LinuxCNC V2.10.0-pre0-5999-g6d1c9d6082

Índice general

I Primeros Pasos y Configuración 1

1. Primeros Pasos con LinuxCNC 2

1.1. Acerca de LinuxCNC . . . 2
1.1.1. Architecture - Context diagram . . . 3
1.1.2. The Operating System . . . 4
1.1.3. Conseguir ayuda . . . 4
1.1.3.1. Web Forum . . . 4
1.1.3.2. IRC . . . 4
1.1.3.3. Mailing List . . . 5
1.1.3.4. Web Forum . . . 5
1.1.3.5. Wiki de LinuxCNC . . . 5
1.1.3.6. Reportar errores . . . 5
1.2. Requisitos del sistema . . . 6
1.2.1. Minimum Requirements . . . 6
1.2.2. Kernel and Version requirements . . . 6
1.2.2.1. Preempt-RT with *linuxcnc-uspace* package . . . 7
1.2.2.2. RTAI with *linuxcnc* package . . . 7
1.2.2.3. Xenomai with *linuxcnc-uspace* package . . . 7
1.2.2.4. RTAI with *linuxcnc-uspace* package . . . 7
1.2.3. Problematic Hardware . . . 7
1.2.3.1. Laptops . . . 7
1.2.3.2. Video Cards . . . 7
1.3. Obteniendo LinuxCNC . . . 7
1.3.1. Descargar la imagen . . . 8
1.3.1.1. Descarga normal . . . 8
1.3.1.2. Descargar usando zsync . . . 8
1.3.1.3. Verificar la imagen . . . 9
1.3.2. Escribir la imagen en un dispositivo de arranque . . . 9

1.3.2.1. Imagen para Raspberry Pi . . . 9
1.3.2.2. Imagen para AMD-64 (x86-64, PC) usando herramientas de GUI . . . 9
1.3.2.3. Línea de comandos - Linux . . . 10
1.3.2.4. Línea de comandos - MacOS . . . 10
1.3.3. Probando LinuxCNC . . . 11
1.3.4. Installing LinuxCNC . . . 11
1.3.5. Actualizando LinuxCNC . . . 11
1.3.6. Problemas de instalación . . . 11
1.3.7. Métodos alternos de instalación . . . 12
1.3.7.1. Installing on Debian Trixie (with Preempt-RT kernel) . . . 12
1.3.7.2. Installing on Debian Trixie (with experimental RTAI kernel) . . . 13
1.3.7.3. Instalación en Raspbian 12 . . . 14
1.4. Ejecutando LinuxCNC . . . 14
1.4.1. Invocando LinuxCNC . . . 14
1.4.2. Selector de configuración . . . 14
1.4.3. Next steps in configuration . . . 17
1.4.4. Simulator Configurations . . . 17
1.4.5. Configuration Resources . . . 17
1.5. Actualizando LinuxCNC . . . 17
1.5.1. Upgrade to the new version . . . 18
1.5.1.1. Configuración de fuentes apt . . . 18
1.5.1.2. Upgrading to the new version . . . 20
1.5.1.3. Ubuntu . . . 21
1.5.2. Updating without Network . . . 21
1.5.3. Updating Configuration Files for 2.9 . . . 21
1.5.3.1. Manejo estricto de intérpretes enchufables . . . 21
1.5.3.2. Canterp . . . 22
1.5.3.3. Spindle limits in the INI . . . 22
1.5.4. Updating Configuration Files for 2.10.y . . . 22
1.5.5. New HAL components . . . 22
1.5.5.1. Non-Realtime . . . 22
1.5.5.2. En tiempo real . . . 22
1.5.6. New Drivers . . . 23
1.6. Preguntas y respuestas frecuentes de Linux . . . 23
1.6.1. Inicio de sesión automático . . . 23
1.6.1.1. Debian . . . 23
1.6.1.2. Ubuntu . . . 23
1.6.2. Arranque automático . . . 24
1.6.3. Terminal . . . 24

1.6.4. Páginas del manual . . . 24
1.6.5. Lista de módulos . . . 24
1.6.6. Edición de archivos de root . . . 25
1.6.6.1. Con la línea de comandos . . . 25
1.6.6.2. Usando la interfaz gráfica . . . 25
1.6.6.3. Acceso tipo super usuario . . . 25
1.6.7. Comandos en la terminal . . . 25
1.6.7.1. Directorio de trabajo . . . 25
1.6.7.2. Cambiar directorios . . . 25
1.6.7.3. Listar los archivos en un directorio . . . 26
1.6.7.4. Encontrar un archivo . . . 26
1.6.7.5. Búsqueda de texto . . . 26
1.6.7.6. Mensajes de diagnóstico . . . 26
1.6.8. Elementos convenientes . . . 27
1.6.8.1. Iniciador de terminal . . . 27
1.6.9. Problemas de hardware . . . 27
1.6.9.1. Información del hardware . . . 27
1.6.9.2. Resolución del monitor . . . 27
1.6.10Rutas . . . 27

2. Información General para el Usuario **28**
2.1. Introducción al usuario de LinuxCNC . . . 30
2.1.1. Introducción . . . 30
2.1.2. How LinuxCNC Works . . . 30
2.1.3. Interfaces de usuario gráficas . . . 32
2.1.4. Interfaces de usuario . . . 40
2.1.5. Paneles Virtuales de Control . . . 40
2.1.6. Languages . . . 43
2.1.7. Think Like a CNC Operator . . . 43
2.1.8. Modes of Operation . . . 44
2.2. Conceptos importantes . . . 44
2.2.1. Control de trayectoria . . . 44
2.2.1.1. Planificación de trayectoria . . . 44
2.2.1.2. Seguimiento de ruta . . . 45
2.2.1.3. Programando el planificador . . . 45
2.2.1.4. Planificando movimientos . . . 46
2.2.2. Códigos G . . . 47
2.2.2.1. Valores predeterminados . . . 47
2.2.2.2. Velocidad de alimentación . . . 47

2.2.2.3. Offset del radio de la herramienta . . . 47
2.2.3. Homing . . . 47
2.2.4. Cambios de herramientas . . . 48
2.2.5. Sistemas de coordenadas . . . 48
2.2.5.1. Coordenadas de máquina G53 . . . 48
2.2.5.2. Coordenadas de usuario G54-59.3 . . . 48
2.2.5.3. Cuando este perdido . . . 48
2.2.6. Configuraciones de la máquina . . . 49
2.3. Iniciando LinuxCNC . . . 50
2.3.1. Ejecutando LinuxCNC . . . 50
2.3.1.1. Selector de configuración . . . 51
2.4. Descripción general de una máquina CNC . . . 52
2.4.1. Componentes mecánicos . . . 52
2.4.1.1. Ejes . . . 52
2.4.1.2. Husillo . . . 53
2.4.1.3. Refrigerante . . . 53
2.4.1.4. Porcentaje de alimentación y velocidad . . . 53
2.4.1.5. Interruptor de borrado de bloque . . . 53
2.4.1.6. Interruptor de parada de programa opcional . . . 53
2.4.2. Componentes de control y datos . . . 54
2.4.2.1. Ejes lineales . . . 54
2.4.2.2. Ejes de rotación . . . 54
2.4.2.3. Punto controlado . . . 54
2.4.2.4. Movimiento lineal coordinado . . . 54
2.4.2.5. Velocidad de alimentación . . . 55
2.4.2.6. Enfriamiento . . . 55
2.4.2.7. Espera . . . 55
2.4.2.8. Unidades . . . 55
2.4.2.9. Posición actual . . . 55
2.4.2.10Plano seleccionado . . . 56
2.4.2.11Carrusel de herramientas . . . 56
2.4.2.12Cambio de herramienta . . . 56
2.4.2.13Pallet Shuttle . . . 56
2.4.2.14Porcentaje de velocidad . . . 56
2.4.2.15Modo de control de ruta . . . 56
2.4.3. Interacción del intérprete con interruptores . . . 56
2.4.3.1. Cancelación de interruptores de avance y velocidad . . . 57
2.4.3.2. Interruptor de borrado de bloque . . . 57
2.4.3.3. Interruptor de parada de programa opcional . . . 57

2.4.4. Tabla de herramientas . . . 57
2.4.5. Parámetros . . . 57
2.5. Información para usuarios de torno . . . 58
2.5.1. Modo torno . . . 58
2.5.2. La tabla de herramientas . . . 59
2.5.3. Orientaciones de herramienta . . . 59
2.5.4. Toque inicial de la herramienta . . . 61
2.5.4.1. Toque inicial X . . . 61
2.5.4.2. Toque inicial Z . . . 62
2.5.4.3. El offset Z de la máquina . . . 62
2.5.5. Movimiento sincronizado del husillo . . . 63
2.5.6. Arcos . . . 63
2.5.6.1. Diseño de arcos en torno . . . 63
2.5.6.2. Modo de radio y diámetro . . . 63
2.5.7. Ruta de la herramienta . . . 64
2.5.7.1. Punto de control . . . 64
2.5.7.2. Corte de ángulos sin compensación . . . 64
2.5.7.3. Cortando un radio . . . 66
2.5.7.4. Usando compensación del cortador . . . 68
2.6. Básicos del corte por plasma para usuarios de LinuxCNC . . . 68
2.6.1. ¿Qué es el plasma? . . . 68
2.6.2. Inicialización de arco . . . 69
2.6.2.1. Inicio por alta frecuencia . . . 69
2.6.2.2. Blowback Start . . . 69
2.6.3. Plasma y CNC . . . 70
2.6.4. Elegir una máquina de plasma para operaciones CNC . . . 71
2.6.5. Tipos de control de altura de antorcha . . . 72
2.6.6. Señal Arc OK . . . 73
2.6.7. Detección de altura inicial . . . 73
2.6.7.1. Interruptores flotantes . . . 73
2.6.7.2. Detección óhmica . . . 74
2.6.7.3. Hipersensibilidad con MESA THCAD-5 . . . 74
2.6.7.4. Ejemplo de código HAL para hipersensibilidad . . . 75
2.6.8. Retraso THC . . . 76
2.6.9. Muestreo de voltaje de antorcha . . . 77
2.6.10 Ruptura de antorcha . . . 77
2.6.11 Bloquear esquinas / Velocidad anti-inmersión . . . 77
2.6.12 Cruce de vacío / Kerf . . . 77
2.6.13 Agujero y corte de forma pequeña . . . 78

2.6.14 Pines de E/S para controladores de plasma . . . 79
2.6.14.1 Arco OK (entrada) . . . 79
2.6.14.2 Antorcha ON (salida) . . . 79
2.6.14.3 Interruptor flotante (entrada) . . . 80
2.6.14.4 Activación del sensor óhmico (salida) . . . 80
2.6.14.5 Detección óhmica (entrada) . . . 80
2.6.14.6 Sensor de ruptura de la antorcha . . . 80
2.6.15 Código G para controladores de plasma . . . 81
2.6.15.1 Habilitar/deshabilitar la operación del THC: . . . 81
2.6.16 Offsets externos y corte por plasma . . . 81
2.6.17 Lectura de voltaje de arco con el THCAD de Mesa . . . 82
2.6.17.1 Conexiones THCAD . . . 83
2.6.17.2 Prueba inicial de THCAD . . . 83
2.6.17.3 ¿Cuál modelo THCAD utilizar? . . . 83
2.6.18 Post procesadores y anidamiento . . . 84
2.6.19 Diseñando para entornos eléctricos ruidosos . . . 84
2.6.20 Mesas por agua . . . 85
2.6.21 Mesas de tiro descendente . . . 85
2.6.22 Diseñando para velocidad y aceleración . . . 86
2.6.23 Distancia recorrida por revolución del motor . . . 86
2.6.24 Configuración de plasma con QtPlasmaC de LinuxCNC . . . 86
2.6.25 Control Hypertherm RS485 . . . 86
2.6.26 Postprocesadores para corte por plasma . . . 87

3. Asistentes de Configuración **88**
3.1. Asistente de configuración de pasos . . . 88
3.1.1. Introducción . . . 88
3.1.2. Página inicial . . . 89
3.1.3. Información básica . . . 91
3.1.4. Ajustes del puerto paralelo . . . 93
3.1.5. Configuración del puerto paralelo 2 . . . 95
3.1.6. Configuración de ejes . . . 96
3.1.6.1. Determinar la velocidad máxima . . . 98
3.1.6.2. Determinando la aceleración máxima . . . 99
3.1.7. Configuración del husillo . . . 100
3.1.7.1. Control de velocidad del husillo . . . 100
3.1.7.2. Movimiento sincronizado del husillo . . . 101
3.1.7.3. Determinación de la calibración del husillo . . . 101
3.1.8. Opciones . . . 102

3.1.9. Completar la configuración de la máquina . . . 103
3.1.10. Carrera y casa de los ejes . . . 103
3.1.10.1. Operar sin interruptores de limite . . . 104
3.1.10.2. Operando sin interruptores de casa . . . 104
3.1.10.3. Opciones de cableado de los interruptores de casa y límite . . . 104
3.2. Asistente de configuración Mesa . . . 105
3.2.1. Instrucciones punto por punto . . . 106
3.2.2. Crear o editar . . . 106
3.2.3. Información básica de la máquina . . . 107
3.2.4. Configuración de externos . . . 109
3.2.5. Configuración de la GUI . . . 111
3.2.6. Configuración de tarjetas Mesa . . . 114
3.2.7. Configuración de E/S Mesa . . . 116
3.2.8. Configuración de puerto paralelo . . . 120
3.2.9. Configuración de ejes . . . 121
3.2.10. Configuración del husillo . . . 129
3.2.11. Opciones avanzadas . . . 130
3.2.12. Componentes HAL . . . 131
3.2.13. Uso avanzado de PnCconf . . . 132

4. Configuración 135
4.1. Conceptos para integradores . . . 135
4.1.1. Ubicaciones de archivos . . . 135
4.1.1.1. Instalado . . . 135
4.1.1.2. Línea de comandos . . . 136
4.1.2. Archivos . . . 136
4.1.3. Sistemas steppers . . . 136
4.1.3.1. Período base . . . 136
4.1.3.2. Tiempos de pasos . . . 137
4.1.4. Servosistemas . . . 137
4.1.4.1. Operación básica . . . 137
4.1.4.2. Término proporcional . . . 139
4.1.4.3. Término integral . . . 139
4.1.4.4. Término derivativo . . . 139
4.1.4.5. Ajuste de bucle . . . 140
4.1.4.6. Afinación manual . . . 140
4.1.5. S-Curve Trajectory Planning . . . 140
4.1.5.1. Enabling . . . 140
4.1.5.2. Tuning . . . 141

4.1.6. RTAI . . . 141
4.1.6.1. ACPI . . . 141
4.1.7. Opciones de hardware de interfaz computadora/máquina . . . 141
4.1.7.1. litehm2/rv901t . . . 141
4.2. Prueba de latencia . . . 141
4.2.1. ¿Qué es latencia? . . . 141
4.2.2. Pruebas de latencia . . . 142
4.2.2.1. Prueba de latencia . . . 142
4.2.2.2. Gráfica de latencia . . . 144
4.2.2.3. Histograma de latencia . . . 144
4.2.3. Ajuste de latencia . . . 146
4.2.3.1. Ajustar el BIOS para latencia . . . 146
4.2.3.2. Ajustar Preempt-RT para latencia . . . 146
4.3. Afinación de generadores de pasos . . . 147
4.3.1. Obtener el máximo rendimiento del software generador de pasos . . . 147
4.3.1.1. Ejecutar una prueba de latencia . . . 147
4.3.1.2. Descubre lo que esperan tus controladores . . . 147
4.3.1.3. Elegir el BASE_PERIOD . . . 148
4.3.1.4. Uso de steplen, stepspace, dirsetup y/o dirhold . . . 149
4.3.1.5. ¡Sin adivinar! . . . 150
4.4. Configuración de INI . . . 150
4.4.1. Componentes del archivo INI . . . 150
4.4.1.1. Comentarios . . . 150
4.4.1.2. Secciones . . . 151
4.4.1.3. Variables . . . 151
4.4.1.4. Secciones personalizadas y variables . . . 153
4.4.1.5. Inclusión de archivos . . . 154
4.4.2. Secciones del archivo INI . . . 155
4.4.2.1. Sección [EMC] . . . 155
4.4.2.2. Sección [DISPLAY] . . . 155
4.4.2.3. Sección [FILTER] . . . 160
4.4.2.4. Sección [RS274NGC] . . . 161
4.4.2.5. Sección [EMCMOT] . . . 163
4.4.2.6. Sección [TASK] . . . 163
4.4.2.7. Sección [HAL] . . . 164
4.4.2.8. Sección [HALUI] . . . 165
4.4.2.9. Sección [APPLICATIONS] . . . 165
4.4.2.10. Sección [TRAJ] . . . 166
4.4.2.11. Sección [KINS] . . . 169

4.4.2.12.Sección [AXIS_<letra>_] . . . 170
4.4.2.13.Secciones [JOINT_<número>] . . . 170
4.4.2.14.Sección(es) [SPINDLE_<número>] . . . 177
4.4.2.15.Sección [EMCIO] . . . 178
4.5. Configuración de homing . . . 179
4.5.1. Descripción general . . . 179
4.5.2. Prerrequisitos . . . 179
4.5.3. Ejemplo de disposición de interruptor de casa separado . . . 180
4.5.4. Ejemplo de disposición de interruptor de límite/casa compartido . . . 181
4.5.5. Secuencia de homing . . . 182
4.5.6. Configuración . . . 184
4.5.6.1. HOME_SEARCH_VEL . . . 184
4.5.6.2. HOME_LATCH_VEL . . . 184
4.5.6.3. HOME_FINAL_VEL . . . 184
4.5.6.4. HOME_IGNORE_LIMITS . . . 185
4.5.6.5. HOME_USE_INDEX . . . 185
4.5.6.6. HOME_INDEX_NO_ENCODER_RESET . . . 185
4.5.6.7. HOME_OFFSET . . . 185
4.5.6.8. HOME . . . 185
4.5.6.9. HOME_IS_SHARED . . . 186
4.5.6.10HOME_ABSOLUTE_ENCODER . . . 186
4.5.6.11HOME_SEQUENCE . . . 186
4.5.6.12VOLATILE_HOME . . . 187
4.5.6.13LOCKING_INDEXER . . . 187
4.5.6.14Homing inmediato . . . 188
4.5.6.15Inhibiendo homing . . . 188
4.6. Configuración de torno . . . 189
4.6.1. Plano predeterminado . . . 189
4.6.2. Configuración INI . . . 189
4.7. Inicio rápido de motores a pasos . . . 190
4.7.1. Prueba de latencia . . . 190
4.7.2. Sherline . . . 190
4.7.3. Xylotex . . . 191
4.7.4. Información de la máquina . . . 191
4.7.5. Información de disposición de pines . . . 191
4.7.6. Información mecánica . . . 192
4.8. Configuración de motor a pasos . . . 193
4.8.1. Introducción . . . 193
4.8.2. Velocidad de paso máxima . . . 193

