradas.

---

nota
La configuración se almacena en la memoria antes de pasar un puntero a una línea específica para el constructor cms. El constructor cms analiza luego la línea nuevamente para extraer un par de variables ... Tendría más sentido hacer TODO el análisis y guardar las variables en una estructura que sea pasada al constructor cms. Esto eliminaría la manipulación de cadenas y reduciría el código duplicado en cms ...

3.22.1.2. Lectura/escritura NML

Las llamadas a NML::read y NML::write realizan tareas similares en el modo de procesar el mensaje; la única variación real está en el dirección del flujo de datos.

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. Dentro de format_xxx() se realiza el trabajo de construir o deconstruir el mensaje. 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. Relaciones NMLmsg y NML

NMLmsg es la clase base de la que se derivan todas las clases de mensajes. Cada clase de mensaje debe tener un ID único definido (y pasado al constructor) y también una función update(*cms). update() será llamado por las funciones de lectura/escritura NML cuando se llama al formateador NML - El puntero al formateador habrá sido declarado en el constructor NML en algún momento. En virtud de las listas enlazadas que crea NML, puede seleccionar el puntero cms que se pasa al formateador y, por tanto, que búfer se utilizará.

3.23. Agregar comandos NML personalizados

LinuxCNC es bastante impresionante, pero algunas partes necesitan algunos ajustes. Como ya sabe, la comunicación se realiza a través de canales NML. Los datos enviados a través de tales canales es una de las clases definidas en emc.hh (implementado en emc.cc). Si alguien necesita un tipo de mensaje que no existe, debería seguir estos pasos para agregar uno nuevo. (El mensaje que se agrega en el ejemplo se llama EMC_IO_GENERIC (hereda EMC_IO_CMD_MSG (hereda RCS_CMD_MSG)))

1. agregar la definición de la clase EMC_IO_GENERIC a emc2/src/emc/nml_intf/emc.hh
2. agregar la definición de tipo: #define EMC_IO_GENERIC_TYPE \((NMLTYPE) 1605)

a. (Se elige 1605 porque esta disponible) en emc2/src/emc/nml_intf/emc.hh

3. agregar el caso EMC_IO_GENERIC_TYPE a emcFormat en emc2/src/emc/nml_intf/emc.cc
4. agregar el caso EMC_IO_GENERIC_TYPE a emc_symbol_lookup en emc2/src/emc/nml_intf/emc.cc
5. agregar la función EMC_IO_GENERIC::update a emc2/src/emc/nml_intf/emc.cc

Al recompilar, el nuevo mensaje debería estar allí. La siguiente parte es enviar tales mensajes desde algún lugar y recibirlos en otro lugar, y hacer algunas cosas con eso.

3.24. La tabla de herramientas y el cambiador de herramientas

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. Abstracción del cambiador de herramientas en LinuxCNC

LinuxCNC admite dos tipos de hardware de cambiador de herramientas, llamados *no-aleatorio* y *aleatorio*. La configuración INI [EMCIO]RANDOM_TOOLCHANGER controla cuál de estos tipos de hardware considera LinuxCNC que está conectado.

3.24.1.1. Cambiadores de herramientas no aleatorios

El hardware de cambiador de herramientas no aleatorio vuelve a colocar cada herramienta en la ranura desde la que fue originalmente cargada.

Ejemplos de hardware de cambiador de herramientas no aleatorio son el cambiador de herramientas ”manual”, torretas de herramientas de torno y cambiadores de herramientas en rack.

Cuando se configura para un cambiador de herramientas no aleatorio, LinuxCNC no cambia el número de ranura en el archivo de la tabla de herramientas a medida que las herramientas se cargan y descargan. Internamente, en el cambio de herramienta la información de la herramienta se **copia** de la ranura fuente de la tabla de herramientas a la ranura 0 (que representa el husillo), reemplazando cualquier información de herramienta que estaba allí anteriormente.

---

nota
Con LinuxCNC configurado para cambiador de herramientas no aleatorio, la herramienta 0 (T0) tiene significado especial: ”sin herramienta”. T0 puede no aparecer en el archivo de tabla de herramientas y cambiar a T0 dará como resultado que LinuxCNC piense que tiene el husillo vacío.

3.24.1.2. Cambiadores de herramientas aleatorios

El hardware de cambiador de herramientas aleatorio intercambia la herramienta en el husillo (si existe) con la herramienta solicitada a cambiar. Así, la ranura donde reside una herramienta cambia a medida que se intercambia dentro y fuera del husillo.

Un ejemplo de hardware de cambiador de herramientas aleatorio es un cambiador de herramientas de carrusel.

Cuando se configura para un cambiador de herramientas aleatorio, LinuxCNC intercambia el número de ranura de la herramienta antigua y la nueva en el archivo de tabla de herramientas cuando se

cargan las herramientas. Internamente, en el cambio de herramienta la información de la herramienta se **intercambia** entre la ranura de origen de la tabla de herramientas y la ranura 0 (que representa el husillo). Por tanto, después de un cambio de herramienta, la ranura 0 en la tabla de herramientas tendrá la información de la herramienta para la nueva herramienta y la ranura de la que la nueva herramienta vino tendrá la información de la herramienta que estaba en el husillo antes del cambio de herramienta, si la había.

---

nota
En LinuxCNC configurado para cambiador de herramientas aleatorio, la herramienta 0 (T0) **no** tiene significado especial. Se trata exactamente como cualquier otra herramienta en la tabla de herramientas. Es habitual utilizar T0 para representar ”sin herramienta” (es decir, una herramienta con TLO cero), de modo que el husillo se pueda vaciar convenientemente cuando sea necesario.

3.24.2. La tabla de herramientas

LinuxCNC realiza un seguimiento de las herramientas en un archivo llamado << sec:tool-table,tabla de herramientas>>. La tabla de herramientas registra la siguiente información para cada herramienta:

número de herramienta
Un entero que identifica de forma exclusiva esta herramienta. Los números de herramienta son manejados de manera diferente por LinuxCNC cuando se configuran cambiadores de herramientas no aleatorios o aleatorios:

- Cuando LinuxCNC está configurado para un cambiador de herramientas no aleatorio, el número debe ser positivo. T0 recibe un manejo especial y no está permitido que aparezca en la tabla de herramientas.
- Cuando LinuxCNC está configurado para un cambiador de herramientas aleatorio este número debe ser positivo o cero. T0 está permitido en la tabla de herramientas y generalmente se usa para representar ”ninguna herramienta”, es decir, ranura vacía.

número de ranura
Un entero que identifica la cavidad o ranura en el hardware del cambiador donde reside la herramienta. Los números de ranura se manejan de manera diferente por LinuxCNC cuando está configurado para cambiadores de herramientas aleatorios y no aleatorio:

- Cuando LinuxCNC está configurado para un cambiador de herramientas no aleatorio, el número de ranura en el archivo de herramientas puede ser cualquier número entero positivo (ranura 0 no está permitida). LinuxCNC compacta en silencio los números de ranura cuando carga el archivo de herramienta, por lo que puede haber una diferencia entre los números de ranura en el archivo de herramientas y los números internos de ranura utilizados por LinuxCNC con cambiador de herramienta no aleatorio.
- Cuando LinuxCNC está configurado para un cambiador de herramientas aleatorio, los números de ranura en el archivo de herramientas deben estar entre 0 y 1000, inclusivamente. Las ranuras 1-1000 están en el cambiador de herramientas; la ranura 0 es el husillo.

diámetro
Diámetro de la herramienta, en unidades de máquina.

offsets de longitud de herramienta
Offset de longitud de herramienta (también llamado TLO), hasta en 9 ejes, en unidades máquina. Los ejes que no tienen un TLO especificado, quedan con 0.

3.24.3. Códigos G que afectan a las herramientas

Los códigos G que usan o afectan la información de la herramienta son:

3.24.3.1. Txxx

Le dice al hardware del cambiador de herramientas que se prepare para cambiar a una determinada herramienta xxx.

Manejado por `Interp::convert_tool_select()`.

1. Se le pide a la máquina que se prepare para cambiar a la herramienta seleccionada llamando a la función `SELECT_TOOL()` de Canon con el número de la herramienta solicitada.
   a. (saicanon) No-op.
   b. (emccanon) Crea un mensaje `EMC_TOOL_PREPARE` con el número de ranura solicitada y lo envía a Task, que lo envía a IO. IO recibe el mensaje y le pide a HAL que prepare la ranura configurando `iocontrol.0.tool-prep-pocket`, `iocontrol.0.tool-prep-number`, y `iocontrol.0.tool`
   IO luego llama repetidamente a `read_tool_inputs()` para sondear el pin HAL `iocontrol.0.tool-` el cual señaliza a IO desde el hardware del cambiador de herramientas, a través de HAL, que la preparación de la herramienta solicitada está completa. Cuando ese pin se vuelve true, IO establece `emcioStatus.tool.pocketPrepped` con el número de ranura de la herramienta solicitada.
2. De vuelta a interp, se le asigna a `settings->selected_pocket` el índice de datos de la herramienta solicitada *xxx*.

---

nota
Los nombres heredados **selected_pocket** y **current_pocket** en realidad hacen referencia a un índice secuencial de datos de herramientas para elementos de herramientas cargados desde una tabla de herramientas ([EMCIO]TOOL_TABLE) o a través de una base de datos de herramientas ([EMCIO]DB_PROGRAM).

3.24.3.2. M6

Le dice al cambiador de herramientas que cambie a la herramienta seleccionada actualmente (seleccionada por el comando Txxx anterior).

Manejado por `Interp::convert_tool_change()`.

1. Se le pide a la máquina que cambie a la herramienta seleccionada llamando a la función `CHANGE_TOOL()` de Canon con `settings->selected_pocket` (un índice de datos de herramienta).
   a. (saicanon) Establece el `_active_slot` de sai en el número de ranura pasado. La información de la herramienta se copia de la ranura seleccionada de la tabla de herramientas (es decir, desde `_tools[_active_slot]` de sai) al husillo (también conocido como el `_tools[0]` de sai).
   b. (emccanon) Envía un mensaje `EMC_TOOL_LOAD` a Task, que lo envía a IO. IO establece `emcioStatus.t` al número de herramienta de la herramienta en el ranura identificada por `emcioStatus.tool.pock` (establecido por `Txxx` alias `SELECT_POCKET()`). Luego solicita que el hardware cambiador de herramientas realice un cambio de herramienta, estableciendo el pin HAL `iocontrol.0.tool-char` a True. Más tarde, `read_tool_inputs()` de IO detectará que el pin HAL `iocontrol.0.tool_change` se ha establecido en True, lo que indica que el cambiador de herramienta ha completado el cambio de herramienta. Cuando esto pasa, llama a `load_tool()` para actualizar el estado de la máquina.

i. `load_tool()` con un cambiador de herramientas no aleatorio, copia la información de la herramienta de la ranura seleccionada al husillo (ranura 0).
ii. `load_tool()` con un cambiador de herramientas aleatorio, intercambia la información de herramienta entre la ranura 0 (el husillo) y la ranura seleccionada, luego guarda la tabla de herramientas.

2. De vuelta en interp, a `settings->current_pocket` se le asigna el nuevo índice de datos de herramienta desde `settings->selected_pocket` (establecido por `Txxx`). Los parámetros numerados relevantes (#5400- #5413) son actualizados con la nueva información de herramienta de la ranura 0 (husillo).