4.8.3. Asignación de pines . . . 193
4.8.3.1. Asignación de pines estándar HAL . . . 194
4.8.3.2. Descripción general . . . 195
4.8.3.3. Cambiar standard_pinout.hal . . . 196
4.8.3.4. Cambio de polaridad de una señal . . . 196
4.8.3.5. Agregar control de velocidad PWM al husillo . . . 196
4.8.3.6. Agregar una señal de habilitación . . . 196
4.8.3.7. Botón externo ESTOP . . . 197
4.9. Diagnóstico de motores a pasos . . . 197
4.9.1. Problemas comunes . . . 197
4.9.1.1. El motor se mueve solo un paso . . . 197
4.9.1.2. Los motores no se mueven en absoluto . . . 197
4.9.1.3. Distancia incorrecta . . . 197
4.9.2. Mensajes de error . . . 197
4.9.2.1. Error de seguimiento . . . 197
4.9.2.2. Error RTAPI . . . 198
4.9.3. Pruebas . . . 198
4.9.3.1. Tiempos de pasos . . . 198
4.10 Programas filtro . . . 199
4.10.1 Introducción . . . 199
4.10.2 Configurar el INI para programas filtro . . . 199
4.10.3 Crear programas filtro basados en Python . . . 200

5. HAL (Hardware Abstraction Layer - Capa de Abstracción de Hardware) 203
5.1. Introducción a HAL . . . 203
5.1.1. Descripción general . . . 203
5.1.2. Comunicación . . . 205
5.1.3. Diseño del sistema HAL . . . 207
5.1.3.1. Selección de componentes . . . 208
5.1.3.2. Diseño de interconexión . . . 208
5.1.3.3. Implementación . . . 208
5.1.3.4. Pruebas . . . 208
5.1.3.5. Resumen . . . 209
5.1.4. HAL: Conceptos . . . 210
5.1.5. Componentes HAL . . . 211
5.1.6. Problemas de sincronización en HAL . . . 211
5.2. HAL Basics . . . 212
5.2.1. HAL Commands . . . 212
5.2.1.1. loadrt . . . 213

5.2.1.2. addf . . . 213
5.2.1.3. loadusr . . . 214
5.2.1.4. net . . . 215
5.2.1.5. setp . . . 216
5.2.1.6. sets . . . 217
5.2.1.7. unlinkp . . . 217
5.2.1.8. Obsolete Commands . . . 217
5.2.2. HAL Data . . . 218
5.2.2.1. Bit . . . 218
5.2.2.2. Float . . . 218
5.2.2.3. s32 . . . 218
5.2.2.4. u32 . . . 218
5.2.2.5. s64 . . . 218
5.2.2.6. u64 . . . 218
5.2.3. HAL Files . . . 219
5.2.4. Parámetro . . . 219
5.2.5. Basic Logic Components . . . 219
5.2.5.1. and2 . . . 219
5.2.5.2. not . . . 220
5.2.5.3. or2 . . . 220
5.2.5.4. xor2 . . . 221
5.2.6. Logic Examples . . . 221
5.2.7. Conversion Components . . . 222
5.2.7.1. weighted_sum . . . 222
5.3. HAL TWOPASS . . . 222
5.3.1. TWOPASS . . . 222
5.3.2. Post GUI . . . 224
5.3.3. Excluding .hal files . . . 225
5.3.4. Ejemplos . . . 225
5.4. HAL Tutorial . . . 226
5.4.1. Introducción . . . 226
5.4.2. Halcmd . . . 226
5.4.2.1. Notation . . . 226
5.4.2.2. Tab-completion . . . 226
5.4.2.3. The RTAPI environment . . . 227
5.4.3. A Simple Example . . . 227
5.4.3.1. Loading a component . . . 227
5.4.3.2. Examining the HAL . . . 227
5.4.3.3. Making realtime code run . . . 229

5.4.3.4. Changing Parameters . . . 230
5.4.3.5. Saving the HAL configuration . . . 231
5.4.3.6. Exiting halrun . . . 231
5.4.3.7. Restoring the HAL configuration . . . 231
5.4.3.8. Removing HAL from memory . . . 232
5.4.4. Halmeter . . . 232
5.4.5. Stepgen Example . . . 234
5.4.5.1. Installing the components . . . 234
5.4.5.2. Connecting pins with signals . . . 236
5.4.5.3. Setting up realtime execution - threads and functions . . . 236
5.4.5.4. Setting parameters . . . 238
5.4.5.5. Run it! . . . 238
5.4.6. Halscope . . . 238
5.4.6.1. Hooking up the scope probes . . . 241
5.4.6.2. Capturing our first waveforms . . . 244
5.4.6.3. Vertical Adjustments . . . 245
5.4.6.4. Triggering . . . 246
5.4.6.5. Horizontal Adjustments . . . 248
5.4.6.6. More Channels . . . 249
5.4.6.7. More samples . . . 250
5.5. HAL Examples . . . 250
5.5.1. Connecting Two Outputs . . . 250
5.5.2. Manual Toolchange . . . 251
5.5.3. Compute Velocity . . . 252
5.5.4. Soft Start Details . . . 253
5.5.5. Stand Alone HAL . . . 255
5.6. Componentes centrales . . . 256
5.6.1. Motion . . . 256
5.6.1.1. Opciones . . . 257
5.6.1.2. Pines . . . 257
5.6.1.3. Parámetros . . . 259
5.6.1.4. Funciones . . . 260
5.6.2. Husillo . . . 260
5.6.2.1. Pines . . . 260
5.6.3. Pines y parámetros de ejes y articulaciones . . . 261
5.6.4. iocontrol . . . 261
5.6.4.1. Pines . . . 262
5.6.5. Configuraciones de INI . . . 262
5.6.5.1. Pines . . . 262

5.7. HAL Component List . . . 263
5.7.1. Componentes . . . 263
5.7.1.1. User Interfaces (non-realtime) . . . 264
5.7.1.2. Motion (non-realtime) . . . 264
5.7.1.3. Controladores de Hardware . . . 264
5.7.1.4. Mesa and other I/O Cards (Realtime) . . . 265
5.7.1.5. Utilities (non-realtime) . . . 266
5.7.1.6. Signal processing (Realtime) . . . 267
5.7.1.7. Signal generation (Realtime) . . . 268
5.7.1.8. Kinematics (Realtime) . . . 269
5.7.1.9. Motion control (Realtime) . . . 270
5.7.1.10Motor control (Realtime) . . . 270
5.7.1.11Simulation/Testing . . . 270
5.7.1.12Other (Realtime) . . . 270
5.7.2. HAL API calls . . . 271
5.7.3. RTAPI calls . . . 272
5.8. HAL Component Descriptions . . . 273
5.8.1. StepGen . . . 273
5.8.1.1. Pines . . . 274
5.8.1.2. Parámetros . . . 275
5.8.1.3. Step Types . . . 275
5.8.1.4. Funciones . . . 276
5.8.2. PWMgen . . . 277
5.8.2.1. Output Types . . . 277
5.8.2.2. Pines . . . 278
5.8.2.3. Parámetros . . . 278
5.8.2.4. Funciones . . . 278
5.8.3. Codificador . . . 279
5.8.3.1. Pines . . . 279
5.8.3.2. Parámetros . . . 280
5.8.3.3. Funciones . . . 280
5.8.4. PID . . . 281
5.8.4.1. Pines . . . 281
5.8.4.2. Funciones . . . 282
5.8.5. Simulated Encoder . . . 283
5.8.5.1. Pines . . . 283
5.8.5.2. Parámetros . . . 283
5.8.5.3. Funciones . . . 283
5.8.6. Debounce . . . 284

5.8.6.1. Pines . . . 284
5.8.6.2. Parámetros . . . 284
5.8.6.3. Funciones . . . 284
5.8.7. SigGen . . . 285
5.8.7.1. Pines . . . 285
5.8.7.2. Parámetros . . . 285
5.8.7.3. Funciones . . . 286
5.8.8. `lut5` . . . 286
5.9. HAL Component Generator . . . 287
5.9.1. Introducción . . . 287
5.9.2. Instalación . . . 288
5.9.3. Compiling . . . 288
5.9.3.1. Inside the source tree . . . 288
5.9.3.2. Realtime components outside the source tree . . . 288
5.9.3.3. Non-realtime components outside the source tree . . . 289
5.9.4. Using a Component . . . 289
5.9.5. Definitions . . . 289
5.9.6. Instance creation . . . 290
5.9.7. Implicit Parameters . . . 290
5.9.8. Sintaxis . . . 290
5.9.8.1. HAL functions . . . 292
5.9.8.2. Opciones . . . 292
5.9.8.3. License and Authorship . . . 294
5.9.8.4. Per-instance data storage . . . 294
5.9.8.5. Comentarios . . . 294
5.9.9. Restrictions . . . 294
5.9.10. Convenience Macros . . . 295
5.9.11. Components with one function . . . 295
5.9.12. Component Personality . . . 295
5.9.13. Ejemplos . . . 296
5.9.13.1. constant . . . 296
5.9.13.2. sincos . . . 296
5.9.13.3. out8 . . . 297
5.9.13.4. hal_loop . . . 297
5.9.13.5. arraydemo . . . 298
5.9.13.6. rand . . . 298
5.9.13.7. logic (using personality) . . . 298
5.9.13.8. General Functions . . . 299
5.9.14. Command Line Usage . . . 300

5.10 HALTCL Files . . . 300
5.10.1 Compatibility . . . 300
5.10.2 Haltcl Commands . . . 301
5.10.3 Haltcl INI-file variables . . . 301
5.10.4 Converting HAL files to Tcl files . . . 302
5.10.5 Haltcl Notes . . . 302
5.10.6 Haltcl Examples . . . 303
5.10.7 Haltcl Interactive . . . 303
5.10.8 Haltcl Distribution Examples (sim) . . . 303
5.11 HAL User Interface . . . 304
5.11.1 Introducción . . . 304
5.11.2 MDI . . . 304
5.11.3 Example Configuration . . . 305
5.11.4 Halui Pin Reference . . . 305
5.11.4.1 Abortar . . . 305
5.11.4.2 E-Stop . . . 305
5.11.4.3 Porcentaje de alimentación . . . 305
5.11.4.4 Mist . . . 305
5.11.4.5 Flood . . . 306
5.11.4.6 Homing . . . 306
5.11.4.7 Machine . . . 306
5.11.4.8 Velocidad máxima . . . 306
5.11.4.9 MDI . . . 306
5.11.4.10 Joint . . . 307
5.11.4.11 Joint Jogging . . . 308
5.11.4.12 Eje . . . 308
5.11.4.13 Axis Jogging . . . 309
5.11.4.14 Modo . . . 309
5.11.4.15 Program . . . 310
5.11.4.16 Rapid Override . . . 310
5.11.4.17 Spindle Override . . . 310
5.11.4.18 Husillo . . . 311
5.11.4.19 Tool . . . 311
5.12 Halui Examples . . . 311
5.12.1 Remote Start . . . 312
5.12.2 Pause & Resume . . . 312
5.13 Creating Non-realtime Python Components . . . 313
5.13.1 Basic usage example . . . 313
5.13.2 Non-realtime components and delays . . . 314

5.13.3 Create pins and parameters . . . 314
5.13.3.1 Changing the prefix . . . 315
5.13.4 Reading and writing pins and parameters . . . 315
5.13.4.1 Driving output (HAL_OUT) pins . . . 315
5.13.4.2 Driving bidirectional (HAL_IO) pins . . . 315
5.13.5 Exiting . . . 315
5.13.6 Helpful Functions . . . 316
5.13.7 Constantes . . . 316
5.13.8 System Information . . . 316
5.14 Canonical Device Interfaces . . . 316
5.14.1 Introducción . . . 316
5.14.2 Digital Input . . . 317
5.14.2.1 Pines . . . 317
5.14.2.2 Parámetros . . . 317
5.14.2.3 Funciones . . . 317
5.14.3 Digital Output . . . 317
5.14.3.1 Pines . . . 317
5.14.3.2 Parámetros . . . 317
5.14.3.3 Funciones . . . 317
5.14.4 Analog Input . . . 317
5.14.4.1 Pines . . . 317
5.14.4.2 Parámetros . . . 317
5.14.4.3 Funciones . . . 318
5.14.5 Analog Output . . . 318
5.14.5.1 Pines . . . 318
5.14.5.2 Parámetros . . . 318
5.14.5.3 Funciones . . . 318
5.15 HAL Tools . . . 318
5.15.1 Halcmd . . . 318
5.15.2 Halmeter . . . 319
5.15.3 Halshow . . . 321
5.15.4 Halscope . . . 322
5.15.5 Sim Pin . . . 322
5.15.6 Simulate Probe . . . 323
5.15.7 HAL Histogram . . . 324
5.15.8 Halreport . . . 325

6. Controladores de Hardware 328
6.1. Parallel Port Driver . . . 328
6.1.1. Loading . . . 328
6.1.2. PCI Port Address . . . 331
6.1.3. Pines . . . 332
6.1.4. Parámetros . . . 332
6.1.5. Funciones . . . 332
6.1.6. Common problems . . . 333
6.1.7. Using DoubleStep . . . 333
6.1.8. probe_parport . . . 333
6.1.8.1. Installing probe_parport . . . 333
6.2. Controlador AX5214H . . . 334
6.2.1. Instalación . . . 334
6.2.2. Pines . . . 334
6.2.3. Parámetros . . . 334
6.2.4. Funciones . . . 335
6.3. Controlador General Mechatronics . . . 335
6.3.1. Conectores de E/S . . . 336
6.3.1.1. Pines . . . 337
6.3.1.2. Parámetros . . . 337
6.3.2. Conectores de eje . . . 338
6.3.2.1. Módulos de interfaz de eje . . . 338
6.3.2.2. Codificador . . . 339
6.3.2.3. Módulo StepGen . . . 342
6.3.2.4. Señales de habilitación y falla . . . 345
6.3.2.5. Axis DAC . . . 345
6.3.3. CAN-bus servo amplifiers . . . 346
6.3.3.1. Pines . . . 347
6.3.3.2. Parámetros . . . 347
6.3.4. Watchdog timer . . . 347
6.3.4.1. Pines . . . 347
6.3.4.2. Parámetros . . . 347
6.3.5. End-, homing- and E-stop switches . . . 348
6.3.5.1. Pines . . . 349
6.3.5.2. Parámetros . . . 349
6.3.6. Status LEDs . . . 349
6.3.6.1. CAN . . . 349
6.3.6.2. RS485 . . . 350
6.3.6.3. EMC . . . 350

6.3.6.4. Boot . . . 350
6.3.6.5. Error . . . 350
6.3.7. RS485 I/O expander modules . . . 350
6.3.7.1. Relay output module . . . 351
6.3.7.2. Digital input module . . . 352
6.3.7.3. DAC & ADC module . . . 353
6.3.7.4. Teach Pendant module . . . 354
6.3.8. Errata . . . 355
6.3.8.1. GM6-PCI card Errata . . . 355
6.4. GS2 VFD Driver . . . 355
6.4.1. Command Line Options . . . 355
6.4.2. Pines . . . 356
6.4.3. Parámetros . . . 357
6.5. HAL Driver for Raspberry Pi GPIO pins . . . 357
6.5.1. Purpose . . . 357
6.5.2. Utilización . . . 357
6.5.3. Pines . . . 358
6.5.4. Parámetros . . . 358
6.5.5. Funciones . . . 359
6.5.6. Pin Numbering . . . 359
6.5.7. Known Bugs . . . 359
6.6. Generic driver for any GPIO supported by gpiod. . . . 359
6.6.1. Purpose . . . 359
6.6.2. Utilización . . . 360
6.6.3. Pines . . . 361
6.6.4. Parámetros . . . 361
6.6.5. Funciones . . . 361
6.6.6. Pin Identification . . . 361
6.6.7. Troubleshooting permissions problems. . . . 362
6.6.8. Author . . . 362
6.6.9. Known Bugs . . . 362
6.7. Mesa HostMot2 Driver . . . 362
6.7.1. Introducción . . . 362
6.7.2. Firmware Binaries . . . 363
6.7.3. Installing Firmware . . . 363
6.7.4. Loading HostMot2 . . . 363
6.7.5. Watchdog . . . 364
6.7.5.1. Pines . . . 364
6.7.5.2. Parámetros . . . 364

6.7.6. HostMot2 Functions . . . 364
6.7.7. Pinouts . . . 364
6.7.8. PIN Files . . . 366
6.7.9. Firmware . . . 366
6.7.10 HAL Pins . . . 366
6.7.11 Configurations . . . 367
6.7.12 GPIO . . . 369
6.7.12.1 Pines . . . 369
6.7.12.2 Parámetros . . . 369
6.7.13 StepGen . . . 369
6.7.13.1 Pines . . . 370
6.7.13.2 Parámetros . . . 370
6.7.13.3 Output Parameters . . . 371
6.7.14 PWMGen . . . 371
6.7.14.1 Pines . . . 371
6.7.14.2 Parámetros . . . 371
6.7.14.3 Output Parameters . . . 372
6.7.15 Codificador . . . 372
6.7.15.1 Pines . . . 372
6.7.15.2 Parámetros . . . 373
6.7.16 5I25 Configuration . . . 373
6.7.16.1 Firmware . . . 373
6.7.16.2 Configuración . . . 373
6.7.16.3 SSERIAL Configuration . . . 374
6.7.16.4 7I77 Limits . . . 374
6.7.17 Example Configurations . . . 374
6.8. MB2HAL . . . 375
6.8.1. Introducción . . . 375
6.8.2. Utilización . . . 375
6.8.3. Opciones . . . 375
6.8.3.1. Init Section . . . 375
6.8.3.2. Transaction Sections . . . 376
6.8.3.3. Error codes . . . 377
6.8.4. Example config file . . . 378
6.8.5. Pines . . . 383
6.8.5.1. fnct_01_read_coils . . . 383
6.8.5.2. fnct_02_read_discrete_inputs . . . 383
6.8.5.3. fnct_03_read_holding_registers . . . 383
6.8.5.4. fnct_04_read_input_registers . . . 383

6.8.5.5. fnct_05_write_single_coil . . . 383
6.8.5.6. fnct_06_write_single_register . . . 383
6.8.5.7. fnct_15_write_multiple_coils . . . 384
6.8.5.8. fnct_16_write_multiple_registers . . . 384
6.9. Mitsub VFD Driver . . . 384
6.9.1. Command Line Options . . . 384
6.9.2. Pines . . . 385
6.9.3. Ejemplo HAL . . . 385
6.9.4. Configuring the Mitsubishi VFD for serial usage . . . 386
6.9.4.1. Connecting the Serial Port . . . 386
6.9.4.2. Modbus setup . . . 386
6.10 Motenc Driver . . . 386
6.10.1 Pines . . . 387
6.10.2 Parámetros . . . 387
6.10.3 Funciones . . . 388
6.11 Opto22 Driver . . . 388
6.11.1 The Adapter Card . . . 388
6.11.2 The Driver . . . 389
6.11.3 Pines . . . 389
6.11.4 Parámetros . . . 389
6.11.5 FUNCTIONS . . . 389
6.11.6 Configuring I/O Ports . . . 390
6.11.7 Pin Numbering . . . 390
6.12 Pico Drivers . . . 390
6.12.1 Command Line Options . . . 391
6.12.2 Pines . . . 391
6.12.3 Parámetros . . . 393
6.12.4 Funciones . . . 394
6.13 Pluto P Driver . . . 394
6.13.1 General Info . . . 394
6.13.1.1 Requisitos . . . 394
6.13.1.2 Connectors . . . 394
6.13.1.3 Physical Pins . . . 395
6.13.1.4 LED . . . 395
6.13.1.5 Power . . . 395
6.13.1.6 PC interface . . . 395
6.13.1.7 Rebuilding the FPGA firmware . . . 395
6.13.1.8 For more information . . . 396
6.13.2 Pluto Servo . . . 396

6.13.2.1 Asignación de pines . . . 396
6.13.2.2 Input latching and output updating . . . 397
6.13.2.3 HAL Functions, Pins and Parameters . . . 398
6.13.2.4 Compatible driver hardware . . . 398
6.13.3 Pluto Step . . . 398
6.13.3.1 Asignación de pines . . . 398
6.13.3.2 Input latching and output updating . . . 399
6.13.3.3 Step Waveform Timings . . . 399
6.13.3.4 HAL Functions, Pins and Parameters . . . 400
6.14 Powermax Modbus Driver . . . 400
6.14.1 Pines . . . 401
6.14.2 Descripción . . . 401
6.14.3 Referencia: . . . 401
6.15 Servo To Go Driver . . . 402
6.15.1 Instalación . . . 402
6.15.2 Pines . . . 402
6.15.3 Parámetros . . . 403
6.15.4 Funciones . . . 403
6.16 Shuttle . . . 403
6.16.1 Descripción . . . 403
6.16.2 Setup . . . 404
6.16.3 Pines . . . 404
6.17 VFS11 VFD Driver . . . 405
6.17.1 Command Line Options . . . 405
6.17.2 Pines . . . 405
6.17.3 Parámetros . . . 407
6.17.4 INI file configuration . . . 407
6.17.5 Ejemplo HAL . . . 408
6.17.6 Panel operation . . . 409
6.17.7 Error Recovery . . . 409
6.17.8 Configuring the VFS11 VFD for Modbus usage . . . 409
6.17.8.1 Connecting the Serial Port . . . 409
6.17.8.2 Modbus setup . . . 409
6.17.9 Programming Note . . . 410

7. Ejemplos de Hardware **411**
7.1. Puerto paralelo PCI . . . 411
7.2. Spindle Control . . . 412
7.2.1. 0-10 Volt Spindle Speed . . . 412
7.2.2. PWM Spindle Speed . . . 412
7.2.3. Spindle Enable . . . 412
7.2.4. Spindle Direction . . . 413
7.2.5. Spindle Soft Start . . . 413
7.2.6. Spindle Feedback . . . 414
7.2.6.1. Movimiento sincronizado del husillo . . . 414
7.2.6.2. Spindle At Speed . . . 415
7.3. MPG Pendant . . . 415
7.4. GS2 Spindle . . . 417
7.4.1. Ejemplo . . . 417

8. Ladder Clásico **419**
8.1. ClassicLadder Introduction . . . 419
8.1.1. History . . . 419
8.1.2. Introducción . . . 419
8.1.3. Ejemplo . . . 420
8.1.4. Basic Latching On-Off Circuit . . . 421
8.2. ClassicLadder Programming . . . 422
8.2.1. Ladder Concepts . . . 422
8.2.2. Languages . . . 422
8.2.3. Componentes . . . 422
8.2.3.1. Archivos . . . 422
8.2.3.2. Realtime Module . . . 423
8.2.3.3. Variables . . . 423
8.2.4. Loading the ClassicLadder non-realtime module . . . 424
8.2.5. ClassicLadder GUI . . . 424
8.2.5.1. Sections Manager . . . 425
8.2.5.2. Section Display . . . 425
8.2.5.3. The Variable Windows . . . 427
8.2.5.4. Symbol Window . . . 429
8.2.5.5. The Editor window . . . 430
8.2.5.6. Config Window . . . 431
8.2.6. Ladder objects . . . 433
8.2.6.1. CONTACTS . . . 433
8.2.6.2. IEC TIMERS . . . 433

8.2.6.3. TIMERS . . . 434
8.2.6.4. MONOSTABLES . . . 434
8.2.6.5. COUNTERS . . . 434
8.2.6.6. COMPARE . . . 435
8.2.6.7. VARIABLE ASSIGNMENT . . . 436
8.2.6.8. COILS . . . 438
8.2.7. ClassicLadder Variables . . . 439
8.2.8. GRAFCET (State Machine) Programming . . . 440
8.2.9. Modbus . . . 442
8.2.10MODBUS Settings . . . 445
8.2.10.1MODBUS Info . . . 446
8.2.10.2Communication Errors . . . 446
8.2.11Debugging modbus problems . . . 446
8.2.11.1Request . . . 448
8.2.11.2Error response . . . 449
8.2.11.3Data response . . . 450
8.2.11.4MODBUS Bugs . . . 451
8.2.12Setting up ClassicLadder . . . 451
8.2.12.1Add the Modules . . . 452
8.2.12.2Adding Ladder Logic . . . 452
8.3. ClassicLadder Examples . . . 457
8.3.1. Wrapping Counter . . . 457
8.3.2. Reject Extra Pulses . . . 458
8.3.3. External E-Stop . . . 459
8.3.4. Timer/Operate Example . . . 462

9. Temas avanzados 464
9.1. Cinemática . . . 464
9.1.1. Introducción . . . 464
9.1.1.1. Joints vs Axes . . . 464
9.1.2. Trivial Kinematics . . . 465
9.1.3. Non-trivial kinematics . . . 466
9.1.3.1. Forward transformation . . . 467
9.1.3.2. Inverse transformation . . . 468
9.1.4. Implementation details . . . 468
9.1.4.1. Kinematics module using the userkins.comp template . . . 469
9.2. Setting up ”modified” Denavit-Hartenberg (DH) parameters for *genserkins* . . . 469
9.2.1. Prelude . . . 469
9.2.2. En general . . . 470

9.2.3. Modified DH-Parameters . . . 470
9.2.4. Modified DH-Parameters as used in *genserkins* . . . 470
9.2.5. Numbering of joints and parameters . . . 471
9.2.6. How to start . . . 471
9.2.7. Special cases . . . 471
9.2.8. Detailed Example (RV-6SL) . . . 471
9.2.9. Credits . . . 490
9.3. 5-Axis Kinematics . . . 490
9.3.1. Introducción . . . 490
9.3.2. 5-Axis Machine Tool Configurations . . . 490
9.3.3. Tool Orientation and Location . . . 490
9.3.4. Translation and Rotation Matrices . . . 491
9.3.5. Configuraciones de 5 ejes con mesas giratorias/inclinables . . . 492
9.3.5.1. Transformations for a xyzac-trt machine tool with work offsets . . . 494
9.3.5.2. Transformations for a xyzac-trt machine with rotary axis offsets . . . 498
9.3.5.3. Transformations for a xyzbc-trt machine with rotary axis offsets . . . 501
9.3.6. Table Rotary/Tilting Examples . . . 504
9.3.6.1. Vismach Simulation Models . . . 504
9.3.6.2. Tool-Length Compensation . . . 504
9.3.7. Custom Kinematics Components . . . 504
9.3.8. Figures . . . 506
9.3.9. REFERENCES . . . 508
9.4. Switchable Kinematics (switchkins) . . . 508
9.4.1. Introducción . . . 508
9.4.2. Switchable Kinematic Modules . . . 509
9.4.2.1. Identity letter assignments . . . 509
9.4.2.2. Backwards compatibility . . . 510
9.4.3. HAL Pins . . . 510
9.4.3.1. HAL Pin Summary . . . 510
9.4.4. Utilización . . . 510
9.4.4.1. HAL Connections . . . 510
9.4.4.2. G-/M-code commands . . . 511
9.4.4.3. INI file limit settings . . . 511
9.4.4.4. Consideraciones de offsets del sistema de coordenadas . . . 513
9.4.4.5. External offset considerations . . . 513
9.4.5. Simulation configs . . . 513
9.4.6. User kinematics provisions . . . 514
9.4.7. Warnings . . . 514
9.4.8. Notas sobre el código . . . 514

9.5. PID Tuning . . . 515
9.5.1. PID Controller . . . 515
9.5.1.1. Control loop basics . . . 515
9.5.1.2. Theory . . . 516
9.5.1.3. Loop Tuning . . . 516
9.5.1.4. Automatic PID tuning . . . 517
9.6. Remap Extending G-code . . . 518
9.6.1. Introduction: Extending the RS274NGC Interpreter by Remapping Codes . . . 518
9.6.1.1. A Definition: Remapping Codes . . . 518
9.6.1.2. Why would you want to extend the RS274NGC Interpreter? . . . 519
9.6.2. Getting started . . . 520
9.6.2.1. Builtin Remaps . . . 520
9.6.2.2. Picking a code . . . 521
9.6.2.3. Parameter handling . . . 521
9.6.2.4. Handling results . . . 522
9.6.2.5. Execution sequencing . . . 522
9.6.2.6. An minimal example remapped code . . . 522
9.6.3. Configuring Remapping . . . 523
9.6.3.1. The REMAP statement . . . 523
9.6.3.2. Useful REMAP option combinations . . . 524
9.6.3.3. The argspec parameter . . . 524
9.6.4. Upgrading an existing configuration for remapping . . . 528
9.6.5. Remapping tool change-related codes: T, M6, M61 . . . 528
9.6.5.1. Descripción general . . . 528
9.6.5.2. Understanding the role of `iocontrol` with remapped tool change codes . . . 529
9.6.5.3. Specifying the M6 replacement . . . 530
9.6.5.4. Configuring `iocontrol` with a remapped M6 . . . 532
9.6.5.5. Writing the change and prepare O-word procedures . . . 532
9.6.5.6. Making minimal changes to the built in codes, including `M6` . . . 533
9.6.5.7. Specifying the T (prepare) replacement . . . 533
9.6.5.8. Error handling: dealing with abort . . . 534
9.6.5.9. Error handling: failing a remapped code NGC procedure . . . 536
9.6.6. Reasignando otros códigos existentes: . . . 536
9.6.6.1. Automatic gear selection be remapping S (set spindle speed) . . . 536
9.6.6.2. Adjusting the behavior of M0, M1 . . . 536
9.6.6.3. Adjusting the behavior of M7, M8, M9 . . . 537
9.6.7. Creating new G-code cycles . . . 537
9.6.8. Configuring Embedded Python . . . 537
9.6.8.1. Python plugin : INI file configuration . . . 538

9.6.8.2. Executing Python statements from the interpreter . . . 538
9.6.9. Programming Embedded Python in the RS274NGC Interpreter . . . 539
9.6.9.1. The Python plugin namespace . . . 539
9.6.9.2. The Interpreter as seen from Python . . . 539
9.6.9.3. The Interpreter `__init__` and `__delete__` functions . . . 539
9.6.9.4. Calling conventions: NGC to Python . . . 540
9.6.9.5. Calling conventions: Python to NGC . . . 542
9.6.9.6. Built in modules . . . 544
9.6.10 Adding Predefined Named Parameters . . . 545
9.6.11 Standard Glue routines . . . 545
9.6.11.1 T: `prepare_prolog` and `prepare_epilog` . . . 545
9.6.11.2 M6: `change_prolog` and `change_epilog` . . . 546
9.6.11.3 G-code Cycles: `cycle_prolog` and `cycle_epilog` . . . 547
9.6.11.4 S (Set Speed) : `setspeed_prolog` and `setspeed_epilog` . . . 547
9.6.11.5 F (Set Feed) : `setfeed_prolog` and `setfeed_epilog` . . . 547
9.6.11.6 M61 Set tool number : `settool_prolog` and `settool_epilog` . . . 548
9.6.12 Remapped code execution . . . 548
9.6.12.1 NGC procedure call environment during remaps . . . 548
9.6.12.2 Nested remapped codes . . . 548
9.6.12.3 Sequence number during remaps . . . 548
9.6.12.4 Debugging flags . . . 548
9.6.12.5 Debugging Embedded Python code . . . 548
9.6.13 Axis Preview and Remapped code execution . . . 550
9.6.14 Remappable Codes . . . 550
9.6.14.1 Existing codes which can be remapped . . . 550
9.6.14.2 Currently unallocated G-codes: . . . 551
9.6.14.3 Currently unallocated M-codes: . . . 554
9.6.15 A short survey of LinuxCNC program execution . . . 554
9.6.15.1 Interpreter state . . . 554
9.6.15.2 Task and Interpreter interaction, Queuing and Read-Ahead . . . 555
9.6.15.3 Predicting the machine position . . . 555
9.6.15.4 Queue-busters break position prediction . . . 555
9.6.15.5 Cómo tratar los busters de cola . . . 556
9.6.15.6 Word order and execution order . . . 556
9.6.15.7 Parsing . . . 556
9.6.15.8 Execution . . . 556
9.6.15.9 Procedure execution . . . 557
9.6.15.10 How tool change currently works . . . 557
9.6.15.11 How Tx (Prepare Tool) works . . . 557

9.6.15.12 How M6 (Change tool) works . . . 558
9.6.15.13 How M61 (Change tool number) works . . . 559
9.6.16 Status . . . 559
9.6.17 Changes . . . 559
9.6.18 Debugging . . . 559
9.7. Componente moveoff . . . 560
9.7.1. Modificar una configuración existente . . . 561
9.8. Intérprete independiente . . . 564
9.8.1. Utilización . . . 564
9.8.2. Ejemplo . . . 564
9.9. External Axis Offsets . . . 565
9.9.1. INI File Settings . . . 565
9.9.2. HAL Pins . . . 566
9.9.2.1. Per-Axis Motion HAL Pins . . . 566
9.9.2.2. Other Motion HAL Pins . . . 566
9.9.3. Utilización . . . 566
9.9.3.1. Offset Computation . . . 566
9.9.3.2. Machine-off/Machine-on . . . 566
9.9.3.3. Límites de software . . . 567
9.9.3.4. Notes . . . 567
9.9.3.5. Warning . . . 568
9.9.4. Related HAL Components . . . 568
9.9.4.1. eoffset_per_angle.comp . . . 568
9.9.5. Pruebas . . . 568
9.9.6. Ejemplos . . . 569
9.9.6.1. eoffsets.ini . . . 569
9.9.6.2. jwp_z.ini . . . 569
9.9.6.3. dynamic_offsets.ini . . . 569
9.9.6.4. opa.ini (eoffset_per_angle) . . . 570
9.10. Tool Database Interface . . . 570
9.10.1 Interface . . . 570
9.10.1.1 INI file Settings . . . 570
9.10.1.2 **db_program** operation (v2.1) . . . 570
9.10.1.3 Utilización . . . 571
9.10.1.4 Programa de ejemplo . . . 572
9.10.1.5 Python tooldb module . . . 573
9.10.2 Simulation configs . . . 573
9.10.2.1 Notes . . . 574

II Utilización 575

10 Interfaces de usuario 576

10.1 GUI AXIS . . . 576
10.1.1 Introducción . . . 576
10.1.2 Primeros pasos . . . 577
10.1.2.1 Configuraciones de INI . . . 577
10.1.2.2 Una sesión típica . . . 578
10.1.3 Pantalla AXIS . . . 578
10.1.3.1 Elementos del menú . . . 579
10.1.3.2 Botones de la barra de herramientas . . . 583
10.1.3.3 Área de visualización gráfica . . . 584
10.1.3.4 Área de visualización de texto . . . 585
10.1.3.5 Control manual . . . 586
10.1.3.6 MDI . . . 588
10.1.3.7 Porcentaje de alimentación . . . 589
10.1.3.8 Porcentaje de velocidad del husillo . . . 589
10.1.3.9 Velocidad de trote . . . 589
10.1.3.10 Velocidad máxima . . . 590
10.1.4 Controles del teclado . . . 590
10.1.4.1 Teclas de porcentaje de alimentación . . . 590
10.1.5 Mostrar estado de LinuxCNC (`linuxcnctop`) . . . 591
10.1.6 Interfaz MDI . . . 592
10.1.7 axis-remote . . . 593
10.1.8 Cambio manual de herramienta . . . 593
10.1.9 Módulos Python . . . 593
10.1.10 AXIS en modo torno . . . 594
10.1.11 Usando AXIS en el modo de corte de espuma . . . 597
10.1.12 Configuración avanzada . . . 598
10.1.12.1 Filtros de programa . . . 599
10.1.12.2 La base de datos de recursos X . . . 600
10.1.12.3 Perilla de trote . . . 600
10.1.12.4 ~/.axisrc . . . 600
10.1.12.5 USER_COMMAND_FILE . . . 601
10.1.12.6 user_live_update() . . . 601
10.1.12.7 user_hal_pins() . . . 601
10.1.12.8 Editor externo . . . 601
10.1.12.9 Panel de control virtual . . . 601
10.1.12.10 Control de vista previa . . . 601

10.1.12.1 Touch off usando posición real . . . 602
10.1.13 Axisui . . . 602
10.1.14 Sugerencias de personalización de AXIS . . . 603
10.1.14.1 La función de actualización . . . 603
10.1.14.2 Deshabilitar el cuadro de diálogo Cerrar . . . 603
10.1.14.3 Cambiar la fuente del texto . . . 603
10.1.14.4 Modificar velocidad rápida con atajos de teclado . . . 604
10.1.14.5 Leer el archivo INI . . . 604
10.1.14.6 Lectura del estado de LinuxCNC . . . 604
10.1.14.7 Cambiar la vista actual . . . 605
10.1.14.8 Crear nuevos pines AXISUI HAL . . . 605
10.1.14.9 Crear nuevos componentes y pines HAL . . . 605
10.1.14.10 Cambiar pestañas con pines HAL . . . 605
10.1.14.11 Agregar un botón para ir a casa . . . 606
10.1.14.12 Agregar botón al marco manual . . . 606
10.1.14.13 Lectura de variables internas . . . 607
10.1.14.14 Ocultar widgets . . . 608
10.1.14.15 Cambiar una etiqueta . . . 608
10.1.14.16 Redirigir un comando existente . . . 608
10.1.14.17 Cambiar el color DRO . . . 609
10.1.14.18 Cambiar los botones de la barra de herramientas . . . 609
10.1.14.19 Cambiar colores del plóter . . . 610
10.2 GMOCCAPY . . . 610
10.2.1 Introducción . . . 610
10.2.2 Requisitos . . . 611
10.2.3 How to get GMOCCAPY . . . 611
10.2.4 Configuración básica . . . 612
10.2.4.1 The DISPLAY Section . . . 613
10.2.4.2 The TRAJ Section . . . 614
10.2.4.3 Macro Buttons . . . 615
10.2.4.4 Pestañas y paneles incrustados . . . 617
10.2.4.5 User Created Messages . . . 620
10.2.4.6 Control de vista previa . . . 621
10.2.4.7 User Command File . . . 621
10.2.4.8 User CSS File . . . 622
10.2.4.9 Logging . . . 622
10.2.5 HAL Pins . . . 623
10.2.5.1 Right and Bottom Button Lists . . . 623
10.2.5.2 Velocities and Overrides . . . 626

10.2.5.3 Jog HAL Pins . . . 628
10.2.5.4 Jog Velocities and Turtle-Jog HAL Pin . . . 629
10.2.5.5 Jog Increment HAL Pins . . . 629
10.2.5.6 Hardware Unlock Pin . . . 630
10.2.5.7 Error/Warning Pins . . . 630
10.2.5.8 User Created Message HAL Pins . . . 630
10.2.5.9 Spindle Feedback Pins . . . 631
10.2.5.10 Pins to Indicate Program Progress Information . . . 631
10.2.5.11 Tool Related Pins . . . 631
10.2.6 Auto Tool Measurement . . . 633
10.2.6.1 Provided Pins . . . 634
10.2.6.2 INI File Modifications . . . 634
10.2.6.3 Needed Files . . . 635
10.2.6.4 Needed HAL Connections . . . 636
10.2.7 The Settings Page . . . 636
10.2.7.1 Appearance . . . 637
10.2.7.2 Behaviour . . . 641
10.2.7.3 Hardware . . . 645
10.2.7.4 Advanced Settings . . . 646
10.2.8 Icon Theme . . . 647
10.2.8.1 Custom Icon Theme . . . 648
10.2.8.2 Symbolic Icons . . . 649
10.2.9 Lathe Specific Section . . . 649
10.2.10 Plasma Specific Section . . . 652
10.2.11 Videos on YouTube . . . 652
10.2.11.1 Basic Usage . . . 653
10.2.11.2 Simulated Jog Wheels . . . 653
10.2.11.3 Settings Page . . . 653
10.2.11.4 Simulated Hardware Button . . . 653
10.2.11.5 User Tabs . . . 653
10.2.11.6 Tool Measurement Videos . . . 653
10.2.12 Known Problems . . . 653
10.2.12.1 Strange numbers in the info area . . . 653
10.2.12.2 Not ending macro . . . 654
10.3 The Touchy Graphical User Interface . . . 654
10.3.1 Panel Configuration . . . 655
10.3.1.1 HAL connections . . . 655
10.3.1.2 Recommended for any setup . . . 656
10.3.2 Setup . . . 656