3.24.3.3. G43/G43.1/G49

Aplica offset de longitud de herramienta. G43 usa el TLO de la herramienta cargada actualmente, o de una herramienta especificada si la palabra H se da en el bloque. G43.1 consigue el TLO de las palabras de eje en el bloque. G49 cancela el TLO (usa 0 para el offset de todos los ejes).

Manejado por `Interp::convert_tool_length_offset()`.

1. Comienza construyendo una `EmcPose` que contiene los offsets de 9 ejes a usar. Para `G43.1`, estos offsets de herramienta provienen de palabras de eje en el bloque actual. Para `G43` estos offsets provienen de la herramienta actual (la herramienta en la ranura 0), o de la herramienta especificada por la palabra H en el bloque. Para G49, los offsets son todos 0.
2. Los offsets se pasan a la función `USE_TOOL_LENGTH_OFFSET()` de Canon.
   a. (saicanon) Graba el TLO en `_tool_offset`.
   b. (emccanon) Crea un mensaje `EMC_TRAJ_SET_OFFSET` que contiene los offsets y lo envía a Task, que copia los offsets en `emcStatus->task.toolOffset` y los envía a Motion a través de un comando `EMCMOT_SET_OFFSET`. Motion copia los offsets a `emcmotStatus->tool_offset`, donde se usa para compensar movimientos futuros.
3. De vuelta en interp, los offsets se registran en `settings->tool_offset`. La ranura efectiva se registra en `settings->tool_offset_index`, aunque este valor nunca se usa.

3.24.3.4. G10 L1/L10/L11

Modifica la tabla de herramientas.

Manejado por `Interp::convert_setup_tool()`.

1. Selecciona el número de herramienta de la palabra P en el bloque y encuentra la ranura para esa herramienta:
   a. Con una configuración de cambiador de herramientas no aleatorio, este es siempre el número de ranura en el cambiador de herramientas (incluso cuando la herramienta está en el husillo).
   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. Averigua cuáles deberían ser los nuevos offsets.
3. La nueva información de la herramienta (diámetro, offsets, ángulos y orientación), junto con el número de herramienta y el número de ranura, se pasan a la llamada de Canon SET_TOOL_TABLE_ENTRY(

a. (saicanon) Copia la información de la nueva herramienta en el ranura especificada (en la tabla de herramientas interna de sai, + _tools +).
b. (emccanon) Crea un mensaje + EMC_TOOL_SET_OFFSET + con la información de la nueva herramienta y la envía a Task, quien la pasa a IO. IO actualiza la ranura especificada en su copia interna de la tabla de herramientas (`emcioStatus.tool.toolTable`), y si la herramienta especificada está cargada actualmente (se compara con `emcioStatus.tool.toolInSpindle`) entonces la información de la nueva herramienta se copia también en el ranura 0 (el husillo). (FIXME: eso es un buglet, solo debería copiarse en máquinas no aleatorias). Finalmente IO guarda la nueva tabla de herramientas.

4. De vuelta en interp, si la herramienta modificada está cargada actualmente en el husillo, y si la máquina es un cambiador de herramientas no aleatorio, entonces la nueva información de herramienta se copia de la ranura de inicio de la herramienta a la ranura 0 (el husillo) en la copia de interp de la tabla de herramientas, `settings->tool_table`. (Esta copia no es necesaria con un cambiador aleatorio, porque allí, las herramientas no tienen un ranura inicial y en su lugar, solo actualizamos directamente la herramienta en la ranura 0). Los parámetros numerados relevantes (#5400-#5413) se actualizan con la información de la herramienta en el husillo (copiando la información en interp de `settings->tool_table` a `settings->parameters`). (FIXME: esto es un buglet, los parámetros sólo deberían actualizarse si la herramienta actual fue la modificada).
5. Si la herramienta modificada está cargada actualmente en el husillo, y si la configuración es para un cambiador de herramientas no aleatorio, entonces el la nueva información de herramienta también se escribe en el ranura 0 de la tabla de herramientas, a través de una segunda llamada a SET_TOOL_TABLE_ENTRY(). (Esta segunda actualización a la tabla de herramientas no es necesaria con cambiadores aleatorios porque allí, las herramientas no tienen un ranura casa y en su lugar solo actualizamos directamente la herramienta en la ranura 0.)

3.24.3.5. M61

Establece el número de herramienta actual. Esto cambia la representación interna de LinuxCNC de cuál herramienta está en el husillo, sin mover realmente el cambiador de herramientas o intercambiar alguna herramienta.

Manejado por `Interp::convert_tool_change()`.

Canon: `CHANGE_TOOL_NUMBER ()`

A settings->current_pocket se le asigna el índice de datos de la herramienta actual especificado por el argumento palabra Q.

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

Habilita la compensación del radio del cortador (normalmente llamado *compensación de cortador*).

Manejado por `Interp::convert_cutter_compensation_on()`.

No hay llamada de Canon, la compensación del cortador ocurre en el intérprete. Usa la tabla de herramientas de la manera esperada: si se proporciona un número de herramienta palabra D busca el número de ranura del número de herramienta especificado en la tabla, y si no se proporciona ninguna palabra D utiliza la ranura 0 (el husillo).

3.24.3.7. G40

Cancela la compensación del radio de corte.

Manejado por `Interp::convert_cutter_compensation_off()`.

No hay llamada de Canon, la compensación del cortador ocurre en el intérprete. No se usa la tabla de herramientas.

3.24.4. Variables de estado interno

¡Esta no es una lista exhaustiva! La información de las herramienta esta esparcida por todo LinuxCNC.

3.24.4.1. IO

`emcioStatus` es de tipo `EMC_IO_STAT`

emcioStatus.tool.pocketPrepped
Cuando IO recibe la señal de HAL de que la preparación del cambiador de herramientas ha terminado (después de un comando `Txxx`), esta variable se establece en la ranura de la herramienta solicitada. Cuando IO recibe la señal de HAL que el cambio de herramienta en sí ha terminado (después de un comando `M6`) esta variable se restablece a -1.

emcioStatus.tool.toolInSpindle
Número de la herramienta instalada actualmente en el husillo. Exportada en el pin HAL `iocontrol.0.to` (s32).

emcioStatus.tool.toolTable[]
Una arreglo de estructuras `CANON_TOOL_TABLE`, `CANON_POCKETS_MAX` long. Cargada desde el archivo de la tabla de herramientas al inicio y mantenida ahí después. El índice 0 es el husillo, los índices 1- (CANON_POCKETS_MAX-1) son las ranuras en el cambiador de herramientas. Esta es una copia completa de la información de las herramientas, mantenida por separado de la `settings.tool_table` de Interp.

3.24.4.2. interp

`settings` es de tipo `settings`, definida como `struct setup_struct` en `src/emc/rs274ngc/interp_interna`

settings.selected_pocket
Índice de datos de la herramienta seleccionada más recientemente por `Txxx`.

settings.current_pocket
Índice original de datos de la herramienta actualmente en el husillo. En otras palabras: de cuál índice de datos de herramienta se cargó la herramienta actual en el husillo.

settings.tool_table[]
Un arreglo de información de herramientas. El índice en el arreglo es el ”número de cavidad” (también conocido como ”número de ranura”). La cavidad 0 es el husillo, las cavidades 1 hasta (CANON_POCKETS_MAX-1) son las cavidades del cambiador de herramientas.

settings.tool_offset_index
No usado. FIXME: Probablemente debería eliminarse.

settings.toolchange_flag
Interp establece esto en verdadero cuando llama a la función CHANGE_TOOL() de Canon. Se verifica en `Interp::convert_tool_length_offset()` para decidir qué índice de datos de herramienta usar para G43 (sin palabra H): `settings->current_pocket` si el cambio de herramienta aún está en progreso, índice de datos de herramienta 0 (el husillo) si se terminó el cambio de herramienta.

settings.random_toolchanger
Establecida desde la variable INI `[EMCIO]RANDOM_TOOLCHANGER` al inicio. Controla varias lógicas de manejo de la tabla de herramientas. (IO también lee esta variable INI y cambia su comportamiento con base en ella. Por ejemplo, al guardar la tabla de herramientas, para un cambiador aleatorio guarda la herramienta en el husillo (ranura 0), pero un cambiador no aleatorio guarda cada herramienta en su ”ranura casa”.)

settings.tool_offset
Esta es una variable `EmcPose`.

- Se utiliza para calcular la posición en varios lugares.
- Enviada a Motion a través del mensaje `EMCMOT_SET_OFFSET`. Todo lo que hace Motion con los offsets es exportarlos a los pines HAL `motion.0.tooloffset.[xyzabcuvw]`. FIXME: exportarlos desde algún lugar más cercano a la tabla de herramientas (io o interp, probablemente) y eliminar el mensaje EMCMOT_SET_OFFSET.

settings.pockets_max
Se usa de manera intercambiable con `CANON_POCKETS_MAX` (una constante #defined, establecida en 1000 a partir de abril de 2020). FIXME: Esta variable de configuración actualmente no es útil y probablemente debería eliminarse.

settings.tool_table
Esta es un arreglo de estructuras `CANON_TOOL_TABLE` (definidas en `src/emc/nml_intf/emctool.h`), con entradas `CANON_POCKETS_MAX`. Indexado por ”número de cavidad”, también conocido como ”número de ranura”. El índice 0 es el husillo, los índices 1 a (CANON_POCKETS_MAX-1) son las ranuras en el cambiador de herramientas. En un cambiador aleatorio, los números de ranura son significativos. En un cambiador no aleatorio, los ranuras no tienen significado; los números de ranura en el archivo de la tabla de herramientas se ignoran y a las herramientas se les asignan ranuras de `tool_table` secuencialmente.

settings.tool_change_at_g30 , settings.tool_change_quill_up , settings.tool_change_with_spindle_

Estas se establecen a partir de variables INI en la sección `[EMCIO]`, y determinan cómo se realizan los cambios de herramienta.

3.25. Recuento de articulaciones y ejes

3.25.1. En el búfer de estado

El búfer de estado es utilizado por Task y las UIs.

FIXME: `axis_mask` y `axes` sobre-especifican el número de ejes

`status.motion.traj.axis_mask`
Una máscara de bits con un ”1” para los ejes que están presentes y un ”0” para los ejes que no están presentes. X es el bit 0 con valor $2^0 = 1$ si esta prendido, Y es el bit 1 con valor $2^1 = 2$, Z es bit 2 con valor 4, etc. Por ejemplo, una máquina con ejes X y Z tendría una `axis_mask` de 0x5, una máquina XYZ tendría 0x7, y una máquina XYZB tendría una `axis_mask` de 0x17.

`status.motion.traj.axes` (eliminada)
Este valor fue eliminado en LinuxCNC versión 2.9. Usar en su lugar `axis_mask`.

`status.motion.traj.joints`
Un conteo del número de articulaciones que tiene la máquina. Un torno normal tiene 2 articulaciones, una manejando el eje X y otra manejando el eje Z. Una fresadora de pórtico XYYZ tiene 4 articulaciones, una manejando X, una manejando un lado de la Y, una manejando el otro lado de la Y, y una manejando Z. Una fresadora XYZA también tiene 4 articulaciones.