10.3.2.1 Enabling Touchy . . . 656
10.3.2.2 Preferences . . . 656
10.3.2.3 Macros . . . 657
10.4 Gscreen . . . 657
10.4.1 Introducción . . . 657
10.4.1.1 Glade File . . . 662
10.4.1.2 PyGTK . . . 662
10.4.2 GladeVCP . . . 663
10.4.2.1 Descripción general . . . 663
10.4.2.2 Build a GladeVCP Panel . . . 664
10.4.3 Building a simple clean-sheet custom screen . . . 665
10.4.4 Handler file example . . . 667
10.4.4.1 Adding Keybindings Functions . . . 668
10.4.4.2 Linuxcnc State Status . . . 668
10.4.4.3 Jogging Keys . . . 669
10.4.5 Gscreen Start Up . . . 669
10.4.6 Configuración INI . . . 671
10.4.7 User Dialog Messages . . . 671
10.4.7.1 Copy the Stock Handler/Glade File For Modification . . . 672
10.5 QtDragon GUI . . . 673
10.5.1 Introducción . . . 673
10.5.1.1 QtDragon . . . 673
10.5.1.2 QtDragon_lathe . . . 674
10.5.1.3 QtDragon_hd . . . 675
10.5.1.4 QtDragon_hd_vertical . . . 675
10.5.2 Getting Started - The INI File . . . 675
10.5.2.1 Display . . . 676
10.5.2.2 Preferences . . . 676
10.5.2.3 Logging . . . 676
10.5.2.4 Anular controles . . . 676
10.5.2.5 Spindle controls . . . 676
10.5.2.6 Jogging increments . . . 677
10.5.2.7 Grid Increments . . . 677
10.5.2.8 Jog speed . . . 677
10.5.2.9 User message dialog system . . . 677
10.5.2.10 Embed Custom VCP Panels . . . 678
10.5.2.11 Subroutine Paths . . . 679
10.5.2.12 Control de vista previa . . . 680
10.5.2.13 Program Extensions/Filters . . . 680

10.5.2.14 Probe/Touchplate/Laser Settings . . . 680
10.5.2.15 Abort detection . . . 681
10.5.2.16 Startup codes . . . 681
10.5.2.17 Macro Buttons . . . 681
10.5.2.18 Post GUI HAL File . . . 682
10.5.2.19 Post GUI HAL Command . . . 682
10.5.2.20 HAL Bridge . . . 682
10.5.2.21 Builtin Sample Configurations . . . 683
10.5.3 Key Bindings . . . 683
10.5.4 Buttons . . . 683
10.5.5 Virtual Keyboard . . . 683
10.5.6 HAL Pins . . . 684
10.5.7 HAL files . . . 685
10.5.8 Manual Tool Changes . . . 685
10.5.9 Husillo . . . 686
10.5.10 Auto Raise Z Axis on Program Pause . . . 686
10.5.11 Z level compensation . . . 687
10.5.11.1 Using G-code Ripper for Z level Compensation . . . 688
10.5.12 Probing . . . 690
10.5.12.1 Versa Probe . . . 691
10.5.12.2 Basic probe . . . 693
10.5.12.3 Customizing Probe Screen Widget . . . 696
10.5.13 Touch plate . . . 696
10.5.14 Auto Tool Measurement . . . 697
10.5.14.1 Descripción general . . . 697
10.5.14.2 Workflow Overview . . . 697
10.5.14.3 Detailed Workflow Example . . . 699
10.5.14.4 Work Piece Height Probing in QtDragon_hd . . . 700
10.5.14.5 Work Piece Height Probing . . . 701
10.5.14.6 Tool Measurement Pins . . . 702
10.5.14.7 Tool Measurement INI File Modifications . . . 703
10.5.14.8 Required HAL Connections . . . 704
10.5.15 Run from Line . . . 705
10.5.16 Laser buttons . . . 705
10.5.17 Tabs Description . . . 705
10.5.17.1 Main tab . . . 705
10.5.17.2 File Tab . . . 705
10.5.17.3 Offsets Tab . . . 706
10.5.17.4 Tool Tab . . . 706

10.5.17.1 Status Tab . . . 706
10.5.17.2 Probe Tab . . . 706
10.5.17.3 Camview Tab . . . 706
10.5.17.4 G-codes Tab . . . 707
10.5.17.5 Setup Tab . . . 707
10.5.17.6 Settings Tab . . . 708
10.5.17.7 Utilities Tab . . . 708
10.5.17.8 User Tab . . . 709
10.5.18 Styles . . . 709
10.5.19 Internationalisation . . . 709
10.5.20 Customization . . . 710
10.5.20.1 Stylesheets . . . 711
10.5.20.2 Qt Designer and Python code . . . 713
10.6 NGCGUI . . . 715
10.6.1 Descripción general . . . 715
10.6.2 Demonstration Configurations . . . 716
10.6.3 Library Locations . . . 718
10.6.4 Standalone Usage . . . 719
10.6.4.1 Standalone NGCGUI . . . 719
10.6.4.2 Standalone PyNGCGUI . . . 719
10.6.5 Embedding NGCGUI . . . 720
10.6.5.1 Embedding NGCGUI in AXIS . . . 720
10.6.5.2 Embedding PyNGCGUI as a GladeVCP tab page in a GUI . . . 721
10.6.5.3 Additional INI File items required for NCGUI or PyNGCGUI . . . 721
10.6.5.4 Truetype Tracer . . . 723
10.6.5.5 INI File Path Specifications . . . 723
10.6.5.6 Summary of INI File item details for NGCGUI usage . . . 724
10.6.6 File Requirements for NGCGUI Compatibility . . . 726
10.6.6.1 Single-File Gcode (.ngc) Subroutine Requirements . . . 726
10.6.6.2 Gcode-meta-compiler (.gcmc) file requirements . . . 728
10.6.7 DB25 Example . . . 730
10.6.8 Creating a subroutine . . . 732
10.7 GUI TkLinuxCNC . . . 733
10.7.1 Introducción . . . 733
10.7.2 Primeros pasos . . . 733
10.7.2.1 Una sesión típica con TkLinuxCNC . . . 733
10.7.3 Elementos de la ventana de TkLinuxCNC . . . 734
10.7.3.1 Botones principales . . . 734
10.7.3.2 Barra de estado de offsets . . . 735

10.7.3.3 Área de visualización de coordenadas . . . 735
10.7.3.4 Intérprete TkLinuxCNC / Control de programa automático . . . 735
10.7.3.5 Control manual . . . 735
10.7.3.6 Entrada de código . . . 736
10.7.3.7 Velocidad de trote . . . 737
10.7.3.8 Porcentaje de alimentación . . . 737
10.7.3.9 Porcentaje de velocidad del husillo . . . 737
10.7.4 Controles del teclado . . . 737
10.8 QtPlasmaC . . . 738
10.8.1 Preamble . . . 738
10.8.2 License . . . 738
10.8.3 Introducción . . . 738
10.8.4 Installing LinuxCNC . . . 741
10.8.4.1 If The User Does Not Have Linux Installed . . . 742
10.8.4.2 Package Installation (Buildbot) If The User Has Linux on Debian 12 (Bookworm) . . . 742
10.8.4.3 Package Installation (Buildbot) If The User Has Linux on Debian 12 (Bookworm) or Debian 11 (Bullseye) . . . 742
10.8.4.4 Run In Place Installation If The User Has Linux Installed . . . 742
10.8.5 Creating A QtPlasmaC Configuration . . . 742
10.8.5.1 Modes . . . 742
10.8.5.2 Available I/Os . . . 743
10.8.5.3 Configuración recomendada: . . . 744
10.8.5.4 Configuring . . . 745
10.8.5.5 Qt Dependency Errors . . . 750
10.8.5.6 Initial Setup . . . 750
10.8.6 Migrating to QtPlasmaC From PlasmaC (AXIS or GMOCCAPY) . . . 754
10.8.7 Other QtPlasmaC Setup Considerations . . . 754
10.8.7.1 Low-pass Filter . . . 754
10.8.7.2 Contact Bounce . . . 754
10.8.7.3 Contact Load . . . 755
10.8.7.4 Desktop Launcher . . . 756
10.8.7.5 QtPlasmaC Files . . . 756
10.8.7.6 Archivo INI . . . 757
10.8.8 QtPlasmaC GUI Overview . . . 759
10.8.8.1 Exiting QtPlasmaC . . . 759
10.8.8.2 MAIN Tab . . . 759
10.8.8.3 Preview Views . . . 766
10.8.8.4 CONVERSATIONAL Tab . . . 766

10.8.8.5 PARAMETERS Tab . . . 767
10.8.8.6 SETTINGS Tab . . . 773
10.8.8.7 STATISTICS Tab . . . 777
10.8.9 Using QtPlasmaC . . . 777
10.8.9.1 Units Systems . . . 778
10.8.9.2 Preamble and Postamble Codes . . . 778
10.8.9.3 Mandatory Codes . . . 779
10.8.9.4 Coordinates . . . 779
10.8.9.5 Cut Feed Rate . . . 779
10.8.9.6 Material File . . . 779
10.8.9.7 Manual Material Handling . . . 781
10.8.9.8 Automatic Material Handling . . . 781
10.8.9.9 Material Addition Via Magic Comments In G-code . . . 782
10.8.9.10 Material Converter . . . 783
10.8.9.11 LASER . . . 786
10.8.9.12 CAMERA . . . 788
10.8.9.13 Path Tolerance . . . 789
10.8.9.14 Paused Motion . . . 790
10.8.9.15 Pause At End Of Cut . . . 790
10.8.9.16 Multiple Tools . . . 790
10.8.9.17 Velocity Reduction . . . 791
10.8.9.18 THC (Torch Height Controller) . . . 792
10.8.9.19 Cutter Compensation . . . 793
10.8.9.20 Initial Height Sense (IHS) Skip . . . 793
10.8.9.21 Probing . . . 794
10.8.9.22 Offset Probing . . . 794
10.8.9.23 Cut Types . . . 795
10.8.9.24 Hole Cutting - Intro . . . 796
10.8.9.25 Hole Cutting . . . 796
10.8.9.26 Hole Cutting - Automatic . . . 798
10.8.9.27 Single Cut . . . 799
10.8.9.28 Thick Materials . . . 801
10.8.9.29 Mesh Mode (Expanded Metal Cutting) . . . 801
10.8.9.30 Ignore Arc OK . . . 802
10.8.9.31 Cut Recovery . . . 802
10.8.9.32 Run From Line . . . 803
10.8.9.33 Scribe . . . 805
10.8.9.34 Spotting . . . 807
10.8.9.35 Tube Cutting . . . 808

10.8.9.36 Virtual Keyboard Custom Layouts . . . 808
10.8.9.37 Keyboard Shortcuts . . . 809
10.8.9.38 MDI . . . 810
10.8.10 Conversational Shape Library . . . 811
10.8.10.1 Conversational Settings . . . 813
10.8.10.2 Conversational Lines And Arcs . . . 813
10.8.10.3 Conversational Single Shape . . . 814
10.8.10.4 Conversational Group Of Shapes . . . 816
10.8.10.5 Conversational Block . . . 816
10.8.10.6 Conversational Saving A Job . . . 817
10.8.11 Mensajes de error . . . 818
10.8.11.1 Error Logging . . . 818
10.8.11.2 Error Message Display . . . 818
10.8.11.3 Critical Errors . . . 818
10.8.11.4 Warning Messages . . . 820
10.8.12 Updating QtPlasmaC . . . 821
10.8.12.1 Standard Update . . . 821
10.8.12.2 Continuous Update . . . 821
10.8.13 Modify An Existing QtPlasmaC Configuration . . . 821
10.8.14 Customizing QtPlasmaC GUI . . . 821
10.8.14.1 Add A Custom Style . . . 822
10.8.14.2 Create A New Style . . . 822
10.8.14.3 Returning To The Default Styling . . . 823
10.8.14.4 Custom Python Code . . . 823
10.8.14.5 Custom G-code Filter . . . 824
10.8.15 QtPlasmaC Advanced Topics . . . 825
10.8.15.1 Custom User Buttons . . . 825
10.8.15.2 Peripheral Offsets (Laser, Camera, Scribe, Offset Probe) . . . 832
10.8.15.3 Keep Z Motion . . . 834
10.8.15.4 External HAL Pins . . . 834
10.8.15.5 Hide Program Buttons . . . 836
10.8.15.6 Tuning Mode 0 Arc OK . . . 836
10.8.15.7 Lost Arc Delay . . . 836
10.8.15.8 Zero Window . . . 837
10.8.15.9 Tuning Void Sensing . . . 837
10.8.15.10 Max Offset . . . 837
10.8.15.11 Enable Tabs During Automated Motion . . . 838
10.8.15.12 Override Jog Inhibit Via Z+ Jog . . . 838
10.8.15.13 QtPlasmaC State Outputs . . . 838

10.8.15.04 QtPlasmaC Debug Print . . . 839
10.8.15.05 Hypertherm PowerMax Communications . . . 839
10.8.15.06 Moving Pierce . . . 840
10.8.16 Internationalisation . . . 843
10.8.17 Appendix . . . 845
10.8.17.1 Example Configurations . . . 845
10.8.17.2 NGC Samples . . . 845
10.8.17.3 QtPlasmaC Specific G-codes . . . 845
10.8.17.4 QtPlasmaC G-code Examples . . . 847
10.8.17.5 Mesa THCAD . . . 849
10.8.17.6 RS485 Connections . . . 850
10.8.17.7 Arc OK With A Reed Relay . . . 853
10.8.17.8 Contact Load Schematics . . . 855
10.8.18 Known Issues . . . 855
10.8.18.1 Keyboard Jogging . . . 855
10.8.18.2 NO_FORCE_HOMING . . . 856
10.8.19 Contributing Code To QtPlasmaC . . . 856
10.8.20 Support . . . 857
10.9 MDRO GUI . . . 857
10.9.1 Introducción . . . 857
10.9.2 Primeros pasos . . . 858
10.9.2.1 INI File Options . . . 858
10.9.2.2 Command Line Options . . . 859
10.9.2.3 Pines . . . 859
10.9.3 MDRO Window . . . 859
10.9.4 Index operations . . . 860
10.9.5 Simulation . . . 860

11 Programación en G-Code 861
11.1 Sistemas de coordenadas . . . 861
11.1.1 Introducción . . . 861
11.1.2 Sistema de coordenadas de la máquina . . . 861
11.1.2.1 Movimientos en coordenadas de máquina: G53 . . . 861
11.1.3 Sistemas de coordenadas . . . 862
11.1.3.1 Sistema de coordenadas predeterminado . . . 864
11.1.3.2 Configuración de offsets de sistema de coordenadas . . . 864
11.1.4 Offsets locales y globales . . . 864
11.1.4.1 El comando G52 . . . 864
11.1.5 G92 Offsets de ejes . . . 865

11.1.5.1 Los comandos G92 . . . 865
11.1.5.2 Configuración de valores G92 . . . 866
11.1.5.3 Precauciones de persistencia G92 . . . 866
11.1.5.4 Precauciones de interacción G92 y G52 . . . 867
11.1.6 Programas de muestra usando offsets . . . 867
11.1.6.1 Programa de muestra utilizando offsets de coordenadas de pieza de trabajo 867
11.1.6.2 Programa de muestra usando offsets G52 . . . 869
11.2 Compensación de herramienta . . . 869
11.2.1 Touch off . . . 869
11.2.1.1 Usando G10 L1/L10/L11 . . . 869
11.2.2 Tabla de herramientas . . . 870
11.2.2.1 Formato de tabla de herramientas . . . 870
11.2.2.2 E/S de herramientas . . . 872
11.2.2.3 Cambiadores de herramienta . . . 873
11.2.3 Compensación de longitud de herramienta . . . 874
11.2.4 Compensación de radio de cortador . . . 875
11.2.4.1 Descripción general . . . 876
11.2.4.2 Ejemplos . . . 878
11.3 Tool Edit GUI . . . 879
11.3.1 Descripción general . . . 879
11.3.2 Column Sorting . . . 880
11.3.3 Columns Selection . . . 881
11.3.4 Stand Alone Use . . . 881
11.4 Overview of G-Code Programming . . . 882
11.4.1 Descripción general . . . 882
11.4.2 Format of a line . . . 883
11.4.2.1 /: Block Delete . . . 883
11.4.2.2 Número de línea opcional . . . 883
11.4.2.3 Words, Parameters, Subroutines, Comments . . . 884
11.4.2.4 End of Line Marker . . . 885
11.4.3 Numbers . . . 885
11.4.4 Parámetros . . . 885
11.4.4.1 Numbered Parameters . . . 886
11.4.4.2 Subroutine Codes and Parameters . . . 888
11.4.4.3 Named Parameters . . . 888
11.4.4.4 Predefined Named Parameters . . . 889
11.4.4.5 System Parameters . . . 891
11.4.5 HAL pins and INI values . . . 892
11.4.6 Expressions . . . 893

11.4.7 Binary Operators . . . 893
11.4.8 Equality and floating-point values . . . 893
11.4.9 Funciones . . . 893
11.4.10 Repeated Items . . . 894
11.4.11 Item order . . . 895
11.4.12 Commands and Machine Modes . . . 895
11.4.13 Polar Coordinates . . . 895
11.4.14 Modal Groups . . . 897
11.4.15 Comentarios . . . 898
11.4.16 Messages . . . 899
11.4.17 Probe Logging . . . 899
11.4.18 Logging . . . 899
11.4.19 Abort Messages . . . 900
11.4.20 Debug Messages . . . 900
11.4.21 Print Messages . . . 900
11.4.22 Comment Parameters . . . 900
11.4.23 File Requirements . . . 901
11.4.24 File Size . . . 901
11.4.25 G-code Order of Execution . . . 901
11.4.26 G-code Best Practices . . . 902
11.4.27 Linear and Rotary Axis . . . 903
11.4.28 Common Error Messages . . . 903
11.5 G-Codes . . . 903
11.5.1 Conventions . . . 903
11.5.2 G-Code Quick Reference Table . . . 904
11.5.3 G0 Rapid Move . . . 905
11.5.3.1 Rapid Velocity Rate . . . 905
11.5.4 G1 Linear Move . . . 906
11.5.5 G2, G3 Arc Move . . . 906
11.5.5.1 Center Format Arcs . . . 907
11.5.5.2 Center Format Examples . . . 909
11.5.5.3 Radius Format Arcs . . . 910
11.5.6 G4 Espera . . . 911
11.5.7 G5 Cubic Spline . . . 911
11.5.8 G5.1 Quadratic Spline . . . 912
11.5.9 G5.2 G5.3 NURBS Block . . . 913
11.5.10 G7 Lathe Diameter Mode . . . 914
11.5.11 G8 Lathe Radius Mode . . . 914
11.5.12 G10 L0 Reload Tool Table Data . . . 915

11.5.15 G10 L1 Set Tool Table . . . 915
11.5.16 G10 L2 Set Coordinate System . . . 916
11.5.16 G10 L10 Set Tool Table . . . 917
11.5.16 G10 L11 Set Tool Table . . . 918
11.5.17 G10 L20 Set Coordinate System . . . 918
11.5.18 G17 - G19.1 Plane Select . . . 919
11.5.19 G20, G21 Units . . . 919
11.5.20 G28, G28.1 Go/Set Predefined Position . . . 919
11.5.21 G30, G30.1 Go/Set Predefined Position . . . 920
11.5.22 G33 Spindle Synchronized Motion . . . 920
11.5.23 G33.1 Rigid Tapping . . . 921
11.5.24 G38.*n* Straight Probe . . . 922
11.5.25 G40 Compensation Off . . . 924
11.5.26 G41, G42 Cutter Compensation . . . 924
11.5.27 G41.1, G42.1 Dynamic Cutter Compensation . . . 925
11.5.28 G43 Tool Length Offset . . . 925
11.5.29 G43.1 Dynamic Tool Length Offset . . . 926
11.5.30 G43.2 Apply additional Tool Length Offset . . . 927
11.5.31 G49 Cancel Tool Length Compensation . . . 927
11.5.32 G52 Local Coordinate System Offset . . . 928
11.5.33 G53 Move in Machine Coordinates . . . 928
11.5.34 G54-G59.3 Select Coordinate System . . . 928
11.5.35 G61 Exact Path Mode . . . 929
11.5.36 G61.1 Exact Stop Mode . . . 929
11.5.37 G64 Path Blending . . . 929
11.5.38 G70 Lathe finishing cycle . . . 932
11.5.39 G71 G72 Lathe roughing cycles . . . 933
11.5.40 G73 Drilling Cycle with Chip Breaking . . . 934
11.5.41 G74 Left-hand Tapping Cycle with Dwell . . . 935
11.5.42 G76 Threading Cycle . . . 936
11.5.43 G80-G89 Canned Cycles . . . 938
11.5.43.1 Common Words . . . 939
11.5.43.2 Sticky Words . . . 939
11.5.43.3 Repeat Cycle . . . 939
11.5.43.4 Retract Mode . . . 939
11.5.43.5 Canned Cycle Errors . . . 939
11.5.43.6 Preliminary and In-Between Motion . . . 940
11.5.43.7 Why use a canned cycle? . . . 940
11.5.44 G80 Cancel Canned Cycle . . . 942

11.5.45 G81 Drilling Cycle . . . 943
11.5.46 G82 Drilling Cycle, Dwell . . . 948
11.5.47 G83 Peck Drilling Cycle . . . 948
11.5.48 G84 Right-hand Tapping Cycle, Dwell . . . 949
11.5.49 G85 Boring Cycle, Feed Out . . . 949
11.5.50 G86 Boring Cycle, Spindle Stop, Rapid Move Out . . . 950
11.5.51 G87 Back Boring Cycle . . . 950
11.5.52 G88 Boring Cycle, Spindle Stop, Manual Out . . . 950
11.5.53 G89 Boring Cycle, Dwell, Feed Out . . . 950
11.5.54 G90, G91 Distance Mode . . . 951
11.5.55 G90.1, G91.1 Arc Distance Mode . . . 951
11.5.56 G92 Coordinate System Offset . . . 951
11.5.57 G92.1, G92.2 Reset G92 Offsets . . . 952
11.5.58 G92.3 Restore G92 Offsets . . . 952
11.5.59 G93, G94, G95 Feed Rate Mode . . . 952
11.5.60 G96, G97 Spindle Control Mode . . . 953
11.5.61 G98, G99 Canned Cycle Return Level . . . 954
11.6 M-Codes . . . 954
11.6.1 M-Code Quick Reference Table . . . 954
11.6.2 M0, M1 Program Pause . . . 955
11.6.3 M2, M30 Program End . . . 955
11.6.4 M60 Pallet Change Pause . . . 956
11.6.5 M3, M4, M5 Spindle Control . . . 956
11.6.6 M6 Tool Change . . . 956
11.6.6.1 Cambio manual de herramienta . . . 956
11.6.6.2 Tool Changer . . . 956
11.6.7 M7, M8, M9 Coolant Control . . . 957
11.6.8 M19 Orient Spindle . . . 957
11.6.9 M48, M49 Speed and Feed Override Control . . . 958
11.6.10 M50 Feed Override Control . . . 958
11.6.11 M51 Spindle Speed Override Control . . . 958
11.6.12 M52 Adaptive Feed Control . . . 958
11.6.13 M53 Feed Stop Control . . . 959
11.6.14 M61 Set Current Tool . . . 959
11.6.15 M62 - M65 Digital Output Control . . . 959
11.6.16 M66 Wait on Input . . . 960
11.6.17 M67 Analog Output, Synchronized . . . 960
11.6.18 M68 Analog Output, Immediate . . . 961
11.6.19 M70 Save Modal State . . . 961

11.6.20 M71 Invalidate Stored Modal State . . . 962
11.6.21 M72 Restore Modal State . . . 963
11.6.22 M73 Save and Autorestore Modal State . . . 963
11.6.23 M98 and M99 . . . 964
11.6.23.1 Selectively Restoring Modal State . . . 964
11.6.24 M100-M199 Comandos definidos por el usuario . . . 965
11.7 O Codes . . . 966
11.7.1 Use of O-codes . . . 966
11.7.2 Numbering . . . 967
11.7.3 Comentarios . . . 967
11.7.4 Subroutines . . . 968
11.7.4.1 Fanuc-Style Numbered Programs . . . 969
11.7.5 Looping . . . 970
11.7.6 Conditional . . . 971
11.7.7 Repeat . . . 972
11.7.8 Indirection . . . 972
11.7.9 Calling Files . . . 973
11.7.10 Subroutine return values . . . 973
11.7.11 Errors . . . 973
11.8 Other Codes . . . 974
11.8.1 F: Set Feed Rate . . . 974
11.8.2 S: Set Spindle Speed . . . 974
11.8.3 T: Seleccionar herramienta . . . 974
11.9 G-Code Examples . . . 975
11.9.1 Mill Examples . . . 975
11.9.1.1 Helical Hole Milling . . . 975
11.9.1.2 Slotting . . . 975
11.9.1.3 Grid Probe . . . 976
11.9.1.4 Smart Probe . . . 977
11.9.1.5 Tool Length Probe . . . 978
11.9.1.6 Hole Probe . . . 978
11.9.1.7 Cutter Compensation . . . 978
11.9.2 Lathe Examples . . . 979
11.9.2.1 Roscado . . . 979
11.10 Image to G-Code . . . 979
11.10.1 What is a depth map? . . . 979
11.10.2 Integrating image-to-gcode with the AXIS user interface . . . 980
11.10.3 Using `image-to-gcode` . . . 980
11.10.4 Option Reference . . . 980

11.10.4.1 Unidades . . . 980
11.10.4.2 Invert Image . . . 980
11.10.4.3 Normalize Image . . . 980
11.10.4.4 Expand Image Border . . . 980
11.10.4.5 Tolerance (units) . . . 980
11.10.4.6 Pixel Size (units) . . . 981
11.10.4.7 Velocidad de alimentación por inmersión (unidades por minuto) . . . 981
11.10.4.8 Velocidad de alimentación (unidades por minuto) . . . 981
11.10.4.9 Velocidad del husillo (RPM) . . . 981
11.10.4.10 Scan Pattern . . . 981
11.10.4.11 Scan Direction . . . 981
11.10.4.12 Depth (units) . . . 981
11.10.4.13 Step Over (pixels) . . . 981
11.10.4.14 Tool Diameter . . . 982
11.10.4.15 Safety Height . . . 982
11.10.4.16 Tool Type . . . 982
11.10.4.17 Face bounding . . . 982
11.10.4.18 Contact angle . . . 982
11.10.4.19 Roughing offset and depth per pass . . . 982
11.11 RS274/NGC Differences . . . 983
11.11.1 Changes from RS274/NGC . . . 983
11.11.2 Additions to RS274/NGC . . . 984

12 Paneles Virtuales de Control 985

12.1 PyVCP . . . 985
12.1.1 Introducción . . . 985
12.1.2 Panel Construction . . . 986
12.1.3 Security . . . 987
12.1.4 AXIS . . . 987
12.1.4.1 Panel de ejemplo . . . 987
12.1.5 Stand Alone . . . 989
12.1.6 Widgets . . . 990
12.1.6.1 Sintaxis . . . 990
12.1.6.2 General Notes . . . 990
12.1.6.3 Label . . . 991
12.1.6.4 Multi_Label . . . 992
12.1.6.5 LEDs . . . 992
12.1.6.6 Buttons . . . 993
12.1.6.7 Number Displays . . . 995

12.1.6.8 Number Inputs . . . 998
12.1.6.9 Images . . . 1002
12.1.6.10 Containers . . . 1004
12.2 PyVCP Examples . . . 1009
12.2.1 AXIS . . . 1009
12.2.2 Floating Panels . . . 1009
12.2.3 Jog Buttons Example . . . 1010
12.2.3.1 Create the Widgets . . . 1011
12.2.3.2 Hacer conexiones . . . 1013
12.2.4 Port Tester . . . 1014
12.2.5 GS2 RPM Meter . . . 1017
12.2.5.1 The Panel . . . 1017
12.2.5.2 The Connections . . . 1019
12.2.6 Rapid to Home Button . . . 1019
12.3 GladeVCP: Glade Virtual Control Panel . . . 1021
12.3.1 ¿Qué es GladeVCP? . . . 1021
12.3.1.1 PyVCP versus GladeVCP at a glance . . . 1021
12.3.2 A Quick Tour with the Example Panel . . . 1022
12.3.2.1 Exploring the example panel . . . 1025
12.3.2.2 Exploring the User Interface description . . . 1026
12.3.2.3 Exploring the Python callback . . . 1026
12.3.3 Creating and Integrating a Glade user interface . . . 1026
12.3.3.1 Prerequisite: Glade installation . . . 1026
12.3.3.2 Running Glade to create a new user interface . . . 1026
12.3.3.3 Testing a panel . . . 1027
12.3.3.4 Preparing the HAL command file . . . 1028
12.3.3.5 Integrating into AXIS, like PyVCP . . . 1028
12.3.3.6 Embedding as a Tab . . . 1029
12.3.3.7 Integrating into Touchy . . . 1029
12.3.3.8 Loading Builtin Panels . . . 1030
12.3.4 GladeVCP command line options . . . 1030
12.3.5 Understanding the GladeVCP startup process . . . 1031
12.3.6 HAL Widget reference . . . 1032
12.3.6.1 Widget and HAL pin naming . . . 1032
12.3.6.2 Python attributes and methods of HAL Widgets . . . 1033
12.3.6.3 Setting pin and widget values . . . 1033
12.3.6.4 The hal-pin-changed signal . . . 1034
12.3.6.5 Buttons . . . 1034
12.3.6.6 Scales . . . 1035

12.3.6.7 SpinButton . . . 1036
12.3.6.8 Hal_Dial . . . 1036
12.3.6.9 Jog Wheel . . . 1038
12.3.6.10 Speed Control . . . 1040
12.3.6.11 Label . . . 1042
12.3.6.12 Containers . . . 1042
12.3.6.13 LED . . . 1043
12.3.6.14 ProgressBar . . . 1044
12.3.6.15 ComboBox . . . 1045
12.3.6.16 Bars . . . 1045
12.3.6.17 Meter . . . 1047
12.3.6.18 HAL_Graph . . . 1048
12.3.6.19 Gremlin tool path preview for NGC files . . . 1048
12.3.6.20 HAL_Offset . . . 1051
12.3.6.21 DRO widget . . . 1051
12.3.6.22 Combi_DRO widget . . . 1054
12.3.6.23 IconView (File Select) . . . 1057
12.3.6.24 Calculator widget . . . 1060
12.3.6.25 Tooleditor widget . . . 1060
12.3.6.26 Offsetpage . . . 1062
12.3.6.27 HAL_sourceview widget . . . 1064
12.3.6.28 MDI history . . . 1065
12.3.6.29 Animated function diagrams: HAL widgets in a bitmap . . . 1066
12.3.7 Action Widgets Reference . . . 1066
12.3.7.1 VCP Action Widgets . . . 1067
12.3.7.2 VCP Action Python . . . 1067
12.3.7.3 VCP ToggleAction widgets . . . 1068
12.3.7.4 The Action_MDI Toggle and Action_MDI widgets . . . 1069
12.3.7.5 A simple example: Execute MDI command on button press . . . 1069
12.3.7.6 Parameter passing with Action_MDI and ToggleAction_MDI widgets . . . 1069
12.3.7.7 An advanced example: Feeding parameters to an O-word subroutine . . . 1070
12.3.7.8 Preparing for an MDI Action, and cleaning up afterwards . . . 1070
12.3.7.9 Using the LinuxCNC Stat object to deal with status changes . . . 1071
12.3.8 GladeVCP Programming . . . 1072
12.3.8.1 User Defined Actions . . . 1072
12.3.8.2 Core Library . . . 1072
12.3.8.3 An example: adding custom user callbacks in Python . . . 1073
12.3.8.4 HAL value change events . . . 1073
12.3.8.5 Programming model . . . 1073

12.3.8.6 Initialization sequence . . . 1075
12.3.8.7 Multiple callbacks with the same name . . . 1076
12.3.8.8 The GladeVCP `-U <useropts>` flag . . . 1076
12.3.8.9 Persistent variables in GladeVCP . . . 1076
12.3.8.10 Using persistent variables . . . 1076
12.3.8.11 Saving the state on GladeVCP shutdown . . . 1077
12.3.8.12 Saving state when Ctrl-C is pressed . . . 1078
12.3.8.13 Hand-editing INI (.ini) files . . . 1078
12.3.8.14 Adding HAL pins . . . 1078
12.3.8.15 Adding timers . . . 1079
12.3.8.16 Setting HAL widget properties programmatically . . . 1079
12.3.8.17 Value-changed callback with hal_glib . . . 1080
12.3.8.18 Examples, and rolling your own GladeVCP application . . . 1080
12.3.9 FAQ . . . 1080
12.3.10 Troubleshooting . . . 1081
12.3.11 Implementation note: Key handling in AXIS . . . 1081
12.3.12 Adding Custom Widgets . . . 1082
12.3.13 Aplicaciones auxiliares GladeVCP . . . 1082
12.4 GladeVCP Library modules . . . 1083
12.4.1 Info . . . 1083
12.4.2 Action . . . 1085
12.5 QtVCP . . . 1087
12.5.1 Showcase . . . 1087
12.5.2 Descripción general . . . 1093
12.5.2.1 QtVCP Widgets . . . 1094
12.5.2.2 Configuración INI . . . 1094
12.5.2.3 Qt Designer UI File . . . 1095
12.5.2.4 Handler Files . . . 1095
12.5.2.5 Libraries Modules . . . 1096
12.5.2.6 Themes . . . 1096
12.5.2.7 Local Files . . . 1096
12.5.2.8 Modifying Stock Screens . . . 1097
12.5.3 VCP Panels . . . 1100
12.5.3.1 Builtin Panels . . . 1100
12.5.3.2 Custom Panels . . . 1104
12.5.4 Build A Simple Clean-sheet Custom Screen . . . 1106
12.5.4.1 Descripción general . . . 1106
12.5.4.2 Get Qt Designer To Include LinuxCNC Widgets . . . 1106
12.5.4.3 Build The Screen `.ui` File . . . 1107

12.5.4.4 Handler file . . . 1110
12.5.4.5 Configuración de INI . . . 1110
12.5.5 Handler File In Detail . . . 1110
12.5.5.1 Descripción general . . . 1111
12.5.5.2 IMPORT Section . . . 1114
12.5.5.3 INSTANTIATE LIBRARIES Section . . . 1114
12.5.5.4 HANDLER CLASS Section . . . 1114
12.5.5.5 INITIALIZE Section . . . 1114
12.5.5.6 SPECIAL FUNCTIONS Section . . . 1115
Bibliografía . . . 1116
12.5.5.7 STATUS CALLBACKS Section . . . 1116
12.5.5.8 CALLBACKS FROM FORM Section . . . 1116
12.5.5.9 GENERAL FUNCTIONS Section . . . 1116
12.5.5.10 KEY BINDING Section . . . 1116
12.5.5.11 CLOSING EVENT Section . . . 1116
12.5.6 Connecting Widgets to Python Code . . . 1116
12.5.6.1 Descripción general . . . 1117
12.5.6.2 Using Qt Designer to add Slots . . . 1117
12.5.6.3 Python Handler Changes . . . 1118
12.5.7 More Information . . . 1119
12.6 QtVCP Virtual Control Panels . . . 1119
12.6.1 Builtin Virtual Control Panels . . . 1119
12.6.1.1 copy . . . 1119
12.6.1.2 spindle_belts . . . 1120
12.6.1.3 test_dial . . . 1122
12.6.1.4 test_button . . . 1123
12.6.1.5 test_led . . . 1123
12.6.1.6 test_panel . . . 1124
12.6.1.7 cam_align . . . 1125
12.6.1.8 sim_panel . . . 1128
12.6.1.9 tool_dialog . . . 1129
12.6.2 vismach 3D Simulation Panels . . . 1129
12.6.2.1 QtVCP vismach_mill_xyz . . . 1129
12.6.2.2 QtVCP vismach_router_atc . . . 1130
12.6.2.3 QtVCP vismach_scara . . . 1131
12.6.2.4 QtVCP vismach_millturn . . . 1132
12.6.2.5 QtVCP vismach_mill_5axis_gantry . . . 1133
12.6.2.6 QtVCP vismach_fanuc_200f . . . 1134
12.6.3 Custom Virtual Control Panels . . . 1135

12.6.4 Embedding QtVCP Virtual Control Panels into QtVCP Screens . . . 1136
12.6.4.1 Embedding Commands . . . 1136
12.6.4.2 Location of builtin Panels . . . 1136
12.6.4.3 Location of Custom Panels . . . 1137
12.6.4.4 Handler Programming Tips . . . 1137
12.6.4.5 Designer Widget Tips . . . 1137
12.6.4.6 Handler Patching - Subclassing Builtin Panels . . . 1137
12.7 QtVCP Widgets . . . 1138
12.7.1 HAL Only Widgets . . . 1139
12.7.1.1 `CheckBox` Widget . . . 1139
12.7.1.2 *DetachTabWidget* - Container Widget With User Detachable Panels . . . 1139
12.7.1.3 `DoubleScale` - Spin Button Entry Widget . . . 1139
12.7.1.4 `FocusOverlay` - Focus Overlay Widget . . . 1139
12.7.1.5 `Gauge` - Round Dial Gauge Widget . . . 1141
12.7.1.6 `GeneralHALInput` - General Signals/Slots Input Connection Widget . . . 1142
12.7.1.7 `GeneralHALOutput` - General Signals/Slots Output Connection Widget . . . 1142
12.7.1.8 `GridLayout` - Grid Layout Widget . . . 1143
12.7.1.9 *HalBar* - HAL Bar Level Indicator . . . 1143
12.7.1.10 `HALPad` - HAL Buttons Joypad . . . 1144
12.7.1.11 `HALLabel` - HAL Label Widget . . . 1146
12.7.1.12 `LCDNumber` - LCD Style Number Readout Widget . . . 1147
12.7.1.13 `LED` - Indicator Widget . . . 1147
12.7.1.14 `PushButton` - HAL Pin Toggle Widget . . . 1148
12.7.1.15 `RadioButton` Widget . . . 1148
12.7.1.16 `Slider` - HAL Pin Value Adjusting Widget . . . 1148
12.7.1.17 `TabWidget` - Tab Widget . . . 1149
12.7.1.18 `WidgetSwitcher` - Multi-widget Layout View Switcher Widget . . . 1149
12.7.1.19 `XEmbed` - Program Embedding Widget . . . 1150
12.7.2 Machine Controller Widgets . . . 1150
12.7.2.1 `ActionButton` - Machine Controller Action Control Widget . . . 1150
12.7.2.2 `ActionToolButton` - Optional Actions Menu Button Widget . . . 1153
12.7.2.3 `AxisToolButton` - Select and Set Axis Widget . . . 1153
12.7.2.4 `BasicProbe` - Simple Mill Probing Widget . . . 1155
12.7.2.5 `CamView` - Workpiece Alignment and Origin Setting Widget . . . 1155
12.7.2.6 `DROLabel` - Axis Position Display Widget . . . 1155
12.7.2.7 `FileManager` - File Loading Selector Widget . . . 1158
12.7.2.8 `GcodeDisplay` - G-code Text Display Widget . . . 1159
12.7.2.9 `GcodeEditor` - G-code Program Editor Widget . . . 1160
12.7.2.10 `GCodeGraphics` - G-code Graphic Backplot Widget . . . 1161