`status.motion.axis[EMCMOT_MAX_AXIS]`
Una arreglo de `EMCMOT_MAX_AXIS` estructuras de eje . `axis[n]` es válido si `(axis_mask & (1 << n))` es verdadero. Si `(axis_mask & (1 << n))` es falso, entonces `axis [n]` no existe en esta máquina y debe ser ignorado.

status.motion.joint[EMCMOT_MAX_JOINTS]
: Una arreglo de EMCMOT_MAX_JOINTS estructuras de articulación. De joint[0] a joint[articulaciones
son válidas, las otras no existen en esta máquina y deben ser ignoradas.

Las cosas ya no son así actualmente en la rama joints-axes, pero desviaciones de este diseño se consideran errores. Como ejemplo de tal error, ver el tratamiento a los ejes en src/emc/ini/initraj.cc: loadTraj (). Indudablemente hay más, y necesito tu ayuda para encontrarlos y arreglarlos.

3.25.2. En Motion

El componente en tiempo real controlador de movimiento Motion obtiene primero el número de articulaciones del parámetro de tiempo de carga num_joints. Esto determina cuántas articulaciones valen para crear pines HAL al inicio.

El número de articulaciones de Motion se puede cambiar en tiempo de ejecución utilizando el comando EMCMOT_SET_NUM_JOINTS desde Task.

El controlador Motion siempre funciona en los ejes EMCMOT_MAX_AXIS. Siempre crea nueve conjuntos de pines axis.*.*.

Capítulo 4

Mensajes NML

Lista de mensajes NML.
Para más detalles, ver `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. Otros

```
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

Estilo de codificación

Este capítulo describe el estilo de código fuente preferido por el equipo LinuxCNC.

5.1. No causar daños

Al realizar pequeñas ediciones en el código con un estilo diferente al descrito a continuación, observar el estilo de codificación local. Cambios rápidos de un estilo de codificación a otro disminuyen la legibilidad del código.

Nunca presentar código después de ejecutar ”indentado” en él. Los cambios de espacios en blanco introducidos por el indentado hacen que sea más difícil seguir el historial de revisión del archivo.

No usar un editor que realice cambios innecesarios en los espacios en blanco (p. ej., 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. Tabulaciones

Una tabulación siempre corresponde a 8 espacios. No escribir código que se muestre correctamente solo con una configuración de tabulación diferente.

5.3. Sangría

Usar 4 espacios por nivel de sangría. Combinando 8 espacios en un tabulador es aceptable pero no requerido.

5.4. Colocación de llaves

Ponga la llave de apertura al final de la línea y la llave de cierre al principio:

```
if (x) {
    // hacer algo apropiado
}
```

La llave de cierre está en una línea propia, excepto en los casos en que sea seguida por una continuación de la misma declaración, es decir, un *while* en una sentencia do o un *else* en una sentencia if, así:

```
do {
    // algo importante
} while (x > 0);
```

y

```
if (x == y) {
    // Haz una cosa
} else if (x < y) {
    // haz otra cosa
} else {
    // haz una tercer cosa
}
```

Esta colocación de llaves también minimiza el número de líneas vacías (o casi vacías), lo que permite una mayor cantidad de código o comentarios visible a la vez en una terminal de un tamaño fijo.

5.5. Nombrado

C es un lenguaje espartano, y así debería ser tu nombrado. A diferencia de Modula-2 y Pascal, los programadores de C no usan nombres pomposos como ThisVariableIsATemporaryCounter. Un programador de C llamaría a esa variable *tmp*, que es mucho más fácil de escribir y no más difícil de comprender.

Sin embargo, los nombres descriptivos para variables globales son imprescindibles. Llamar a una función global *foo* es un crimen.

Las variables GLOBALES (para usarlas solo si **realmente** se precisan) necesitan tener nombres descriptivos, al igual que las funciones globales. Si se tiene una función que cuenta el número de usuarios activos, debería llamarse *count_active_users()* o similar, **no** debe llamarse *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.

Los nombres de variables LOCALES deben ser cortos y concretos. Si tienes algún contador entero de bucles aleatorio, probablemente debería llamarse *i*. Llamarlo *loop_counter* no es productivo, si no hay posibilidad de ser mal entendido. Del mismo modo, *tmp* puede ser casi cualquier tipo de variable que se utiliza para mantener un valor temporal.

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. Funciones

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.

La longitud máxima de una función es inversamente proporcional a la complejidad y nivel de sangría de esa función. Por tanto, si tienes una función conceptualmente simple que es solo un largo (pero simple) switch-case, donde tienes que hacer muchas cosas pequeñas para muchos casos diferentes, está bien tener una función más larga.

Sin embargo, si tienes una función compleja y sospechas que un estudiante de primer año de secundaria no superdotado no podría siquiera entender de qué se trata la función, debes respetar los límites máximos más de cerca. Usar funciones de ayuda con nombres descriptivos (puedes pedirle al compilador que los incorpore si piensas que es crítico para el rendimiento, y probablemente hará un mejor trabajo que el tuyo).

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. Comentarios

Los comentarios son buenos, pero también existe el peligro de comentar en exceso. NUNCA intentes explicar CÓMO funciona tu código en un comentario; es mucho mejor escribir el código para que el **funcionamiento** sea obvio, y es un desperdicio de tiempo explicar un código mal escrito.

En general, que tus comentarios digan QUÉ hace tu código, no CÓMO. Un comentario en recuadro que describe la función, el valor de retorno y quién lo llama, colocado encima del cuerpo es bueno. Además, tratar de evitar poner comentarios dentro del cuerpo de una función; si la función es tan compleja que necesitas comentar partes por separado, probablemente deberías volver a leer la sección de funciones. Puedes hacer pequeños comentarios para anotar o advertir sobre algo particularmente inteligente (o feo), pero trata de evitar el exceso. En su lugar, coloca los comentarios al frente de la función, diciéndole a las personas lo que hace, y posiblemente POR QUÉ lo hace.

Si se utilizan comentarios /* Fix me */ en la línea, por favor, por favor, di por qué algo necesita ser arreglado. Cuando se ha realizado un cambio en la parte afectada del código, elimina el comentario o enmiéndalo para indicar que se ha realizado un cambio y necesita pruebas.

5.8. Scripts de Shell y Makefiles

No todos tienen las mismas herramientas y paquetes instalados. Algunas personas usan vi, otros emacs; algunos incluso evitan tener cualquiera de estos paquetes instalados, prefiriendo un editor de texto liviano como nano o el construido en Midnight Commander.

gawk versus mawk - Nuevamente, no todos tendrán gawk instalado, mawk es casi una décima parte del tamaño y, sin embargo, se ajusta al POSIX AWK estándar. Si se necesita algún comando específico de gawk oscuro que mawk no proporciona, el script se fallará para algunos usuarios. Lo mismo se aplicaría a mawk. En concreto, usar la invocación genérica awk en preferencia a gawk o mawk.

5.9. Convenciones de C++

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.

NO usar *Not* en un nombre booleano, solo conduce a confusión al hacer pruebas lógicas. p. ej. isNotOnLimit o is_not_on_limit son MALOS.

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. Convenciones de nomenclatura de métodos específicos

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.

Para los métodos que involucran atributos booleanos, se prefiere set y reset. Justificación: como arriba. p. ej. set_amp_enable reset_amp_fault

Los métodos intensivos en matemáticas deben usar el cálculo como prefijo. Razón fundamental: mostrar que es computacionalmente intensivo y acaparará la CPU. p. ej. compute_PID

Las abreviaturas en los nombres deben evitarse siempre que sea posible. La excepción es para nombres de variables locales. Justificación: claridad del código. p. ej. se prefiere pointer a ptr, compute a cmp, o compare a (de nuevo) 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.

Los declaraciones include de archivos de encabezado deben incluirse en la parte superior de un archivo fuente y no dispersas por todo el cuerpo. Deberían estar ordenadas y agrupadas por su posición jerárquica dentro del sistema con los archivos de bajo nivel incluidos primero. Las rutas de archivos include NUNCA deben ser absolutas; usar la bandera del compilador -I en su lugar para extender la ruta de búsqueda. Razón fundamental: Los encabezados pueden no estar en el mismo lugar en todos los sistemas.

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];`.

Del mismo modo, deben evitarse las sentencias ejecutables en condicionales, p. ej. `if (fd = open(nombre_a` es malo.

Deben evitarse las sentencias condicionales complejas - Introducir variables booleanas temporales en su lugar.

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

Nombres de archivo: las fuentes y los encabezados de C++ usan la extensión .cc y .hh. El uso de .c y .h están reservados para C. Los encabezados son para clases, métodos, y declaraciones de estructuras, no para código (a menos que las funciones estén declaradas en línea).

5.10. Estándares de codificación de Python

Utilizar el estilo PEP 8 para código de Python.

5.11. Estándares de codificación de componentes

En la parte de declaración de un archivo .comp, comenzar cada declaración en la primera columna. Insertar líneas en blanco adicionales cuando ayuden a agrupar elementos relacionados.

En la parte del código de un archivo .comp, seguir el estilo de codificación C normal.

Capítulo 6

Referencia de desarrollo de GUI

Este documento pretende ser una referencia de *mejores prácticas* para el desarrollo de pantallas de uso general.
Aunque es posible programar cualquier cosa que tan solo trabaje con LinuxCNC, utilizar marco de trabajo, lenguaje y requisitos de configuración en común facilita la transición entre pantallas y más desarrolladores para mantenerlas.
Dicho lo anterior, nada en este documento está escrito en piedra.

6.1. Lenguaje

Python es actualmente el lenguaje preferido para el código de pantallas de LinuxCNC.
Python tiene una barrera de entrada baja para que usuarios nuevos modifiquen las pantallas a sus necesidades.
Python tiene conjunto enriquecido de documentación, tutoriales y librerías al alcance.
Ya se usa y esta integrado en los requisitos de sistema de LinuxCNC.
Aunque se puede usar C o C+, limita severamente quien pueda mantener y desarrollarlo. Sería mejor extender Python con módulos C/C+ para cualquier función que lo requiera.

6.2. Localización de números de punto flotante en GUIs

Diferentes locales usan distintos separadores de decimales y de miles. Se deben evitar funciones de conversión cadena a número de punto flotante a una locale específica, ya que pueden dar resultados inesperados. (Por ejemplo la cadena de texto ”1.58” en de_DE será convertida a 158 por atof()). Se sugieren las siguientes pautas (basadas en evitar la ambigüedad mas que en la ”exactitud” en una locale específica) si se analiza sintácticamente un número de punto flotante para una cadena y viceversa:

- En el caso de que la entrada permita coma (,) o punto (.) como separador de decimales, pero rechace cualquier entrada que tenga más de uno de cualquiera. Se debería aceptar espacio mas no requerirlo como separador de miles.
- En el caso de despliegue usar punto (.) consistentemente o usar el formato de localización actual consistentemente. El énfasis aquí es en ”consistentemente”.

6.3. Configuración básica

Actualmente, la mayoría de las pantallas usan una combinación de archivo INI y entradas de archivo de preferencias para configurar sus funciones.

Se usan normalmente archivos de texto INI para las configuraciones de controlador de máquina comunes, mientras que los archivos de preferencias basados en texto se usan para propiedades más relacionadas con la GUI (como sonidos, tamaño, colores).
Pueden haber otros archivos usados para traducciones, estilizando y personalización de funciones. Estos son altamente dependientes del kit de herramientas de widgets subyacente.

6.3.1. INI [DISPLAY]

La sección **[DISPLAY]** del INI es para especificar configuraciones relacionadas con la pantalla.

6.3.1.1. Display

Lo más importante es especificar el nombre de la pantalla que el script LinuxCNC usará para cargar.
El programa de pantalla normalmente reconoce cambios como el de establecer pantalla completa.
El título es para la ventana y el ícono se usa en la iconización de la ventana.

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

6.3.1.2. Tiempo de ciclo

Si es configurable, así es como se establece el tiempo de ciclo de la visualización de GUI.
Es a menudo la tasa de actualización en vez del tiempo de espera entre actualizaciones.
Un valor común es 100 ms (0.1 s), aunque no es raro un rango entre 50 - 200 ms.

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

6.3.1.3. Rutas de archivo

Si estas funciones están disponibles en la pantalla, así deben especificarse las rutas a usar.
Estas deben referenciar desde el archivo INI actual, o permitir ~ para el directorio de inicio, o permitir el uso de rutas absolutas.

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

6.3.1.4. Incrementos de trote

Generalmente, se usan radio botones o cajas de selección para la selección de incrementos.
Los incrementos lineales pueden ser una mezcla de pulgadas y milímetros.
Los incrementos angulares se especifican en grados.
La palabra *continuo* se usa para especificar trote continuo y probablemente debería agregarse incluso si no se incluye en la línea INI.

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

6.3.1.5. Indicio de tipo de máquina

A menudo la pantalla necesita ajustarse con base en el tipo de máquina. Los tornos tienen controles diferentes y muestran los DROs de manera distinta. Las máquinas de espuma muestran el trazo de forma diferente.
La vieja forma de hacer esto era agregar interruptores LATHE = 1, FOAM = 1 etc