12.7.2.11 JointEnableWidget - FIXME . . . 1165
12.7.2.12 JogIncrements - Jog Increments Value Selection Widget . . . 1165
12.7.2.13 MacroTab - Special Macros Widget . . . 1166
12.7.2.14 OperatorValueLine - Operator Value Line Entry Widget . . . 1168
12.7.2.15 MDILine - MDI Commands Line Entry Widget . . . 1169
12.7.2.16 MDIHistory - MDI Commands History Widget . . . 1170
12.7.2.17 MDITouchy - Touch Screen MDI Entry Widget . . . 1171
12.7.2.18 OriginOffsetView - Origins View and Setting Widget . . . 1173
12.7.2.19 RadioAxisSelector - FIXME . . . 1174
12.7.2.20 RoundButton - Round Shapped ActionButton Widget . . . 1174
12.7.2.21 StateLabel - Controller Modes State Label Display Widget . . . 1174
12.7.2.22 StatusLabel - Controller Variables State Label Display Widget . . . 1175
12.7.2.23 StatusImageSwitcher - Controller Status Image Switcher . . . 1177
12.7.2.24 StatusStacked - Mode Status Display Switching Widget . . . 1179
12.7.2.25 ScreenOption - General Options Setting widget . . . 1179
12.7.2.26 StatusSlider - Controller Setting Adjustment Slider Widget . . . 1184
12.7.2.27 StateLED - Controller State LED Widget . . . 1185
12.7.2.28 StatusAdjustmentBar - Controller Value Setting Widget . . . 1186
12.7.2.29 SystemToolButton - User System Selection Widget . . . 1187
12.7.2.30 StateEnableGridlayout - Controller State Enabled Container Widget . . . 1187
12.7.2.31 StatusImageSwitcher - Controller Status Image Switching Widget . . . 1187
12.7.2.32 ToolOffsetView - Tools Offsets View And Edit Widget . . . 1188
12.7.2.33 VersaProbe - Mill Probing Widget . . . 1190
12.7.3 Dialog Widgets . . . 1190
12.7.3.1 LcncDialog - General Message Dialog Widget . . . 1191
12.7.3.2 ToolDialog - Manual Tool Change Dialog Widget . . . 1192
12.7.3.3 FileDialog - Load and Save File Chooser Dialog Widget . . . 1193
12.7.3.4 OriginOffsetDialog - Origin Offset Setting Dialog Widget . . . 1194
12.7.3.5 ToolOffsetDialog - Tool Offset Setting Dialog Widget . . . 1195
12.7.3.6 ToolChooserDialog - Tool Chooser Dialog Widget . . . 1195
12.7.3.7 MachineLog - Machine Events Journal Display Widget . . . 1196
12.7.3.8 MacroTabDialog - Macro Launch Dialog Widget . . . 1197
12.7.3.9 CamViewDialog - WebCam Part Alignment Dialog Widget . . . 1197
12.7.3.10 EntryDialog - Edit Line Dialog Widget . . . 1197
12.7.3.11 CalculatorDialog - Calculator Dialog Widget . . . 1197
12.7.3.12 RunFromLine - Run-From-Line Dialog Widget . . . 1199
12.7.3.13 VersaProbeDialog - Part Touch Probing Dialog Widget . . . 1200
12.7.3.14 MachineLogDialog - Machine and Debugging Logs Dialog Widget . . . 1201
12.7.4 Other Widgets . . . 1201

12.7.4.1 `NurbsEditor` - NURBS Editing Widget . . . 1202
12.7.4.2 `JoyPad` - 5 button D-pad Widget . . . 1202
12.7.4.3 *WebWidget* . . . 1204
12.7.5 BaseClass/Mixin Widgets . . . 1205
12.7.5.1 `IndicatedPushButtons` . . . 1205
12.7.6 Import-Only Widgets . . . 1208
12.7.6.1 Auto Height . . . 1208
12.7.6.2 G-code Utility . . . 1208
12.7.6.3 Facing . . . 1209
12.7.6.4 Hole Circle . . . 1209
12.7.6.5 Hole Enlarge . . . 1210
12.7.6.6 Qt NGCGUI . . . 1210
12.7.6.7 Qt PDF . . . 1212
12.7.6.8 Qt Vismach . . . 1212
12.7.6.9 Hal Selection Box . . . 1212
12.8 QtVCP Libraries modules . . . 1212
12.8.1 `Status` . . . 1212
12.8.1.1 Utilización . . . 1213
12.8.1.2 Ejemplo . . . 1213
12.8.2 `Info` . . . 1213
12.8.2.1 Available data and defaults . . . 1214
12.8.2.2 User message dialog info . . . 1215
12.8.2.3 Información del programa embebido . . . 1215
12.8.2.4 Helpers . . . 1215
12.8.2.5 Utilización . . . 1215
12.8.3 `Action` . . . 1216
12.8.3.1 Helpers . . . 1216
12.8.3.2 Utilización . . . 1216
12.8.4 Qhal . . . 1218
12.8.4.1 Attributes . . . 1218
12.8.4.2 Constantes . . . 1219
12.8.4.3 References . . . 1219
12.8.5 QPin . . . 1219
12.8.5.1 Signals . . . 1219
12.8.5.2 Attributes . . . 1219
12.8.5.3 References . . . 1220
12.8.5.4 Ejemplo . . . 1220
12.8.6 `Tool` . . . 1220
12.8.6.1 Helpers . . . 1220

12.8.7 Path . . . 1221
12.8.7.1 Referenced Paths . . . 1221
12.8.7.2 Helpers . . . 1223
12.8.7.3 Utilización . . . 1223
12.8.8 VCPWindow . . . 1223
12.8.8.1 Utilización . . . 1223
12.8.9 Aux_program_loader . . . 1224
12.8.9.1 Helpers . . . 1224
12.8.9.2 Utilización . . . 1224
12.8.10 Keylookup . . . 1225
12.8.10.1 Utilización . . . 1225
12.8.10.2 Key Defines . . . 1226
12.8.11 Messages . . . 1228
12.8.11.1 Properties . . . 1228
12.8.11.2 HAL Pins . . . 1229
12.8.11.3 Ejemplos . . . 1229
12.8.12 multimessages . . . 1230
12.8.12.1 Properties . . . 1230
12.8.12.2 Ejemplos . . . 1231
12.8.13 Notify . . . 1231
12.8.13.1 Properties . . . 1232
12.8.14 Preferences . . . 1232
12.8.15 Player . . . 1232
12.8.15.1 Sounds . . . 1232
12.8.15.2 Utilización . . . 1233
12.8.15.3 Ejemplo . . . 1233
12.8.16 Virtual Keyboard . . . 1233
12.8.17 Toolbar Actions . . . 1234
12.8.17.1 Actions . . . 1234
12.8.17.2 Submenus . . . 1234
12.8.17.3 Utilización . . . 1234
12.8.17.4 Ejemplos . . . 1234
12.8.18 Qt Vismach Machine Graphics library . . . 1235
12.8.18.1 Builtin Samples . . . 1235
12.8.18.2 Primitives Library . . . 1235
12.8.18.3 Utilización . . . 1237
12.8.18.4 More Information . . . 1238
12.9 QtVismach . . . 1238
12.9.1 Introducción . . . 1238

12.9.2 Hierarchy of Machine Design . . . 1240
12.9.3 Iniciar el script . . . 1241
12.9.4 Pines HAL. . . . 1241
12.9.5 Creating Parts . . . 1241
12.9.5.1 Import STL or OBJ Files . . . 1241
12.9.5.2 Build from Geometric Primitives . . . 1242
12.9.6 Moving Model Parts . . . 1243
12.9.6.1 Translating Model parts . . . 1243
12.9.6.2 Rotating Model Parts . . . 1243
12.9.7 Animating Parts . . . 1243
12.9.7.1 `HalTranslate` . . . 1243
12.9.7.2 `HalRotate` . . . 1244
12.9.7.3 *HalToolCylinder* . . . 1244
12.9.7.4 *HalToolTriangle* . . . 1244
12.9.7.5 HAL Adjustable Primitives . . . 1244
12.9.8 Assembling the model . . . 1245
12.9.9 Other functions . . . 1246
12.9.9.1 Color . . . 1246
12.9.9.2 HALColorFlip . . . 1246
12.9.9.3 HALColorRGB . . . 1246
12.9.9.4 Heads Up Display . . . 1247
12.9.9.5 HAL Heads Up Display . . . 1247
12.9.9.6 HideCollection . . . 1247
12.9.9.7 Plot Color Based on Motion Type . . . 1248
12.9.9.8 Capture . . . 1248
12.9.9.9 main . . . 1248
12.9.10 Tips . . . 1249
12.9.11 Basic structure of a `QtVismach` script . . . 1249
12.9.12 Builtin Vismach Sample Panels . . . 1250
12.10 QtVCP: Building Custom Widgets . . . 1250
12.10.1 Descripción general . . . 1250
12.10.1.1 Widgets . . . 1250
12.10.1.2 Qt Designer . . . 1251
12.10.1.3 Initialization Process . . . 1251
12.10.1.4 Proceso de limpieza . . . 1252
12.10.2 Custom HAL Widgets . . . 1252
12.10.3 Custom Controller Widgets Using `STATUS` . . . 1253
12.10.3.1 En la sección de importaciones . . . 1254
12.10.3.2 In The *Instantiate Libraries* Section . . . 1255

12.10.3.3 In The *Custom Widget Class Definition* Section . . . 1255
12.10.4 Custom Controller Widgets with Actions . . . 1257
12.10.5 Stylesheet Property Changes Based On Events . . . 1259
12.10.6 Use Stylesheets To Change Custom Widget Properties . . . 1260
12.10.7 Widget Plugins . . . 1260
12.10.7.1 Ejemplo de Gridlayout . . . 1261
12.10.7.2 SystemToolbutton Example . . . 1261
12.10.7.3 Making a plugin with a MenuEntry dialog box . . . 1262
12.11 QtVCP Handler File Code Snippets . . . 1265
12.11.1 Preference File Loading/Saving . . . 1265
12.11.2 Use `QSettings` To Read/Save Variables . . . 1265
12.11.3 Add A Basic Style Editor . . . 1266
12.11.4 Request Dialog Entry . . . 1266
12.11.5 Speak a Startup Greeting . . . 1267
12.11.6 ToolBar Functions . . . 1268
12.11.7 Add HAL Pins That Call Functions . . . 1269
12.11.8 Read/Write System HAL Pins Directly . . . 1269
12.11.9 Add A Special Max Velocity Slider Based On Percent . . . 1270
12.11.10 Toggle Continuous Jog On and Off . . . 1270
12.11.11 Class Patch The File Manager Widget . . . 1271
12.11.12 Adding Widgets Programmatically . . . 1273
12.11.13 Update/Read Objects Periodically . . . 1276
12.11.14 External Control With ZMQ . . . 1277
12.11.14.1 ZMQ Messages Reading . . . 1277
12.11.14.2 ZMQ Messages Writing . . . 1278
12.11.15 Sending Messages To Status Bar Or Desktop Notify Dialogs . . . 1279
12.11.16 Catch Focus Changes . . . 1280
12.11.17 Read Command Line Load Time Options . . . 1280
12.11.18 Gcode to read Qt preferences . . . 1281
12.12 QtVCP Development . . . 1282
12.12.1 Descripción general . . . 1282
12.12.2 Builtin Locations . . . 1282
12.12.3 QtVCP Startup To Shutdown . . . 1282
12.12.3.1 QtVCP Startup . . . 1283
12.12.3.2 QtVCP Shutdown . . . 1283
12.12.4 Path Information . . . 1283
12.12.5 Idiosyncrasies . . . 1284
12.12.5.1 Error Code Collecting . . . 1284
12.12.5.2 Velocidad de trote . . . 1284

12.12.5.1 Keybinding . . . 1284
12.12.5.2 Preference File . . . 1285
12.12.5.3 Widget Special Setup Functions . . . 1285
12.12.5.4 Dialogs . . . 1285
12.12.5.5 Styles (Themes) . . . 1285

13 Interfaces de usuario para Programación 1286

13.1 Panelui . . . 1286
13.1.1 Introducción . . . 1286
13.1.2 Loading Commands . . . 1286
13.1.3 panelui.ini file reference . . . 1287
13.1.4 Internal Command reference . . . 1289
13.1.5 ZMQ Messages . . . 1292
13.1.6 Handler File Extension . . . 1292
13.2 El módulo Python de LinuxCNC . . . 1293
13.2.1 Introducción . . . 1293
13.2.2 Patrones de uso de la interfaz NML LinuxCNC . . . 1294
13.2.3 Leer el estado de LinuxCNC con el módulo Python linuxcnc . . . 1294
13.2.3.1 Atributos en linuxcnc.stat . . . 1294
13.2.3.2 El diccionario `axis` . . . 1300
13.2.3.3 El diccionario `joint` . . . 1300
13.2.3.4 El diccionario `spindle` . . . 1302
13.2.4 Preparación para enviar comandos . . . 1302
13.2.5 Enviar comandos a través de `linuxcnc.command` . . . 1303
13.2.5.1 Atributos de `linuxcnc.command` . . . 1304
13.2.5.2 Métodos de `linuxcnc.command`: . . . 1304
13.2.6 Lectura del canal de error . . . 1307
13.2.7 Lectura de valores de archivo INI . . . 1308
13.2.8 El tipo `linuxcnc.positionlogger` . . . 1309
13.2.8.1 miembros . . . 1309
13.2.8.2 métodos . . . 1309
13.3 El módulo Python HAL . . . 1309
13.3.1 Uso básico . . . 1309
13.3.2 Funciones . . . 1310
13.4 GStat Python Module . . . 1313
13.4.1 Intro . . . 1313
13.4.2 Sample GStat Code . . . 1313
13.4.2.1 Sample HAL component code pattern . . . 1314
13.4.2.2 GladeVCP Python extension code pattern . . . 1314

13.4.2.3 QtVCP Python extension code pattern . . . 1315
13.4.3 Messages . . . 1316
13.4.4 Funciones . . . 1322
13.4.5 Known Issues . . . 1325
13.5 Vismach . . . 1325
13.5.1 Iniciar el script . . . 1326
13.5.2 Crea los pines HAL. . . . 1326
13.5.3 Creating Parts . . . 1326
13.5.4 Moving Parts . . . 1327
13.5.5 Animating Parts . . . 1328
13.5.6 Assembling the model. . . . 1328
13.5.7 Other functions . . . 1329
13.5.8 Basic structure of a Vismach script. . . . 1330

III Glosario, Derechos de Copia e Historia 1331

14 Overleaf 1332

15 Glosario 1333

16 Términos de Derechos de Copia 1339
16.1 Sección legal . . . 1339
16.1.1 Términos de derechos de autor . . . 1339
16.1.2 Licencia GNU de Documentation Libre . . . 1339

17 Historia de LinuxCNC 1344
17.1 Origen . . . 1344
17.1.1 Cambio de nombre . . . 1345
17.1.2 Información adicional . . . 1345

Parte I

Primeros Pasos y Configuración

Capítulo 1

Primeros Pasos con LinuxCNC

1.1. Acerca de LinuxCNC

LinuxCNC (the Enhanced Machine Control) is a software system for computer control of machine tools such as milling machines and lathes, robots such as puma and scara and other computer controlled machines up to 9 axes. LinuxCNC is free software with open source code. Current versions of LinuxCNC are entirely licensed under the GNU General Public License and Lesser GNU General Public License (GPL and LGPL).

To lower the entry-hurdle, LinuxCNC provides: * easy discovery and testing without installation with the Live Image, * easy installation from the Live Image, * easy to use graphical configuration wizards to rapidly create a configuration specific to the machine, * directly availability as regular packages of recent releases of Debian (since Bookworm) and Ubuntu (since Kinetic Kudu).

LinuxCNC provides a graphical user interface with many flavours to choose from to match your personal preferences and technical needs. Advanced users may directly exploit * graphical interface creation tools (Glade, Qt), * the interpreter for *G-code* (the RS-274 machine tool programming language), * operation of low-level machine electronics such as sensors and motor drives, * an easy to use *breadboard* layer for quickly creating a unique configuration for your machine, * un PLC en software programable con diagramas de escalera.

Under the hood, LinuxCNC provides * a realtime motion planning system with look-ahead, * support for non-Cartesian motion systems is provided via custom kinematics modules. Available architectures include hexapods (Stewart platforms and similar concepts) and systems with rotary joints to provide motion such as PUMA or SCARA robots. * support for a variety of hardware interfaces. The control can operate true servos (analog or PWM) with the feedback loop closed by the LinuxCNC software at the computer, or open loop with step-servos or stepper motors. * Motion control features include: cutter radius and length compensation, path deviation limited to a specified tolerance, lathe threading, synchronized axis motion, adaptive feedrate, operator feed override, and constant velocity control. * LinuxCNC runs on Linux using real-time extensions.

LinuxCNC expects G-code that if not entered manually is provided by another software, which supports CAM (Computer Automated Manufacturing) and determines what tool shall be used at what speed for what geometry. Many prominent CAD (Computer Automated Design) tools that determine the desired final shape of your work piece (or the assembly of multiple work pieces that area to be produced individually) offer a CAM module.

1.1.1. Architecture - Context diagram

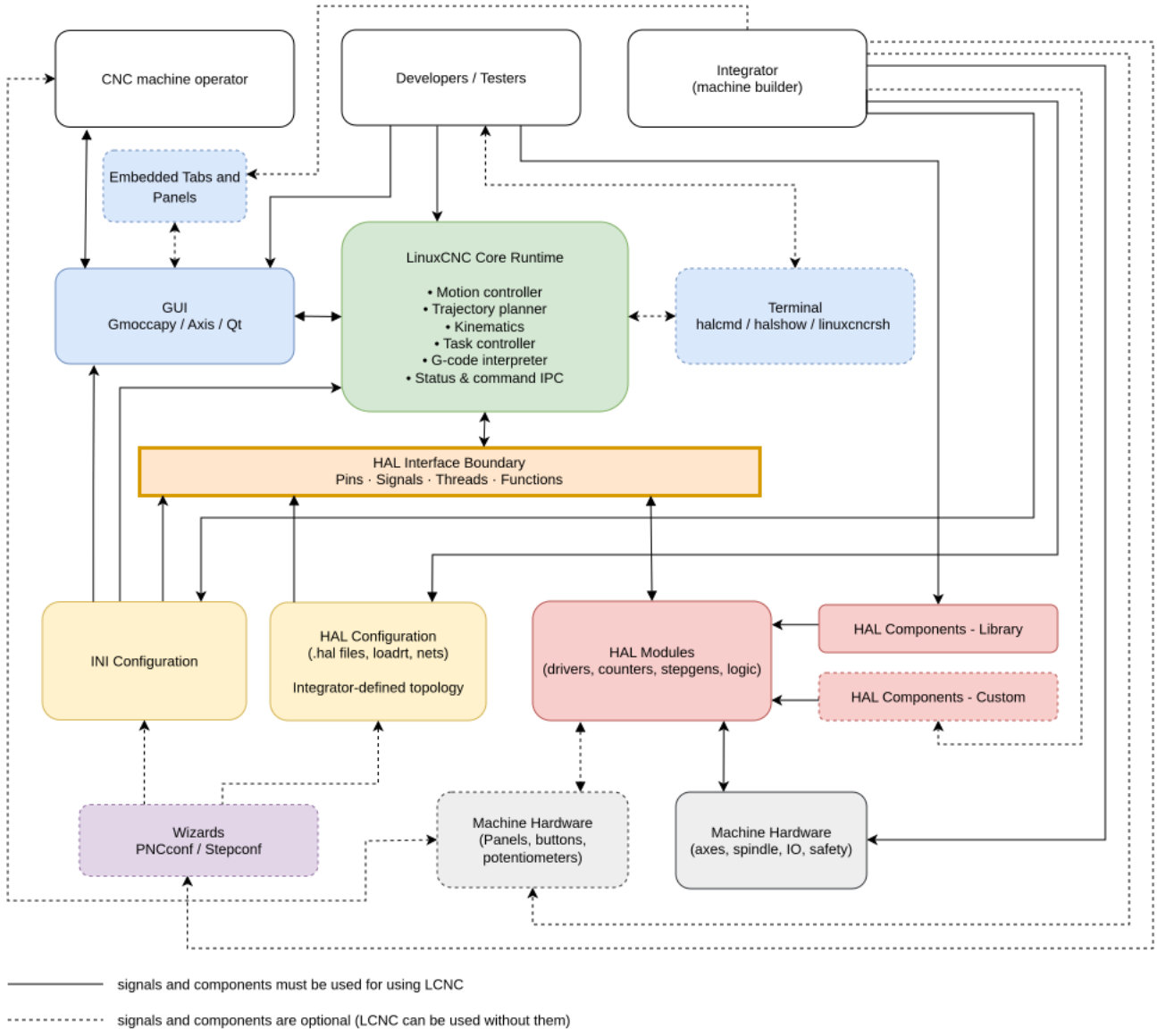


Figura 1.1: Roles of operators, integrators, developers and hardware

The diagram presents the components and players of the LinuxCNC ecosystem and how they interact. It is not intended to help you understand the functionality of LinuxCNC. Please refer to the following chapters for this.

Operator

One a machine is set up, its operator will only use one of the many graphical user interfaces that LinuxCNC and external groups are providing. The requirements for the operator are determined by how the integrator has set up the machine. The integrator has the option of setting up the machine so that the operator only presses one button to start the machining process, or leaves the GUI in its default state and the operator will fully control the CNC machine using the GUI functionality and G,M,O-codes. The integrator may or may not create a physical or virtual panel for the operator with various buttons and various indicators.

Integrator
: It is on an integrator (machine builder) to ensure that the LinuxCNC configuration matches the hardware setup both in the wiring and the protocols spoken on those wires. The integrator can choose whether to set up the machine using the Wizard or to configure it manually. If the Wizard is used, the integrator's knowledge of LinuxCNC is minimal. It is enough to understand the machine hardware. If the integrator wants to use the maximum potential of LinuxCNC, he must be able to create or edit configuration files manually. To do this, it is enough to have knowledge of HAL, INI configuration and ideally the creation of custom HAL components or embedded panels. This knowledge will allow the connection of various hardware combinations with LCNC. Using INI, the integrator selects the GUI (Gmoccapy, Axis, Qt, ...), kinematics, number of axes, parameters (velocities, acceleration, distance, ...). Using HAL, the integrator selects the hardware control method (velocity mode / position mode, on-off control / analog control, without / with feedback, ...). Using a suitable HAL module, various components can be controlled via various buses (PCI, USB, Ethernet, EtherCAT, Modbus RTU/TCP, Parallel port, ...)

Developer
: The LinuxCNC developers may be coming up with drivers for new hardware or other new features in the GUI and anything in between a mouse click and a motor turning. For testing, monitoring or possibly also the communication between multiple machines, also a text-based interface to LinuxCNC is available. Since LinuxCNC is an Open-source project, you can modify it in any way you like, provided you meet the very benevolent license conditions. You can create these modifications for the official LinuxCNC community, or for your own needs. Both paths have their advantages and disadvantages. If you offer your modification or improvement to the official developers, if they are interested, they can help you improve it even more and you will receive feedback. If you keep your modification to yourself, you do not have to worry about whether it will interest the official developers, but it may be a problem in the future if someone unfamiliar with these modifications were to maintain the machine you built (modifications, updates, fixes, ...). Of course, the developers modify all the code that is part of LinuxCNC, but the diagram only shows the links for which the developer's skills are necessary (C, C++, Python, Bash, GTK, Glade, QT, Linux OS, GitHub, PC hardware, ...)

Wizard
: Wizards are standalone programs that LinuxCNC and external groups are providing. They can work without other LinuxCNC components. The main output of Wizards are configuration files (*.ini, *.hal and others). Therefore, it is possible to do your first machine setup using the Wizard and only later, after a deeper study of the LCNC configuration, can you edit the files generated by the Wizard.

1.1.2. The Operating System

LinuxCNC is available as ready-to-use packages for Debian distributions.

1.1.3. Conseguir ayuda

1.1.3.1. Web Forum

A web forum can be found at https://forum.linuxcnc.org or by following the link at the top of the linuxcnc.org home page.

This is quite active but the demographic is more user-biased than the mailing list. If you want to be sure that your message is seen by the developers then the mailing list is to be preferred.

1.1.3.2. IRC

IRC stands for Internet Relay Chat. It is a live connection to other LinuxCNC users. The LinuxCNC IRC channel is #linuxcnc on libera.chat.

The simplest way to get on the IRC is to use the embedded web client client from libera.

Some IRC etiquette

- Ask specific questions… Avoid questions like ”Can someone help me?”.
- If you’re really new to all this, think a bit about your question before typing it. Make sure you give enough information so someone can answer your question or solve your problem.
- Tenga paciencia al esperar una respuesta; a veces lleva tiempo formular una respuesta, o todos pueden estar ocupados en el trabajo o en otra cosa.
- Configure su cuenta IRC con su nombre único para que las personas sepan quién es usted. Si usa el cliente java, use el mismo nombre cada vez que inicie sesión. Esto ayuda a las personas a recordar quién es usted y si ha estado conectado antes, muchos recordarán las discusiones pasadas, lo que ahorrará tiempo a todos.

Sharing Files

The most common way to share files on the IRC is to upload the file to one of the following or a similar service and paste the link:

- *Para texto*: https://pastebin.com/, https://gist.github.com/, https://0bin.net, https://paste.debian.net/-
- *Para imágenes*: https://imagebin.org/, https://imgur.com/, https://bayimg.com/
- *Para archivos*: https://filedropper.com/, http://filefactory.com/, http://1fichier.com/

1.1.3.3. Mailing List

An Internet Mailing List is a way to put questions out for everyone on that list to see and answer at their convenience. You get better exposure to your questions on a mailing list than on the IRC but answers take longer. In a nutshell you e-mail a message to the list and either get daily digests or individual replies back depending on how you set up your account.

Puede suscribirse a la lista de correo emc-users en: https://lists.sourceforge.net/lists/listinfo/emc-users.

1.1.3.4. Web Forum

A web forum can be found at https://forum.linuxcnc.org/ or by following the link at the top of the https://linuxcnc.org/ home page.

This is quite active but the demographic is more user-biased than the mailing list. If you want to be sure that your message is seen by the developers then the mailing list is to be preferred.

1.1.3.5. Wiki de LinuxCNC

A Wiki site is a user maintained web site that anyone can add to or edit.

El sitio Wiki LinuxCNC mantenido por usuarios contiene abundante información y consejos en: http://wiki.lin

1.1.3.6. Reportar errores

Report bugs on the LinuxCNC Github github bug tracker.

1.2. Requisitos del sistema

1.2.1. Minimum Requirements

The minimum system to run LinuxCNC and Debian / Ubuntu may vary depending on the exact usage. Stepper systems in general require faster threads to generate step pulses than servo systems. You can use the Live CD to test the software before committing to a permanent installation on a computer. Keep in mind that the Latency Test numbers are more important than the processor speed for software step generation. More information on the Latency Test is here. In addition, LinuxCNC needs to be run on an operating system that uses a specially modified kernel, see Kernel and Version Requirements.

Se puede encontrar información adicional en el sitio wiki de LinuxCNC: Requerimientos de hardware

LinuxCNC and Debian Linux should run reasonably well on a computer with the following minimum hardware specification. These numbers are not the absolute minimum but will give reasonable performance for most stepper systems.

- 1.2 GHz 64-bit x86 processor or Raspberry Pi 4 or better.
- 512 MB en RAM, 4 GB con GUI para evitar sorpresas
- No es necesario disco duro para LiveCD, 8 GB o más para una instalación permanente
- Tarjeta gráfica capaz de una resolución de al menos de 1024x768, que no use controladores propietarios NVidia o ATI fglrx. Los chipsets de gráficos integrados más modernos parecen estar bien en general.
- Conexión a internet (no es estrictamente necesaria, pero resulta muy útil para realizar actualizaciones y contactar con la comunidad de LinuxCNC)

Minimum hardware requirements change as Linux distributions evolve so check the Debian web site for details on the Live CD you're using. Older hardware may benefit from selecting an older version of the Live CD when available.

Si planeas no depender de una distribución de programas ejecutables ya preparados ("binarios") y/o pretendes contribuir al árbol de código fuente de LinuxCNC, entonces es muy probable que quieras una segunda computadora para hacer la compilación. Aunque LinuxCNC y tus desarrollos pudieran ser ejecutados al mismo tiempo con respecto a espacio en disco, RAM e incluso velocidad del procesador, una máquina atareada tendrá peores latencias, así que es improbable que quieras compilar tu código fuente y producir virutas el mismo tiempo.

1.2.2. Kernel and Version requirements

LinuxCNC requiere un kernel modificado para uso con tiempo real para controlar hardware real de máquinas. Sin embargo, puede ejecutarse en un núcleo estándar en modo de simulación para fines tales como verificar el código G, probar archivos de configuración y aprendizaje del sistema. Para trabajar con estas versiones de kernel se distribuyen dos versiones de LinuxCNC. Los nombres de los paquetes son "linuxcnc" y "linuxcnc-uspace".

The realtime kernel options are preempt-rt, RTAI and Xenomai.

Puede descubrir la versión del kernel de su sistema con el comando:

```
uname -a
```

Si ve (como arriba) `-rt-` en el nombre del kernel, entonces está ejecutando el preempt-rt kernel y debería instalar la versión "uspace" de LinuxCNC. También debe instalar uspace para configuraciones "sim" en kernels que no son en tiempo real.

If you see `-rtai-` in the kernel name then you are running RTAI realtime. See below for the LinuxCNC version to install.

1.2.2.1. Preempt-RT with *linuxcnc-uspace* package

Preempt-RT is the newest of the realtime systems, and is also the version that is closest to a mainline kernel. Preempt-RT kernels are available as precompiled packages from the main repositories. The search term ”PREEMPT_RT” will find them, and one can be downloaded and installed just like any other package. Preempt-RT will generally have the best driver support and is the only option for systems using the Mesa ethernet-connected hardware driver cards. In general preempt-rt has the worst latency of the available systems, but there are exceptions.

1.2.2.2. RTAI with *linuxcnc* package

RTAI ha sido el pilar de las distribuciones LinuxCNC durante muchos años. Generalmente dará el mejor rendimiento en tiempo real, en términos de baja latencia, pero podría tener un soporte de periféricos pobre y no tantas resoluciones de pantallas. Un Kernel RTAI está disponible desde el repositorio de paquetes de LinuxCNC. Si instaló desde la imagen Live/Install, el cambio de núcleo y sabor de LinuxCNC se describe en [Installing-RTAI].

1.2.2.3. Xenomai with *linuxcnc-uspace* package

Xenomai is also supported, but you will have to find or build the kernel and compile LinuxCNC from source to utilise it.

1.2.2.4. RTAI with *linuxcnc-uspace* package

También es posible ejecutar LinuxCNC con RTAI en modo de espacio de usuario. Como con Xenomai, necesitará compilar desde las fuentes para hacer esto.

1.2.3. Problematic Hardware

1.2.3.1. Laptops

Los computadores portátiles, en general, no son buenos para la generación de pasos por software en tiempo real. De nuevo, la prueba de latencia ejecutada por un periodo de tiempo prolongado te dará la información necesaria para determinar si resulta apropiado su uso.

1.2.3.2. Video Cards

Si tu instalación arranca con una resolución de 800 x 600, en la mayoría de los casos eso significa que Debian no reconoció apropiadamente tu monitor o tarjeta de vídeo. Esto se puede remediar a veces instalando controladores o creando / editando archivos Xorg.conf.

1.3. Obteniendo LinuxCNC

Esta sección describe la forma recomendada de descargar y hacer una nueva instalación de LinuxCNC. Para los mas atrevidos, también hay métodos alternativos de instalación. Si su máquina tiene una instalación existente que desea actualizar, vaya a la sección Actualizando LinuxCNC.

nota
Para operar maquinaria, LinuxCNC requiere un kernel especial con extensiones para tiempo real. Existen tres posibilidades: preempt-rt, RTAI o Xenomai. Además hay dos versiones de LinuxCNC que funcionan con estos kernels. Para más detalles, vea la tabla mas abajo. No obstante, para pruebas de código y simulaciones es posible ejecutar la aplicación `linux-uspace` en un kernel de stock de la distribución.

Las instalaciones nuevas de LinuxCNC se crean fácilmente utilizando una imagen Live/Install. Esta es una imagen ISO híbrida del sistema de archivos que se puede grabar en un dispositivo de almacenamiento USB o un DVD para arrancar una computadora. En el momento del arranque se le dará la opción de arrancar el sistema "Live" (para ejecutar LinuxCNC sin hacer ningún cambio permanente en tu computadora) o arrancar un Instalador (para instalar LinuxCNC y su sistema operativo en el disco duro de tu computadora).

A grandes razgos, el proceso es como:

1. Descargar la imagen Live/Install.
2. Grabar la imagen en un dispositivo de almacenamiento USB o DVD.
3. Arrancar el sistema Live para probar LinuxCNC.
4. Arrancar el instalador para instalar LinuxCNC.

1.3.1. Descargar la imagen

Esta sección describe algunos métodos para descargar una imagen Live/Install.

1.3.1.1. Descarga normal

Software for LinuxCNC to download is presented on the project's Downloads page. Most users will aim for the disk image for Intel/AMD PCs, the URL will resemble https://www.linuxcnc.org/iso/linuxcnc_2.9.8-amd64.hybrid.iso.

Para la Raspberry Pi, se proporcionan múltiples imágenes para manejar las diferencias entre las RPi4 y RPi5.

nota
No uses la distribución normal de Raspbian para LinuxCNC que pudiera estar incluida en tu kit de inicio de RPi, esa no tendrá el núcleo de tiempo real y no podrás migrar desde Raspbian a la imagen del kernel de Debian.

1.3.1.2. Descargar usando zsync

zsync es una aplicación de descarga que reanuda eficientemente descargas interrumpidas y transfiere de manera eficiente archivos de gran tamaño con pequeñas modificaciones (si tienes una copia local anterior). Considera que necesita usar el protocolo http y no https. Usa zsync si tu descarga de la imagen se interrumpe frecuentemente usando el método de descarga normal.

zsync en Linux

1. Instalar zsync usando Synaptic o ejecutando lo siguiente en una terminal

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

2. Luego, ejecute este comando para descargar el iso a tu computadora

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

Por favor, no olvides confirmar la suma de verificación (checksum) del iso descargado como se describe a continuación, ya que no se garantiza la autenticidad del servidor con el protocolo http.

zsync en Windows Hay una versión para Windows de zsync. Funciona como una aplicación de consola y puede descargarse desde https://www.assembla.com/spaces/zsync-windows/documents .

1.3.1.3. Verificar la imagen

(Este paso no es necesario si usaste zsync)

1. Después de la descarga, verifique la suma de comprobación de la imagen para garantizar su integridad.

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

ó

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

1. Luego, compare las sumas de verificación

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

Verificar md5sum en Windows o Mac Windows does not come with an md5sum program, but there are alternatives. More information can be found at: How To MD5SUM

1.3.2. Escribir la imagen en un dispositivo de arranque

La imagen LinuxCNC Live/Install es una imagen ISO híbrida que se puede grabar directamente a un dispositivo de almacenamiento USB (unidad flash) o un DVD y se usa para arrancar un ordenador. La imagen es demasiado grande para caber en un CD.

1.3.2.1. Imagen para Raspberry Pi

The Raspbery Pi image is a complete SD card image and should be written to an SD card with the [Raspberry Pi Imager App](https://www.raspberrypi.com/software/). Note that the imager app can open the .zip file directly, no need to expand.

1.3.2.2. Imagen para AMD-64 (x86-64, PC) usando herramientas de GUI

Descarga e instala Balena Etcher desde https://etcher.balena.io/#download-etcher (Linux, Windows, Mac) y escribe la imagen descargada en una unidad USB.

Si tu imagen no arranca, por favor intenta también con Rufus. Se ve más complicado pero parece ser más compatible con varios BIOS.

1.3.2.3. Línea de comandos - Linux

1. Conecte un dispositivo de almacenamiento USB (por ejemplo, una unidad flash o un dispositivo similar).
2. Determine the device file corresponding to the USB flash drive. This information can be found in the output of `sudo dmesg` after connecting the device. `cat /proc/partitions` may also be helpful.
3. Use el comando `dd` para escribir la imagen en su dispositivo de almacenamiento USB. Por ejemplo, si su dispositivo de almacenamiento aparece como `/dev/sde`, use este comando:

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

1.3.2.4. Línea de comandos - MacOS

1. Abra una ventana de terminal teclee

```
diskutil list
```

2. Inserte el USB y anote el nombre del nuevo disco que aparece, p. Ej. /dev/disk5.
3. Desmonte el USB. El número encontrado arriba debe ser sustituido en lugar de la N.

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

4. Transfiera los datos con dd, como en Linux. Tenga en cuenta que el nombre del disco tiene una ”r” agregada al principio.

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

Grabar la imagen en un DVD en Linux

1. Inserte un DVD en blanco en su grabadora. Aparecerá una ventana con *CD/DVD Creator* o *Choose Disc Type*. Cierre eso, ya que no lo usaremos.
2. Busque la imagen descargada con el navegador de archivos.
3. Haga clic con el botón derecho en el archivo de imagen ISO y elija Escribir en Disco.
4. Seleccione la velocidad de escritura. Se recomienda que escriba a la menor velocidad posible.
5. Comience el proceso de grabación.
6. Si aparece una ventana para *elegir un nombre de archivo para la imagen del disco*, simplemente elija Aceptar.

Escribir la imagen en un DVD en Windows

1. Descargue e instale Infra Recorder, un programa de grabación de imagen gratuito de codigo abierto: https://infrarecorder.org/ .
2. Inserte un DVD en blanco en la unidad y seleccione No hacer nada o Cancelar si aparece el diálogo de ejecución automática.
3. Abra Infra Recorder, y seleccione el menú *Acciones*, luego *Grabar imagen*.