```
MACHINE_TYPE_HINT = LATHE
```

6.3.1.6. Ajuste manual

El ajuste manual permite al usuario ajustar la velocidad de alimentación o la velocidad del husillo al vuelo. Se acostumbra usar un control deslizante o una perilla.
Estas configuraciones son en porcentaje.

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

6.3.1.7. Velocidad de trote

La mayoría de las pantallas tiene controles deslizantes para ajustar el porcentaje de velocidad lineal y angular de trote.
Estas configuraciones deberían estar especificadas en unidades de máquina por minuto para lineal y grados por minuto para angular.
Default se refiere al porcentaje inicial cuando se carga por primera vez la pantalla.

```
DEFAULT_LINEAR_VELOCITY =
MIN_LINEAR_VELOCITY =
MAX_LINEAR_VELOCITY =

DEFAULT_ANGULAR_VELOCITY =
MIN_ANGULAR_VELOCITY =
MAX_ANGULAR_VELOCITY =
```

6.3.1.8. Control manual de husillo

Los controles manuales del husillo pueden ser botones, deslizantes, perillas o una combinación de ellos.
Se pueden establecer límites menores a lo que el controlador de la máquina puede utilizar configurando estas entradas.
Si la pantalla es capaz de manejar múltiples husillos, entonces debería aceptar entradas mayores a el 0 mostrado.

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

6.3.2. INI [MDI_COMMAND]

Algunas pantallas usan botones para ejecutar comandos NGC *Macro*.
Pueden especificarse como estos ejemplos compactos.
Comandos NGC separados por dos puntos se ejecutan hasta su término antes que el siguiente.
La coma opcional separa texto para el botón del código NGC.

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

6.3.3. INI [FILTER]

Esta sección permite configurar qué archivos se muestran en el selector de archivo y qué programas filtro preprocesarán su salida antes de enviarla a LinuxCNC.
Las extensiones siguen este patrón:
PROGRAM_EXTENSION = .extension,.extensión2[espacio]Descripción de las extensiones
Las definiciones de los programas filtro son como:
extensión_de_filtro = programa_a_ejecutar

```
[FILTER]
# Controla que programas se muestran en el administrador de archivos:
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]

La mayoría de las pantallas necesitarán algunos pines HAL. Es necesario conectarlos después que la pantallas los crea.

6.3.4.1. Archivo Hal postgui

Estos archivos se deben ejecutar uno tras de otro en orden, después de que se hayan creado todos los pines HAL de la GUI.

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

6.3.4.2. Archivos de comandos Hal postgui

Estos archivos deben ejecutarse uno tras de otro en orden, después de que se hayan ejecutado todos los archivos POSTGUI.

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

6.4. Configuración extendida

6.4.1. Incrustar elementos GUI

Una personalización común y muy útil es permitir a los usuarios construir paneles pequeños independientes que puedan incrustarse en la pantalla principal. Algunas pantallas permiten incrustar programas externos de terceros, otras solo paneles basados en los widgets nativos del kit de herramientas. Normalmente son incrustados en pestañas o widgets de panel lateral.
Así es como se describe el título opcional, el comando de carga y el nombre del widget de ubicación:

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

6.4.2. Mensajes de diálogo del usuario

Las ventanas de diálogo con el usuario se usan para mostrar emergentemente información (normalmente errores) que el usuario puede considerar importante.
Algunas permanecen a la vista hasta que se haya arreglado el problema, algunas requieren aceptación y otras una elección si/no.
Un pin HAL de E/S botaría el diálogo, el diálogo restablecería el pin E/S y establecería cualquier pin de salida de respuesta.

```
[DISPLAY]
MESSAGE_BOLDTEXT = Este es un mensaje informativo
MESSAGE_TEXT = Esto es baja prioridad
MESSAGE_DETAILS = pulse OK para limpiar
MESSAGE_TYPE = okdialog status
MESSAGE_PINNAME = bothtest
MESSAGE_ICON = INFO
```

Este estilo proporciona mensajes múltiples definidos por un número.
Este ejemplo muestra 3 mensajes posibles con base en un número de error de VFD.

```
[DISPLAY]
MULTIMESSAGE_ID = VFD

MULTIMESSAGE_VFD_NUMBER = 1
MULTIMESSAGE_VFD_TYPE = okdialog status
MULTIMESSAGE_VFD_TITLE = Error VFD: 1
MULTIMESSAGE_VFD_TEXT = Este es el texto más largo PARA EL MENSAJE NUMERO 1
MULTIMESSAGE_VFD_DETAILS = DETALLES para error VFD 1
MULTIMESSAGE_VFD_ICON = WARNING

MULTIMESSAGE_VFD_NUMBER = 2
MULTIMESSAGE_VFD_TYPE = nonedialog status
MULTIMESSAGE_VFD_TITLE = Error VFD: 2
MULTIMESSAGE_VFD_TEXT = Este es el texto más largo PARA EL MENSAJE NUMERO 2
MULTIMESSAGE_VFD_DETAILS = DETALLES para error VFD 2
MULTIMESSAGE_VFD_ICON = INFO

MULTIMESSAGE_VFD_NUMBER = 3
MULTIMESSAGE_VFD_TYPE = status
MULTIMESSAGE_VFD_TITLE = Error VFD: 3
MULTIMESSAGE_VFD_TEXT = Este es el texto más largo PARA EL MENSAJE NUMERO 2.
MULTIMESSAGE_VFD_DETAILS = Deberíamos hacer algo de este mensaje.
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, los del proyecto LinuxCNC, esperamos que compartas tus cambios, para que la comunidad pueda beneficiarse de tu trabajo. GitHub hace que compartir sea muy fácil: Después de pulir tus cambios y añadirlos a tu bifurcación GitHub, envíanos una solicitud de extracción (pull request).

7.2.1. Inicio rápido

Para los impacientes, intenten 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!, lo cual no te convierte en una mala persona, solo significa que debes leer todo este documento para averiguar cómo solucionar tus problemas. Especialmente la sección sobre Satisfacer dependencias de compilación.

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 construcción 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! Lee todo este documento, pero especialmente la sección en Configuración del entorno de prueba.

7.3. Plataformas compatibles

El proyecto LinuxCNC apunta a distribuciones modernas basadas en Debian, que incluyen Debian, Ubuntu y Mint. Continuamente probamos en las plataformas listadas en http://buildbot.linuxcnc.org.

LinuxCNC compila en la mayoría de las otras distribuciones de Linux, aunque la gestión de dependencias será más manual y menos automática. Siempre son bienvenidos parches para mejorar la portabilidad a nuevas plataformas .

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. Tendrás que compilar u obtener un kernel Xenomai por ti mismo.

Preempt-RT
: De https://rt.wiki.kernel.org. Un kernel de Linux con el parche Preempt-RT, que está disponible ocasionalmente en el archivo Debian en https://www.debian.org, y desde la ”máquina wayback” en https://snapshot.debian.org.

Tiempo diferido
: LinuxCNC también puede compilarse y ser ejecutado en plataformas en tiempo no-real, como una instalación normal de Debian o Ubuntu sin ningún kernel de tiempo real especial.
En este modo LinuxCNC no es muy útil para controlar máquinas herramientas, pero es útil para simular la ejecución de código G y para probar partes del sistema en tiempo no-real (como interfaces de usuario, algunos tipos de componentes y controladores de dispositivo).
Para hacer uso de las capacidades de tiempo real de LinuxCNC, ciertas partes de él necesitan ejecutarse con privilegios de root. Para habilitar dichas partes, ejecuta este comando extra después del `make` que compila LinuxCNC:

```
$ sudo make setuid
```

7.4. Modos de compilación

Hay dos modos de compilar LinuxCNC en una máquina: El modo amigable con el desarrollador ”ejecución en el sitio” (run in place o RIP) y el modo amigable con el usuario empaquetado Debian.

7.4.1. Compilación para ejecución en sitio

En una compilación Run-In-Place, los programas de LinuxCNC se compilan desde las fuentes y luego se ejecuta directamente desde el directorio de compilación. Nada queda instalado fuera del directorio de compilación. Es fácil y rápido, y adecuado para una iteración rápida de cambios. El conjunto de pruebas de LinuxCNC solo se corre en una compilación Run-In-Place. La mayoría de los desarrolladores compilan primordialmente con este modo.

Una compilación para ejecución en sitio sigue los pasos de la sección Inicio rápido en la parte superior de este documento, posiblemente con argumentos diferentes 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 directorio-codigo-fuente-linuxcnc/src
$ ./configure --help
```

Los argumentos más utilizados son:

`--with-realtime=uspace`
: Compilar para cualquier plataforma en tiempo real, o para tiempo diferido. Los ejecutables LinuxCNC resultantes se ejecutarán tanto en un kernel de Linux con parches Preempt-RT (que proporcionan control de la máquina en tiempo real) como en un kernel de Linux original (sin parches) (que proporciona simulación de código G pero sin control de máquina en tiempo real).

  Si los archivos de desarrollo están instalados para Xenomai (típicamente del paquete libxenomai-dev) o RTAI (típicamente desde un paquete con un nombre que comienza por "rtai-modules"), también estará habilitado el soporte para estos kernels en tiempo real.

`--with-realtime=/usr/realtime-$VERSION`
: Compilación para la plataforma RTAI en tiempo real utilizando el antiguo modelo "kernel realtime". Esto requiere tener un kernel RTAI y los módulos RTAI instalados en `/usr/realtime-$VERSION`. Los ejecutables LinuxCNC resultantes solo se ejecutarán en el kernel RTAI especificado. A partir de LinuxCNC 2.7, esto produce el mejor rendimiento en tiempo real.

`--enable-build-documentation`
: Construir la documentación además de los ejecutables. Esta opción aumenta significativamente el tiempo requerido para la compilación, ya que construir los documentos consumen bastante tiempo. Si no trabajas activamente en la documentación querrás omitir este argumento.

`--disable-build-documentation-translation`
: Deshabilitar la traducción de documentación para todos los lenguajes disponibles. La construcción de la documentación traducida toma una enorme cantidad de tiempo, así que se recomienda omitirla si no es realmente necesaria.

7.4.1.2. Argumentos `make`

El comando `make` admite dos argumentos opcionales útiles.

Compilación paralela
: `make` admite un argumento opcional `-j`N (donde N es un número). Esto permite compilación en paralelo con N procesos simultáneos, que puede acelerar significativamente tu construcción.

  Un valor útil para N es la cantidad de CPUs en tu sistema de compilación.

Puedes averiguar el número de CPUs ejecutando `nproc`.

Construir solo un objetivo específico
: Si quieres construir solo una parte específica de LinuxCNC, puedes nombrar lo que quieres construir en la línea de comando `make`. Por ejemplo, si estás trabajando en un componente llamado `froboz`, puedes construir su ejecutable con los comandos:

```
> cd directorio-codigo-fuente-linuxcnc/src
> make ../bin/froboz
```

7.4.2. Construyendo paquetes Debian

Al construir paquetes Debian, los programas LinuxCNC se compilan desde el código fuente y luego se almacenan en un paquete Debian, completado con información de dependencias. Este proceso incluye de forma predeterminada la construcción de la documentación, lo cual toma su tiempo debido a toda la E/S para muchos idiomas, pero no puede ser omitida. Entonces LinuxCNC es instalado como parte de esos paquetes en la misma máquina o en cualquier máquina con la misma arquitectura a la que se copien los archivos .deb. LinuxCNC no puede ejecutarse hasta que se instalen los paquetes Debian en una máquina destino, y hasta entonces estarán disponibles los ejecutables en /usr/bin y /usr/lib, tal y como otro software del sistema.

Este modo de compilación es principalmente útil cuando se empaqueta el software para entrega a usuarios finales, o para construir el software para una máquina que no tiene instalado el entorno de compilación, o que no tiene acceso a Internet.

Para los impacientes, intenten esto:

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

La construcción de paquetes Debian se hace con la herramienta `dpkg-buildpackage` que viene en el paquete `dpkg-dev`. Para ejecutarla se tiene una serie de pre-requisitos que se detallan a continuación: * se debe instalar la infraestructura general de construcción, p. ej. compiladores, etc. * deben estar instaladas las dependencias de compilación, p. ej. los archivos de encabezados para las librerías de código externo utilizadas, como se describe en la sección Satisfacer dependencias de compilación. * el archivo que describe el paquete debe estar completado y en el directorio debian

Las herramientas de compilación han sido reunidas en un paquete virtual nombrado `build-essential`. Para instalarlo ejecute:

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

Una vez que se cumplen esos requisitos previos, la construcción de los paquetes Debian consiste en dos pasos.

El primer paso es generar los scripts y metadatos del paquete Debian desde el repositorio git ejecutando esto:

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

---

nota
¡El script `debian/configure` es diferente del script `src/configure`!
El script `debian/configure` acepta argumentos diferentes dependiendo de la plataforma en/para la que se está compilando; ver la sección argumentos debian/configure. Esta predeterminado para correr LinuxCNC en espacio de usuario (”uspace”), esperando que el kernel preempt_rt minimice latencias.

Una vez que los scripts del paquete Debian y los metadatos estén configurados, construir el paquete ejecutando `dpkg-buildpackage`:

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

nota
Se necesita que `dpkg-buildpackage` se ejecute desde la raíz del árbol de fuentes, el cual podrías haber nombrado `directorio-codigo-fuente-linuxcnc`, y **no** desde `directorio-codigo-fuente-linuxcnc/debian`.
`dpkg-buildpackage` toma un argumento opcional ``-j`N` (donde *N* es un número). Esto habilita la ejecución de múltiples tareas simultáneamente.

7.4.2.1. Argumentos `debian/configure` de LinuxCNC

El árbol de fuentes de LinuxCNC tiene un directorio debian con toda la información de cómo armar el paquete Debian, pero algunos archivos esenciales solo se distribuyen como plantillas. La secuencia de comandos `debian/configure` prepara esas instrucciones de armado para las utilerías de empaquetamiento Debian habituales y por lo tanto deben ser ejecutadas antes que `dpkg-checkbuilddeps` o `dpkg-buildpackage`.

La secuencia de comandos `debian/configure` admite un solo argumento que especifica la subyacente plataforma de tiempo real o de tiempo no-real para la que se compila. Los valores normales para este argumento son:

`no-docs`
: Saltar construcción de documentación.

`uspace`
: Configura el paquete Debian para Preempt-RT en tiempo real o para no tiempo real (estos dos son compatibles).

`noauto , rtai , xenomai`
: Normalmente, se detectan automáticamente las listas de RTOS para uspace en tiempo real soportados. Sin embargo, si lo deseas, puedes especificar uno o más de estos RTOS después de `uspace` para habilitar el soporte para estos RTOS. Para deshabilitar la autodetección, especificar `noauto`.

  Si solo quieres el tradicional ”módulo de kernel” RTAI en tiempo real, usa `-r` o `$KERNEL_VERSION` en su lugar.

`rtai=<nombre del paquete>`
: Si el paquete de desarrollo para RTAI, lxrt, no comienza con ”rtai-modules”, o si el primer paquete de este tipo aparece en la búsqueda de apt-cache no es el deseado, especifique explícitamente el nombre del paquete.

`-r`
: Configura el paquete Debian para el kernel RTAI actualmente en ejecución. ¡Debes ejecutar un kernel RTAI en tu máquina de compilación para que esto funcione!

`$KERNEL_VERSION`
: Configura el paquete Debian para la versión de kernel RTAI especificada (por ejemplo, ”3.4.9-rtai-686-pae”). Los encabezados del kernel del paquete Debian coincidente deben estar instalados en su máquina de compilación (p. ej. ”linux-headers-3.4.9-rtai-686-pae”). Tenga en cuenta que puede *construir* LinuxCNC en esta configuración, pero si no está ejecutando el kernel RTAI coincidente, no podrá *ejecutar* LinuxCNC, incluyendo el conjunto de pruebas.

7.4.2.2. Satisfacer dependencias de compilación

En las plataformas basadas en Debian, proporcionamos metadatos de empaquetado que saben qué paquetes de software externos deben instalarse para compilar LinuxCNC. Esas son referidas como *dependencias de compilación* de LinuxCNC, p. ej. aquellos paquetes que deben estar disponibles de tal manera que

- la compilación se exitosa y
- la compilación pueda compilarse reproduciblemente.

Puedes usar estos metadatos para enlistar fácilmente los paquetes requeridos faltantes en tu sistema de compilación. Primero, ve al árbol de fuentes LinuxCNC e inicia su auto-configuración predeterminada, si aún no se ha realizado:

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

Esto preparará el archivo debian/control que contiene la lista de paquetes Debian a crear con la dependencias en tiempo de ejecución para esos paquetes y, para nuestra causa, las dependencias de compilación para aquellos paquetes a ser creados.

La manera más directa de instalar todas esas dependencias de compilación es simplemente ejecutar (desde el mismo directorio):

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

la cual instalará todas las dependencias requeridas, aún no instaladas, pero disponibles. El . es parte de la línea de comandos, p. ej. una instrucción para obtener a mano las dependencias para el árbol de fuentes, no para las dependencias de otro paquete. Esto completa la instalación de dependencias de compilación.

El resto de esta sección describe un enfoque semiautomático. La lista de dependencias en debian/control es larga y es tedioso comparar el estado actual de los paquetes ya instalados. Los sistemas Debian proporcionan un programa llamado `dpkg-checkbuilddeps` que analiza los metadatos del paquete y compara los paquetes enumerados como dependencias de compilación contra la lista de paquetes instalados, y te dice lo que falta.

Primero, instala el programa `dpkg-checkbuilddeps` ejecutando:

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

Esto genera el archivo `debian/control` en un formato legible para el usuario yaml, el cual enlista las dependencias de compilación cercanas a lo más alto. Puedes usar estos metadatos para enlistar los paquetes requeridos faltantes en tu sistema de compilación. Puedes decidir inspeccionar manualmente esos archivos si tienes un buen entendimiento de lo que ya esta instalado.

Alternativamente, los sistemas Debian proporcionan un programa llamado `dpkg-checkbuilddeps` que analiza los metadatos del paquete y compara los paquetes enlistados como dependencias de compilación contra la lista de paquetes instalados, y te dice qué falta. Además, `dpkg-buildpackage` te informará de lo que falta, y estará bien. Sin embargo, reportará las dependencias de compilación faltantes solo después de que se hayan aplicado automáticamente los parches del directorio `debian/patches` (si hay alguno). Si eres nuevo en la gestión de versiones en Linux y git, un inicio desde cero puede ser preferible para evitar complicaciones.

Se le puede pedir al programa `dpkg-checkbuilddeps` (también del paquete dpkg-dev que se instala como parte de build-essential-dependencies) que haga su trabajo (ten en cuenta que necesita ser ejecutado desde el directorio `directorio-codigo-fuente-linuxcnc`, **no** desde `directorio-codigo-fuente-linux`

```
$ dpkg-checkbuilddeps
```

Esto emitirá una lista de paquetes necesarios para compilar LinuxCNC en tu sistema, pero que aún no están instalados. Ahora puedes instalar las dependencias de compilación faltantes de forma

manual
: Se instalan todas con `sudo apt-get install`, seguido de los nombres de los paquetes. Puedes volver a ejecutar `dpkg-checkbuilddeps` cuando quieras para enlistar los paquetes faltantes, lo cual no tiene efecto en el árbol de fuentes.

automatizada
Ejecute `sudo apt build-dep .`.

En caso de duda sobre lo que trae un paquete en particular de una dependencia de compilación, revisa la descripción del paquete con ``apt-cache show`` *nombre-de-paquete*.

7.4.2.3. Opciones para `dpkg-buildpackage`

Para un armado de paquete típico de Debian, ejecutarías dpkg-buildpackage sin argumentos. Como si señaló anteriormente, el comando tiene dos opciones extras a pasarle. Como en todas las buenas herramientas de Linux, la página del manual tiene todos los detalles con `man dpkg-buildpackage`.

`-uc`
No firmar digitalmente los binarios resultantes. Querrás firmar tus paquetes con una llave GPG tuya solo si quieres distribuirlo a otros. El no tener la opción establecida y fallar la firma del paquete no afectará el archivo .deb.

`-b`
Solo compila los paquetes dependientes de arquitectura (como los binarios de `linuxcnc` y GUIs). Esta es muy útil para evitar compilar lo que es dependiente de hardware. Para LinuxCNC, es la documentación, la cual esta disponible en línea de todos modos.

Si llegaras a tener dificultades con la compilación, revisa el foro de LinuxCNC en línea.

Actualmente esta surgiendo el soporte de la variable de ambiente DEB_BUILD_OPTIONS. Se le puede asignar

`nodocs`
para omitir la construcción de la documentación, preferentemente usa en su lugar la bandera `-B` de dpkg-buildpackage.

`nocheck`
para omitir las auto-pruebas del proceso de compilación de LinuxCNC. Esto ahorra algo de tiempo y reduce la demanda de algunos paquetes de software que pudieran no estar disponibles para tu sistema. Por ejemplo, el xvfb en particular. No deberías establecer esta opción para tener más confianza en el desempeño esperado de tu compilación a menos que caigas en dificultades meramente técnicas con las dependencias de software específicas para pruebas.

Una variable de ambiente puede ser establecida junto con la ejecución del comando, p. ej.

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

combinará todas las opciones presentadas en esta sección.

7.4.2.4. Instalando paquetes Debian auto-compilados

Un paquete Debian puede ser reconocido por su extensión .deb. La herramienta que lo instala es `dpkg` y es parte de toda instalación Debian. Los archivos .deb creados por `dpkg-buildpackage` se encuentran en el directorio arriba del directorio-codigo-fuente-linuxcnc, p. ej. en `...` Para ver qué archivos vienen en un paquete ejecuta

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

La versión de LinuxCNC será parte del nombre del archivo, la que deberá estar en el lugar del asterisco. Pueden haber demasiados archivos enlistados como para que quepan en tu pantalla. Si no puedes hacer desplazamiento hacia arriba en tu terminal, agrega `| more` al comando para que su salida pase por lo que se llama un ”paginador”, del cual sales presionando la tecla ”q”.

Para instalar los paquetes ejecuta

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

7.5. Configuración del entorno

Esta sección describe los pasos especiales necesarios para configurar una máquina para ejecutar los programas LinuxCNC, incluidas las pruebas.

7.5.1. Aumentar el límite de memoria bloqueada

LinuxCNC intenta mejorar su latencia en tiempo real bloqueando la memoria que utiliza en la RAM. Hace esto para evitar que el sistema operativo intercambie LinuxCNC al disco, lo que tendría malos efectos sobre la latencia. Normalmente, bloquear memoria en RAM es mal visto, y el sistema operativo pone un límite estricto sobre cuánta memoria se le permite bloquear a un usuario.

Cuando se utiliza la plataforma de tiempo real Preempt-RT, LinuxCNC se ejecuta con suficiente privilegio para aumentar su límite de bloqueo de memoria. Cuando se usa la plataforma RTAI en tiempo real, no tiene suficientes privilegios, y el usuario debe elevar el límite de bloqueo de memoria.

Si LinuxCNC muestra el siguiente mensaje al inicio, el problema es el límite de memoria bloqueada configurado en tu sistema:

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

Para solucionar este problema, agrega un archivo llamado `/etc/security/limits.d/linuxcnc.conf` (como root) con tu editor de texto favorito (p. ej., `sudo gedit/etc/security/limits.d/linuxcnc.conf`). El archivo debe contener la línea siguiente:

```
* - memlock 20480
```

Cierra y vuelve a iniciar sesión para que los cambios surtan efecto. Verifica que el límite de bloqueo de memoria se aumentó con el comando siguiente:

```
$ ulimit -l
```

7.6. Compilación en Gentoo

Es posible compilar en Gentoo, pero sin soporte. Asegúrate que ejecutas un perfil de escritorio. Este proyecto usa el conjunto de Widgets Tk, asciidoc y tiene algunas otras dependencias; deben ser instaladas como 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
```

Puedes cambiarte de vuelta a un usuario normal para la mayoría del resto de la instalación. Como ese usuario, crea un ambiente virtual para pip y luego instala los paquetes pip:

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

Entonces puedes continuar normalmente:

```
(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
```

No hay necesidad de ejecutar ”make suid”, solo asegúrate que tu usuario está en el grupo ”dialout”. Para arrancar linuxcnc, debes estar en el Ambiente Virtual Python y configurar el ambiente linuxcnc:

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

7.7. Opciones para ver el repositorio de git

Las instrucciones de Inicio rápido en la parte superior de este documento indican hacer un clon de nuestro repositorio en http://github.com/LinuxCNC/linuxcnc.git. Esta es la manera más rápida y fácil de empezar. Sin embargo, hay otras opciones a considerar.

7.7.1. Bifurcación en Github

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. Puedes crear fácilmente (y sin costo) una bifurcación (una segunda instancia con una copia que tú controlas) del repositorio de git de LinuxCNC en GitHub. Entonces podrás usar esa bifurcación para rastrear y publicar tus cambios, recibir comentarios a tus cambios y aceptar parches de la comunidad. .

Después de crear tu propia bifurcación en GitHub de LinuxCNC, clónala en tu computadora de desarrollo y procede con tu hackeo como de costumbre.

Nosotros, el proyecto LinuxCNC, esperamos que compartas tus cambios, para que la comunidad pueda beneficiarse de tu trabajo. GitHub hace que compartir sea muy fácil; después de pulir tus cambios y añadirlos a tu bifurcación GitHub, envíanos una solicitud de extracción.

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

Contribuir a LinuxCNC

9.1. Introducción

Este documento contiene información para desarrolladores sobre la infraestructura de LinuxCNC, y describe las mejores prácticas para contribuir con código y actualizaciones de documentación para el proyecto LinuxCNC.

En este documento, ”fuente” significa tanto el código fuente de programas y bibliotecas, como el texto fuente para la documentación.

9.2. Comunicación entre desarrolladores de LinuxCNC

Las dos formas principales en que los desarrolladores de proyectos se comunican entre sí son:

- Via IRC, en #linuxcnc-devel en Libera.chat.
- Por correo electrónico, en la lista de correo de los desarrolladores

9.3. El proyecto LinuxCNC en Source Forge

Utilizamos Source Forge para las listas de correo.

9.4. El sistema de control de revisiones Git

Toda la fuente de LinuxCNC se mantiene en el sistema de control de revisiones Git.

9.4.1. Repositorio Git oficial de LinuxCNC

El repositorio oficial git de LinuxCNC está en https://github.com/linuxcnc/linuxcnc/

Cualquiera puede obtener una copia de solo lectura del árbol fuente de LinuxCNC a través de git:

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

Si eres un desarrollador con acceso push, entonces sigue las instrucciones de github para configurar un repositorio donde puedas hacer push.

Ten en cuenta que el comando clone coloca el repositorio local de LinuxCNC en un directorio llamado `linuxcnc-dev`, en lugar del predeterminado `linuxcnc`. Esto se debe a que el software LinuxCNC por defecto espera configuraciones y programas en código G en un directorio llamado $HOME/linuxcnc`, y tener el repositorio git ahí mismo causa confusiones.

Issues y pull requests (abreviado PRs) son bienvenidos en GitHub: . https:///github.com/LinuxCNC/-linuxcnc/issues . https://github.com/LinuxCNC/linuxcnc/pulls

9.4.2. Uso de Git en el proyecto LinuxCNC

Utilizamos los flujos de trabajo git ”merging upwards” y ”topic branches” descritos aquí:

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

Tenemos una rama de desarrollo llamada `master`, y una o más ramas estables con nombres como `2.6` o `2.7` que indican el número de versión de los lanzamientos que hacemos.

Las correcciones de errores van en la rama estable aplicable más antigua, y esa rama se fusiona con la siguiente rama estable más nueva, y así sucesivamente hasta `master`. El committer o confirmador de la corrección de error, puede hacer las fusiones por sí mismo, o dejarlas a otra persona.

Las nuevas características generalmente van en la rama `master`, pero algunos tipos de características (específicamente controladores de dispositivo y documentación bien aislados) pueden (a discreción de los gerentes de la rama estable) entrar en una rama estable y fusionarse como lo hacen las correcciones de errores.

9.4.3. Tutoriales de git

Hay muchos excelentes tutoriales gratuitos de git en Internet.

El primer lugar para buscar es probablemente la página de manual ”gittutorial”. Se puede acceder a esta página de manual ejecutando ”man gittutorial” en una terminal (si tiene instaladas las páginas de manual de git). El gittutorial y su consiguiente documentación también están disponibles en línea aquí:

- tutorial de git: https://www.kernel.org/pub/software/scm/git/docs/gittutorial.html
- tutorial git 2: https://www.kernel.org/pub/software/scm/git/docs/gittutorial-2.html
- Git del diario con unos 20 comandos: https://www.kernel.org/pub/software/scm/git/docs/giteveryday.html
- Manual del usuario de Git: https://www.kernel.org/pub/software/scm/git/docs/user-manual.html

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

Otro tutorial en línea que se ha recomendado es ”Git for the Lazy”: http://wiki.spheredev.org/Git_for_the_lazy

9.5. Descripción general del proceso

La descripción general de alto nivel de cómo contribuir cambios a la fuente va así:

- Comunícate con los desarrolladores del proyecto y dinos qué andas hackeando. Explica lo que estas haciendo y por qué.

- Clona el repositorio git.
- Haz tus cambios en una rama local.
- Agregar documentación y escribir pruebas es una parte importante de agregar una nueva característica. De lo contrario, otros no sabrán cómo usar tu característica, y si otros cambios corrompen tu característica, pueden pasar desapercibidos sin una prueba.
- Comparte tus cambios con los otros desarrolladores del proyecto de una de estas maneras:
  - Haz push de tu rama a github y crea una pull request de github hacia https://github.com/linuxcnc/-linuxcnc (esto requiere una cuenta github), o
  - Haz push de tu rama a un repositorio git visible públicamente (como github, o tu propio servidor de acceso público, etc.) y comparte esa ubicación en la lista de correo de emc-developers, o
  - Envía tus confirmaciones (commits) por correo electrónico a la lista de correo de LinuxCNC-developers (<emc-developers@lists.sourceforge.net>) (usa `git format-patch` para crear parches).
- Defiende tu parche:
  - Explica qué problema aborda y por qué debería incluirse en LinuxCNC.
  - Sé receptivo a las preguntas y comentarios de la comunidad de desarrolladores.
  - No es raro que un parche pase por varias revisiones antes de ser aceptado.

9.6. Configuración de git

Para ser considerada la inclusión en las fuentes de LinuxCNC, los commits deben tener campos de autor correctos que identifiquen al autor de la confirmación. Una buena manera de garantizar esto es estableciendo tu configuración global de git:

```
git config --global user.name ”Tu nombre completo”
git config --global user.email ”tucorreo@example.com”
```

Usa tu nombre real (no un identificador) y una dirección de correo electrónico clara (no ofuscada).

9.7. Uso efectivo de git

9.7.1. Contenido de confirmaciones

Mantén tus confirmaciones pequeñas y concretas. Cada commit debe aportar un cambio lógico al repositorio.

9.7.2. Escribir buenos mensajes de confirmación

Mantén los mensajes de confirmación alrededor de 72 columnas de ancho (de modo que en un tamaño predeterminado de ventana de terminal, no se partan cuando se muestren con `git log`).

Usa la primera línea como un resumen de la intención del cambio (casi como la línea de asunto de un correo electrónico). Seguida por una línea en blanco, y luego un mensaje más largo explicando el cambio. Ejemplo:

9.7.3. Confirmar a la rama adecuada

Las correcciones de errores deben ir en la rama aplicable más antigua. Las nuevas funciones deberían ir a la rama master. Si no estás seguro a dónde pertenece un cambio, pregunta en el irc o en la lista de correo.

9.7.4. Usar confirmaciones múltiples para organizar cambios

Cuando sea apropiado, organiza tus cambios en una rama (una serie de confirmaciones) donde cada commit es un paso lógico hacia su objetivo máximo. Por ejemplo, primero factoriza un código complejo en una nueva función. Luego, en una segunda confirmación, corrige algún error subyacente. Después, en un tercer commit agrega una nueva característica que se hizo más fácil por la refactorización y que no hubiera funcionado sin arreglar aquel error.

Esto es útil para los revisores, porque es más fácil ver que el paso ”factorizar el código en una nueva función” era correcto, sin otras ediciones mezcladas; es más fácil ver que el error se corrige cuando el cambio que lo arregla es independiente de la nueva característica; y así sucesivamente.

9.7.5. Seguir el estilo del código circundante

Haz un esfuerzo por seguir el estilo de sangría predominante en el código circundante. En particular, los cambios en los espacios en blanco hacen que sea más difícil para otros desarrolladores rastrear cambios a lo largo del tiempo. Cuando se debe reformatear código, hazlo en una confirmación separada de cualquier cambio semántico.

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. Simplificar historial complicado antes de compartir con otros desarrolladores

Con git, es posible grabar cada edición y comienzo en falso en una confirmación separada. Esta es una forma muy conveniente de crear puntos de control durante el desarrollo, pero a menudo no querrás compartir estos comienzos en falso con otros.

Git proporciona dos formas principales de limpiar el historial, las cuales se pueden hacer libremente antes de compartir el cambio:

`git commit --amend` te permite hacer cambios adicionales a tu última confirmación, modificando opcionalmente también el mensaje de confirmación. Utiliza esto si te das cuenta de inmediato que dejaste algo fuera de la confirmación, o si cometiste un error tipográfico en el mensaje.

`git rebase --interactive` upstream-branch te permite volver a través de cada confirmación realizada desde que bifurcaste tu rama de características desde la rama superior, posiblemente editando, descartando o comprimiendo (combinando) commits con otros. Rebase también se puede usar para dividir confirmaciones individuales en múltiples confirmaciones nuevas.

9.7.8. Asegurarse que cada confirmación compile

Si tu cambio consta de varios parches, `git rebase -i` puede usarse para reordenarlos en una secuencia de confirmaciones que exponga más claramente los pasos de tu trabajo. Una consecuencia potencial de reordenar parches es que podrían aparecer dependencias incorrectas, por ejemplo, introducir el uso de una variable y, en un parche posterior, la declaración de esa variable.

Aunque la rama HEAD compile, no todas las confirmaciones podrían compilar en tal caso. Eso rompe `git bisect`, algo que alguien podría usar más tarde para encontrar la confirmación que introdujo un error. Así que más allá de asegurarte que tu rama compile, es importante asegurar que cada confirmación individual también compile.

Hay una forma automática de verificar que cada confirmación en una rama sea compilable - ver https://dustin.sallings.org/2010/03/28/git-test-sequence.html y el código en https://github.com/dustin/-bindir/blob/master/git-test-sequence. Usarlo de la siguiente manera (en este caso probando cada confirmación desde origin/master a HEAD, incluida la ejecución de pruebas de regresión):

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

Esto informará que todo está bien (*All is well*) o que tronó (*Broke on <confirmación>*)

9.7.9. Renombrar archivos

Utiliza la capacidad de cambiar el nombre de los archivos con mucho cuidado. Así como el indentar archivos individuales, los cambios de nombre hacen más difícil dar seguimiento a cambios en el tiempo. Como mínimo deberías buscar consenso, en IRC o en la lista de correo, de que el cambio de nombre es una mejora.

9.7.10. Preferir "rebase"

Utiliza `git pull --rebase` en lugar de un puro `git pull` para mantener un buen historial lineal. Cuando rebasas, siempre retienes tu trabajo como revisiones por delante de origin/master, por lo que puedes hacer cosas como `git format-patch` para compartirlo con otros sin hacer push al repositorio central.

9.8. Traducciones

El proyecto LinuxCNC usa `gettext` para traducir el software a varios idiomas. ¡Damos la bienvenida a contribuciones y ayuda en esta área!. Mejorar y extender las traducciones es fácil: no necesitas saber de programación ni necesitas instalar algún programa especial de traducción u otro 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/

La documentación sobre cómo usar Weblate está aquí: https://docs.weblate.org/en/latest/user/basic.html

9.9. Otras formas de contribuir

Hay muchas formas de contribuir a LinuxCNC que no se abordan en este documento. Estas formas incluyen:

- Responder preguntas en el foro, listas de correo y en IRC
- Informar errores en el seguidor de errores, foro, listas de correo o en IRC
- Ayudando a probar características experimentales

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

Sección legal

Las traducciones de este archivo proporcionadas en el árbol de código fuente no son legalmente vinculantes.

11.1. Términos de derechos de autor

11.2. Licencia GNU de Documentation Libre

Licencia de Documentación Libre GNU Versión 1.1, Marzo 2000

0. PREÁMBULO

El propósito de esta licencia es hacer a un manual, libro de texto u otro documento escrito ”libre” en el sentido de libertad: garantizar a todos la libertad efectiva para copiarlo y redistribuirlo, modificándolo o no, ya sea comercialmente o no. Secundariamente, La Licencia preserva para el autor y el editor un forma de obtener crédito por su obra, mientras no se le considere responsable por las modificaciones hechas por otros.

Esta Licencia es un tipo de ”izquierdo de copia”, lo cual significa que las obras derivadas del documento deben por si mismos ser libres en el mismo sentido. Complementa la Licencia GNU de Público General, la cual es una licencia de izquierdo de copia diseñada para software libre.

Hemos diseñado esta Licencia con el fin de usarla para manuales de software libre, por que el software libre necesita documentación libre: un programa libre debe venir con manuales que proporcionen las mismas libertades de las que goza el software. Pero esta Licencia no esta limitada a manuales de software; puede ser usada para cualquier obra en texto, independientemente del tema o de si es publicado como libro impreso. Recomendamos esta Licencia principalmente para obras cuyo propósito es la instrucción o la referencia.

1. APLICABILIDAD Y DEFINICIONES

Esta Licencia aplica a cualquier manual u otra obra que contenga un aviso del titular de los derechos de autor manifestando que puede ser distribuido bajo los términos de esta Licencia. El ”Documento”, a continuación, se refiere a cualquier manual u obra del tipo. Cualquier miembro del público es un licenciante, y es referido como ”usted”.

Una ”Versión modificada” del Documento se refiere a cualquier obra que contenga el Documento o una porción de él, ya sea copia literal o con modificaciones y/o traducciones a otro idioma.

Una ”Sección secundaria” es un apéndice con nombre o una sección de tema principal del Documento que trata exclusivamente con la relación entre el editor o autores del Documento con el tema general del Documento (o con materias relacionadas) y contiene nada que pudiera caer directamente dentro del tema general. Por ejemplo, si el Documento es en parte un libro de texto de matemáticas, una Sección secundaria no puede explicar algo de matemáticas. La relación puede ser una conexión histórica con el tema o con materias relacionadas, o de posición legal, comercial, filosófica, ética o política con respecto a ellas.

Las ”Secciones invariantes” son ciertas Secciones secundarias cuyos títulos están designados como ser de Secciones invariantes en el aviso que manifiesta que el Documento esta liberado bajo esta Licencia.

Los ”Textos de portada” son ciertos pasajes cortos de texto enlistados como textos de portada o de contraportada en el aviso que manifiesta que el Documento esta liberado bajo esta Licencia.

Una copia ”Transparente” del Documento se refiere a una copia legible por una máquina, representada en un formato cuya especificación esta disponible para el público en general, cuyo contenido puede ser visto y editado directamente con editores de texto genéricos o (para imágenes compuesta por pixeles) programas de pintura genéricos o (para dibujos) algún editor de dibujo ampliamente disponible, y que es adecuado para entrar en formateadores de texto o para traducción automática a una variedad de formatos adecuados para entrar en formateadores de texto. Una copia hecha en un formato de archivo contrario a uno Transparente cuyo marcado ha sido diseñado para frustrar o desalentar modificaciones subsecuentes por los lectores no es Transparente. A una copia que no es ”Transparente” se le denomina ”Opaca”.

Ejemplos de formatos adecuados para copias Transparentes incluyen ASCII plano sin marcado, formato de entrada Textinfo, formatos de entrada LaTeX, SGML o XML usando un DTD disponible públicamente, y HTML simple conforme a estándar designado para modificación humana. Los formatos opacos incluyen PostScript, PDF, formatos propietarios que solo pueden ser leídos y editados por procesadores de palabras propietarios, SGML o XML para el cual el DTD y/o las herramientas de proceso no están generalmente disponibles, y el HTML generado por máquina producido por algún procesador de palabras de propósito único de salida.

La ”Página de título” se refiere, para un libro impreso, a la página del título en sí, además de las páginas siguientes las cuales son necesarias para contener legiblemente, el material que esta Licencia requiere que aparezca en el título de la página. Para obras en formatos que no tengan una página de título como tal, ”Página de título” se refiere al texto cerca de la aparición más prominente del título de la obra, precediendo el comienzo del cuerpo del texto.

2. COPIA LITERAL

Puede copiar o distribuir el Documento por cualquier medio, ya sea comercial o no, siempre que esta Licencia, el aviso de derechos de autor y el aviso de la licencia que manifiesta que esta Licencia aplica a todo el Documento, estén reproducidos en todas la copias, y que usted no agregue cualquier otra condición a las de esta Licencia. Usted no puede tomar medidas técnicas para obstruir o controlar la lectura o copia posterior de las copias que haga o distribuya. No obstante, puede aceptar compensación a cambio de copias. Si usted distribuye una cantidad suficientemente grande de copias, debe también cumplir la condiciones de la sección 3.

También puede prestar copias, bajo las mismas condiciones citadas arriba, y puede mostrar copias públicamente.

3. COPIA EN CANTIDAD

Si usted publica copias impresas del Documento en cantidad superior a 100, y el aviso de licencia del Documento requiere Texto de portada, usted debe cubrir las copias con cubiertas que contengan clara y legiblemente, todos estos Textos de portada: Textos de portada frontal en la cubierta frontal y Textos de contraportada en la cubierta trasera. Ambas cubiertas también deben clara y legiblemente identificarlo a usted como el editor de dichas copias. La cubierta frontal debe presentar el título completo con todas sus palabras igualmente prominentes y visibles. Usted puede agregar otro material en las cubiertas en adición. Copias con cambios limitados a las cubiertas pueden ser tratadas como copias literales en otros aspectos mientras conserven el título del Documento y satisfagan estas condiciones.

Si los textos obligatorios para cualquier cubierta son demasiado voluminosos para caber legiblemente, usted debe colocar los primeros enlistados (tantos como quepan razonablemente) en la cubierta actual, y continuar con el resto dentro de las páginas adyacentes.

Si usted publica o distribuye copias Opacas del Documento en cantidad mayor a 100, usted debe incluir una copia Transparente legible para una máquina junto con cada copia Opaca, o declarar dentro o con cada copia Opaca una ubicación de computadora-red accesible públicamente conteniendo la copia Transparente completa del Documento, libre de material adicional, donde el público general usuario de la red tenga acceso para descargar anónimamente sin cargo alguno usando protocolos de red estándar públicos. Si usted usa la última opción, debe tomar razonablemente pasos prudentes, cuando comience la distribución de copias Opacas en cantidad, para garantizar que esta copia Transparente se mantenga accesible en la ubicación declarada hasta por lo menos un año después de la última vez que distribuyó una copia Opaca (directamente o a través de sus agentes o minoristas) de esa edición al público.

Se solicita, mas no se requiere, que contacte a los autores del Documento mucho antes de redistribuir cualquier número grande de copias, para darles oportunidad de proporcionarle una versión actualizada del Documento.

4. MODIFICACIONES

Usted puede copiar y distribuir una Versión modificada del Documento bajo las condiciones de las secciones 2 y 3 de arriba, dado que usted emite la Versión modificada bajo esta Licencia precisamente, con la Versión modificada cumpliendo el rol de Documento, con lo que se otorga licencia de distribución y modificación de la Versión modificada a quien posea una copia de ella. Adicionalmente, usted debe hacer esto en la Versión modificada:

A. Usar en la Página de título (y en las cubiertas, si las hay) un título distinto al del Documento y al de aquellos en versiones previas (que deberían, en caso de existir, estar enlistadas en la sección de Historial del Documento). Usted puede usar el mismo título que una versión previa si el editor original de la versión le otorga permiso. B. Listar en la Página de título, como autores, a una o mas personas o entidades responsables de autoría de las modificaciones en la Versión modificada, junto con por lo menos cinco de los autores principales del Documento (todos los autores principales si son menos de cinco). C. Mencionar en la Página de título el nombre del editor de la Versión modificada, como el editor. D. Preservar todos los avisos de derechos de autor del Documento. E. Agregar un aviso de derechos de autor para sus modificaciones adyacentes a los otros avisos de derechos de autor. F. Incluir inmediatamente después de los avisos de derechos de autor, un aviso de licencia otorgando el permiso público para usar la Versión modificada bajos los términos de esta Licencia, en la forma en que se muestra en el Addendum abajo. G. Preservar en ese aviso de licencia las listas completas de Secciones invariantes y Textos de portada obligatorios indicados en el aviso de licencia del Documento. H. Incluir una copia sin alteraciones de esta Licencia. I. Preservar la sección titulada ”Historial” y su título, y agregarle un elemento que mencione por lo menos el título, año, autores nuevos, y editor de la Versión modificada como esté en la Página de título. Si en el Documento no hay sección titulada ”Historial”, crear una mencionando título, año, autores y editor del Documento como esté en su Página de título, y luego agregar un elemento describiendo la Versión modificada como se establece en la oración previa. J. Conservar la ubicación de red, en caso de haberla, proporcionada en el Documento para acceso público a una Copia transparente del Documento, así como las ubicaciones de red proporcionadas en el Documento para versiones previas en las que este basado. Estas pueden colocarse en la sección de ”Historial”. Usted puede omitir una ubicación de red para una obra

que ha sido publicada por lo menos cuatro años antes que el Documento mismo, o si el publicador de la versión en cuestión otorga el permiso. K. Preservar el título de cualquier sección titulada ”Agradecimientos” o ”Dedicatorias”, y preservar en la sección toda la esencia y tono de cada uno de los reconocimientos de colaboración y/o las dedicatorias ahi dadas. L. Preservar todas las Secciones invariantes del Documento, sin alterar su texto ni sus títulos. Los números de sección o equivalentes no se consideran parte de los títulos de la sección. M. Eliminar cualquier sección titulada como ”Endosos”. Tales secciones pueden no ser incluídas en la Versión modificada. N. No retitule cualquier sección existente como ”Endosos” o que conflictúe con el título de cualquier Sección invariante.

Si la Versión modificada incluye nuevas secciones de materia frontal o apéndices que califiquen como Secciones secundarias y no contienen material copiado del Documento, usted puede optar por designar algunas o todas esas secciones como invariantes. Para hacer eso, agregue sus títulos a la lista de Secciones invariantes en el aviso de licencia de la Versión modificada. Estos títulos debes ser distintos a los de cualquier otra sección.

Usted puede agregar a sección titulada ”Endosos”, siempre que contenga nada mas que endosos de su Versión modificada por varios participantes, por ejemplo, declaraciones de revisión de pares o de que el texto ha sido aprobado por una organización como la definición autorizada de un estándar.

Usted puede agregar un pasaje de hasta cinco palabras como Texto de portada, y un pasaje de hasta 25 palabras como Texto de Contraportada, al final de la lista de Textos de cubierta en la Versión modificada. Solo un pasaje de Texto de portada y uno de Texto de contraportada pueden ser añadidos por (o a través de acuerdos hechos con) una entidad cualquiera. Si el Documento ya incluye un texto de cubierta para la misma cubierta, añadido previamente por usted o por acuerdo realizado con la misma entidad por parte de la que usted actúa, usted no puede añadir otro; pero puede reemplazar uno anterior con la autorización explícita del editor que lo agregó.

El(los) autor(es) y editor(es) del Documento no conceden, por medio de esta Licencia, permiso para usar sus nombres para publicitar o afirmar o implicar el endoso de cualquier Versión modificada.

5. COMBINANDO DOCUMENTOS

Usted puede combinar el Documento con otros documentos liberados bajo esta Licencia, bajo los términos definidos en la sección 4 de arriba para versiones modificadas, siempre que usted incluya en la combinación a todas la Secciones invariantes de todos los documentos originales, sin modificar, y listándolas todas como Secciones invariantes de su obra combinada en su aviso de licencia.

La obra combinada solo necesita contener una copia de esta Licencia, y múltiples Secciones invariantes idénticas pueden ser reemplazadas con una sola copia. Si hay múltiples Secciones invariantes con el mismo nombre pero con distinto contenido, haga que el título de tales secciones sea único agregando al final de él, entre paréntesis, el nombre del autor original o editor de la sección si se es conocido, o de otro modo, cualquier número único. Haga los mismos ajustes a los títulos de sección en la lista de Secciones invariantes en el aviso de licencia de la obra combinada.

En la combinación, usted debe combinar cualquier sección titulada ”Historial” en los distintos documentos originales, formando una sección titulada ”Historial”; del mismo modo, combinar cualquier sección titulada ”Agradecimientos” y cualquier sección titulada ”Dedicatorias”. Usted debe eliminar todas las secciones tituladas ”Endosos.”

6. COLECCIONES DE DOCUMENTOS

Usted puede hacer una colección consistente de el Documento y otros documentos liberados bajo esta Licencia, y reemplazar las copias individuales de esta Licencia en los varios documentos con una sola copia que este incluida en la colección, siempre que usted siga las reglas de esta Licencia para copias literales de cada uno de los documentos en todos los otros aspectos.

Usted puede extraer un documento individual de tal colección, y distribuirlo individualmente bajo esta Licencia, siempre que inserte una copia de esta Licencia en el documento extraído, y siga esta Licencia en todos los otros aspectos en relación a copia literal de ese documento.

7. AGREGACIÓN CON OBRAS INDEPENDIENTES

Una compilación del Documento o sus derivados, con otros documentos separados e independientes u obras, en un volumen de un medio de almacenamiento o distribución, no cuenta en su totalidad como una Versión modificada del Documento, siempre que no se reclame derechos de autor de compilación para la compilación. A dicha compilación se le denomina un ”agregado”, y esta Licencia no aplica para los otras obras auto-contenidas así compiladas con el Documento, por estar así compilados, si no son por si mismas, obras derivadas del Documento.

Si es aplicable el requisito de Texto de cubierta de la sección 3 a esas copias del Documento, entonces si el Documento es menor a un cuarto del agregado entero, los Textos de cubierta del Documento pueden ser colocados en las cubiertas que envuelvan solo al Documento dentro del agregado. De otro modo, deberán aparecer en las cubiertas que envuelven al agregado completo.

8. TRADUCCIÓN

A la traducción se le considera un tipo de modificación, así que usted puede distribuir traducciones del Documento bajo las términos de la sección 4. Reemplazar Secciones invariantes con traducciones requiere un permiso especial de los propietarios de los derechos de autor, pero puede incluir traducciones de algunas o todas la Secciones invariantes además de la versiones originales de esas Secciones invariantes. Usted puede incluir una traducción de esta Licencia siempre que usted incluya la versión original en Inglés de esta Licencia. En caso de un desacuerdo entre la traducción y la versión original en Inglés de esta Licencia, la versión original en Inglés prevalecerá.

9. TERMINACIÓN

Usted no puede copiar, modificar, sublicenciar o distribuir el Documento salvo lo dispuesto expresamente en esta Licencia. Cualquier otro intento de copiar, modificar, sublicenciar o distribuir el Documento es nulo, y terminará automáticamente sus derechos bajo esta Licencia. No obstante, aquellas partes que hayan recibido copias, o derechos de usted, bajo esta Licencia no tendrán por terminadas sus licencias mientras tales partes queden en pleno cumplimiento.

10. REVISIONES FUTURAS DE ESTA LICENCIA

La Fundación de Software Libre puede publicar nuevas versiones revisadas de la Licencia GNU de Documentación Libre de vez en cuando. Tales versiones nuevas serán similares en el espíritu de la versión presente, pero pueden diferir en detalle para encausar nuevos problemas o preocupaciones. Ver https://www.gnu.org/copyleft/.

A cada versión de la Licencia se le da un número de versión distintivo. Si el Documento especifica una versión particular numerada de esta Licencia ”o cualquier versión posterior” aplica a él, usted tiene la opción de seguir los términos y condiciones ya sea de esa versión especificada o de cualquier otra versión posterior que haya sido publicada (no como borrador) por la Fundación de Software Libre. Si el Documento no especifica un número de versión de esta Licencia, usted puede elegir cualquier versión ya publicada (no como borrador) por la Fundación de Software Libre.

ADDENDUM: Cómo usar esta Licencia para sus documentos

Para usar esta Licencia en un documento que usted haya escrito, incluya una copia de la Licencia en el documento y coloque los siguientes avisos de derechos de autor justo después de la página de título:

Derechos de autor (c) AÑO SU NOMBRE. Se otorga permiso para copiar, distribuir y/o modificar este documento bajo la términos de la Licencia GNU de Documentación Libre, versión 1.1 o cualquier versión posterior publicada por la Fundación de Software Libre; siendo las Secciones invariantes LISTA DE TITULOS, con los Textos de portada siendo LISTA, y los Textos de contraportada siendo LISTA. Se incluye una copia de la Licencia en la sección titulada ”Licencia GNU de Documentación Libre”.

Si no tiene Secciones invariantes, escriba ”sin Secciones invariantes” en lugar de declarar cuáles son invariantes. Si no tienen Textos de portada escriba ”sin Textos de portada” en lugar de ”Textos de portada siendo LISTA”; del mismo modo para Textos de contraportada.

Si su documento contiene ejemplos no triviales de código de programa, recomendamos liberar esos ejemplos en paralelo bajo la licencia de software libre de su elección, como la Licencia GNU de Público General, para permitir su uso en software libre.