Escribir la imagen en un DVD en Mac OSX

1. Descargue el archivo .iso
2. Haga clic derecho en el archivo en la ventana Finder y seleccione ”Grabar en disco” (La opción de grabar en disco solo aparecerá si la máquina tiene un unidad óptica instalada o conectada)

1.3.3. Probando LinuxCNC

Con el dispositivo de almacenamiento USB conectado o el DVD en la unidad de DVD, apague el ordenador y vuelva a encenderlo. Esto arrancará el ordenador con la imagen Live/Install y elija la opción de arranque *Live*.

nota
Si el sistema no se inicia desde el DVD o la memoria USB, podría ser necesario cambiar el orden de arranque en el BIOS de la PC.

Una vez que el ordenador se haya iniciado, puede probar LinuxCNC sin instalarlo. No puede crear configuraciones personalizadas o modificar la mayoría de ajustes del sistema en una sesión en vivo, pero puede (y debe) correr el test de latencia.

Para probar LinuxCNC: desde el menú Aplicaciones/CNC, seleccione LinuxCNC. Se abrirá un cuadro de diálogo desde el cual puede elegir una de entre muchas configuraciones de muestra. En este punto, solo tiene sentido elegir una configuración ”sim”. Algunas de las configuraciones de muestra incluyen en pantalla máquinas simuladas en 3D, busque ”Vismach” para verlas.

Para ver si su ordenador es adecuado para la generación de pulsos de pasos por software ejecute la prueba de latencia como se muestra aquí.

Hasta el momento en que se escribió este documento, la imagen en vivo solo esta disponible con el kernel preempt-rt y su correspondiente LinuxCNC. En algún hardware podría no dar una latencia suficientemente buena. Una versión experimental esta disponible usando el kernel en tiempo real RTAI que suele dar una mejor latencia.

1.3.4. Installing LinuxCNC

Para instalar LinuxCNC desde el LiveCD, seleccione *Instalación (Gráfica)* al arrancar.

1.3.5. Actualizando LinuxCNC

Con la instalación normal, si tiene conexión a internet, el Administrador de actualizaciones le notificará de las actualizaciones de LinuxCNC y le permitirá actualizar fácilmente sin conocimientos de Linux. Está bien actualizar todo, excepto el sistema operativo, cuando se le pida.



aviso
No actualices el sistema operativo a una nueva versión mayor (upgrade) si se solicita hacerlo. En cambio, debes aceptar las actualizaciones menores de versión (*updates*), especialmente las de seguridad.

1.3.6. Problemas de instalación

In rare cases you might have to reset the BIOS to default settings if during the Live CD install it cannot recognize the hard drive during the boot up.

1.3.7. Métodos alternos de instalación

La manera más fácil y preferida de instalar LinuxCNC es usar la imagen Live/Install tal como se ha descrito arriba. Ese método es tan simple y confiable como hemos podido, y es adecuado tanto para usuarios principiantes como para usuarios experimentados. Sin embargo, esto normalmente reemplazará cualquier sistema operativo existente. Si tienes archivos en la PC destino que quieras mantener, entonces usa uno de los métodos descritos en esta sección.

Además, para usuarios experimentados que estén familiarizados con la administración de sistemas Debian (encontrar imágenes de instalación, manipular fuentes apt, cambiar variantes del kernel, etc.), las nuevas instalaciones son compatibles con las siguientes plataformas: (”amd64” significa ”64 bits”, y no es específico de los procesadores AMD; funcionará en cualquier sistema x86 de 64 bits)

Debian Trixie	**amd64 y arm64**	**preempt-rt**	**linuxcnc-uspace**	**machine control & simulation**
Debian Trixie	amd64	RTAI	linuxcnc	control de máquina
Distribution	Architecture	Kernel	Package name	Typical use
Debian Bookworm	amd64 y arm64	preempt-rt	linuxcnc-uspace	machine control & simulation
Debian Bookworm	amd64	RTAI	linuxcnc	control de máquina
Debian Bullseye	amd64	preempt-rt	linuxcnc-uspace	machine control & simulation
Cualquiera	Cualquiera	Stock	linuxcnc-uspace	SOLO simulación

nota
LinuxCNC v2.9 no es compatible con Debian 9 o versiones anteriores.

Kernels preempt-RT Los kernels Preempt-rt están disponibles para Debian desde el archivero regular de debian.org. El paquete se llama `linux-image-rt-*`. Simplemente instale el paquete de la misma manera que cualquier otro paquete desde al administrador de paquetes Synaptic o con apt-get en la línea de comandos.

Kernels RTAI Los kernels RTAI están disponibles para su descarga desde el archivador debian en linuxcnc.org. Las fuentes de apt son:

- Debian Trixie: `deb http://linuxcnc.org trixie base`
- Debian Bookworm: `deb http://linuxcnc.org bookworm base`
- Debian Bullseye: `deb http://linuxcnc.org bullseye base`
- Debian Buster: `deb http://linuxcnc.org buster base`

LinuxCNC con el kernel RTAI ahora está disponible para sistemas operativos de 64 bits, pero hay unos pocos sistemas sobrevivientes que no pueden correr un sistema operativo de 64 bits.

1.3.7.1. Installing on Debian Trixie (with Preempt-RT kernel)

1. Install Debian Trixie (Debian 13), amd64 version. You can download the installer here: https://www.debi
   distrib/
2. Después de quemar el iso y arrancar, si no quiere el escritorio Gnome, seleccione *Opciones avanzadas > Entornos de escritorio alternativos* y elija el que le guste. Luego seleccione *Instalar* o ’ Instalación gráfica’.



aviso
No ingrese una contraseña de root; si lo hace, sudo queda deshabilitado y no podrá completar los siguientes pasos.

3. Ejecute lo siguiente en una terminal para actualizar la máquina con los paquetes más recientes.

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

nota
It is possible to download a version of LinuxCNC directly from Debian (currently version 2.9.4) but a more up-to-date version (2.9.8) can be installed from the LinuxCNC repository.

4. Instale el kernel y los módulos Preempt-RT

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

5. Reinicie y seleccione el kernel de Linux 6.1.0-10-rt-amd64. La versión exacta del kernel podría ser diferente, observe el sufijo ”-rt”. Esto podría estar oculto en el submenú ”Opciones avanzadas para Debian Bookworm” en Grub. Cuando usted inicie sesión, verifique que el siguiente comando informe `PREEMPT RT`.

```
uname -v
```

6. Abra el menú de Aplicaciones > Sistema > Administrador de Paquetes Synaptic, busque *linux-image* y haga clic derecho en el original no rt y seleccione *Marcar para eliminación completa*. Reinicie. Esto es para forzar al sistema a arrancar desde el kernel RT. Si prefiere retener ambos kernels, no es necesario eliminar los otros, pero se necesitarán cambios de la configuración de arranque de Grub que van más allá del alcance de este documento.

7. Agrega la llave de firma del archivero de LinuxCNC a tu llavero apt descargando [el script de instalación de LinuxCNC](https://www.linuxcnc.org/linuxcnc-install.sh). Necesitarás hacer al script ejecutable para correrlo:

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

```
Entonces podrás ejecutar el instalador:
```

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

1.3.7.2. Installing on Debian Trixie (with experimental RTAI kernel)

1. This kernel and LinuxCNC version can be installed on top of the Live DVD install, or alternatively on a fresh Install of Debian Trixie 64-bit as described above.

2. Puedes agregar la llave de firma del archivero de LinuxCNC e información del repositorio descargando y ejecutando el script de instalación como se describió arriba. Si se detecta un kernel RTAI se detendrá antes de instalar algún paquete.

3. Actualice la lista de paquetes desde linuxcnc.org

```
sudo apt-get update
```

4. Remove the existing uspace version of LinuxCNC and install the new realtime kernel, RTAI and the RTAI-version of LinuxCNC.

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

Reinicia la máquina, asegurándote que el sistema arranque desde el nuevo kernel 5.4.258-rtai.

1.3.7.3. Instalación en Raspbian 12

No lo hagas. Las latencias son muy malas con el kernel predeterminado; y el kernel PREEMPT_RT (la RT es importante) de Debian no arranca en la Pi (a partir de 1/2024). Por favor refiérase a las imágenes .iso proporcionadas en línea con la página de descargas de LinuxCNC. Puedes crearlas por ti mismo siguiendo los scripts proporcionados en línea.

1.4. Ejecutando LinuxCNC

1.4.1. Invocando LinuxCNC

Después de la instalación, LinuxCNC comienza como cualquier otro programa de Linux: ejecútelo desde una terminal mediante el comando *linuxcnc*, o selecciónelo en el menú *Aplicaciones > CNC*.

1.4.2. Selector de configuración

Al iniciar LinuxCNC (desde el menú CNC o desde la línea de comandos sin especificar un archivo INI), se inicia el cuadro de diálogo Selector de Configuración.

El diálogo Selector de Configuración permite al usuario elegir una de las configuraciones existentes (Mis Configuraciones) o seleccionar una nueva (Configuraciones de Muestra) para copiarla a su directorio de inicio. Las configuraciones copiadas aparecerá en Mis configuraciones en la próxima invocación del Selector de Configuración.

El selector de configuración ofrece una selección de configuraciones organizada asi:

- *Mis configuraciones* - Configuraciones de usuario ubicadas en linuxcnc/configs en tu directorio home.
- *Configuraciones de ejemplo* - Estas configuraciones, cuando se seleccionan, se copian en linuxcnc/configs. Una vez que copie una configuración a su directorio local, el lanzador lo ofrecerá como *Mis configuraciones*. Los nombres bajo los cuales se presentan dichas configuraciones locales corresponden al de los directorios dentro del directorio configs/:
  - *sim* - Configuraciones que incluyen hardware simulado. Pueden ser usadas para probar o aprender cómo funciona LinuxCNC.
  - *by_interface* - Configuraciones organizadas por tipo de GUI.
  - *by_machine* - Configuraciones organizadas por tipo de máquina.
  - *apps* - Aplicaciones que no requieren iniciar linuxcnc pero pueden ser útiles para pruebas o aplicaciones como PyVCP o GladeVCP.
  - *attic* - Configuraciones obsoletas o históricas.

The sim configurations are often the most useful starting point for new users and are organized around supported GUIs:

- *axis* - Keyboard and Mouse GUI
- *craftsman* - GUI de pantalla táctil (¿ya no se mantiene?)
- *gmoccapy* - Touch Screen GUI
- *gscreen* - Touch Screen GUI
- *pyvcp_demo* - Panel de control virtual Python
- *qtaxis* - Touch Screen GUI, axis lookalike
- *qtdragon* - Touch Screen GUI
- *qtdragon_hd* - Touch Screen GUI, high definition
- *qtplasmac* - Touch Screen GUI, for plasma tables
- *qttouchy* - Touch Screen GUI
- *tklinuxcnc* - GUI de teclado y mouse (ya no se mantiene)
- *touchy* - Touch Screen GUI
- *woodpecker* - Touch Screen GUI

A GUI configuration directory may contain subdirectories with configurations that illustrate special situations or the embedding of other applications.

The *by_interface* configurations are organized around common, supported interfaces like:

- general mechatronics
- mesa
- parport
- pico
- pluto
- servotogo
- vigilant
- vitalsystems

Related hardware may be required to use these configurations as starting points for a system.

The *by_machine* configurations are organized around complete, known systems like:

- boss
- cooltool
- scortbot erIII
- sherline
- smithy
- tormach

A complete system may be required to use these configurations.

Las *apps* típicas son cualquiera de:

1. Utilerías que no requieren iniciar linuxcnc
2. Demostraciones de aplicaciones que pueden usarse con linuxcnc
   - info - Crea un archivo con información del sistema que puede ser útil para diagnóstico de problemas.
   - gladevcp - Example GladeVCP applications.
   - halrun - Starts halrun in an terminal.
   - latency - Applications to investigate latency
     - latency-histogram-1 - histogram for single servo thread
     - latency-histogram - histogram
     - latency-test - standard test
     - latency-plot - stripchart
   - parport - Applications to test parport.
   - pyvcp - Example pyvcp applications.
   - xhc-hb04 - Applications to test an xhc-hb04 USB wireless MPG

---

nota
Under the Apps directory, only applications that are usefully modified by the user are offered for copying to the user's directory.

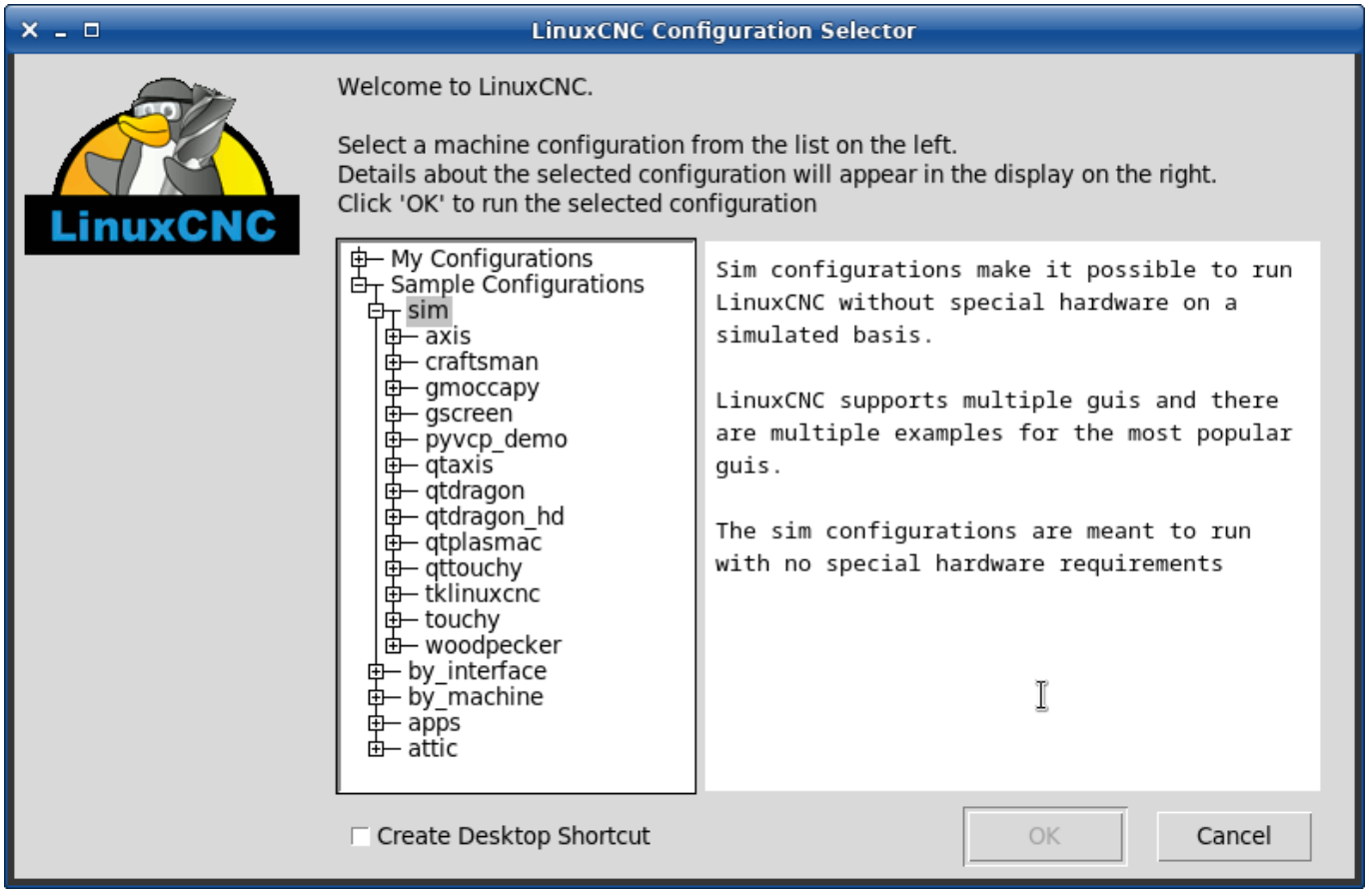


Figura 1.2: Selector de configuración de LinuxCNC

Click any of the listed configurations to display specific information about it. Double-click a configuration or click OK to start the configuration.

Select *Create Desktop Shortcut* and then click *OK* to add an icon on the Ubuntu desktop to directly launch this configuration without showing the Configuration Selector screen.

When you select a configuration from the Sample Configurations section, it will automatically place a copy of that config in the ~/linuxcnc/configs directory.

1.4.3. Next steps in configuration

After finding the sample configuration that uses the same interface hardware as your machine (or a simulator configuration), and saving a copy to your home directory, you can customize it according to the details of your machine. Refer to the Integrator Manual for topics on configuration.

1.4.4. Simulator Configurations

All configurations listed under Sample Configurations/sim are intended to run on any computer. No specific hardware is required and real-time support is not needed.

These configurations are useful for studying individual capabilities or options. The sim configurations are organized according to the graphical user interface used in the demonstration. The directory for axis contains the most choices and subdirectories because it is the most tested GUI. The capabilities demonstrated with any specific GUI may be available in other GUIs as well.

1.4.5. Configuration Resources

The Configuration Selector copies all files needed for a configuration to a new subdirectory of ~/linuxcnc/configs (equivalently: /home/username/linuxcnc/configs). Each created directory will include at least one INI file (iniflename.ini) that is used to describe a specific configuration.

File resources within the copied directory will typically include one or more INI file (filename.ini) for related configurations and a tool table file (toolfilename.tbl). Additionally, resources may include HAL files (filename.hal, filename.tcl), a README file for describing the directory, and configuration specific information in a text file named after a specific configuration (inifilename.txt). That latter two files are displayed when using the Configuration Selector.

The supplied sample configurations may specify the parameter HALFILE (filename.hal) in the configuration INI file that are not present in the copied directory because they are found in the system HAL file library. These files can be copied to the user configuration directory and altered as required by the user for modification or test. Since the user configuration directory is searched first when finding HAL files, local modifications will then prevail.

The Configuration selector makes a symbolic link in the user configuration directory (named hallib) that points to the system HAL file library. This link simplifies copying a library file. For example, to copy the library core_sim.hal file in order to make local modifications:

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

1.5. Actualizando LinuxCNC

Updating LinuxCNC to a new minor release (ie to a new version in the same stable series, for example from 2.9.7 to 2.9.8) is an automatic process if your PC is connected to the internet. You will see an update prompt after a minor release along with other software updates. If you don’t have an internet connection to your PC see Updating without Network.

1.5.1. Upgrade to the new version

Esta sección describe cómo actualizar LinuxCNC de la versión 2.8.x a la nueva versión 2.9.y. Se supone que tiene una instalación 2.8 existente que quiere actualizar.

To upgrade LinuxCNC from a version older than 2.8, you have to first upgrade your old install to 2.8, then follow these instructions to upgrade to the new version.

If you do not have an old version of LinuxCNC to upgrade, then you're best off making a fresh install of the new version as described in the section Getting LinuxCNC.

Furthermore, if you are running Ubuntu Precise, Debian Wheezy or Debian Buster it is well worth considering making a backup of the "linuxcnc" directory on removable media and performing a clean install of a newer OS and LinuxCNC version as these releases were EOL in 2017, 2018 and 2022 respectively. If you are running on Ubuntu Lucid then you will have to do this, as Lucid is no longer supported by LinuxCNC (it was EOL in 2013).

To upgrade major versions like 2.8 to 2.9 when you have a network connection at the machine you need to disable the old linuxcnc.org apt sources in the file /etc/apt/sources.list and add a new linuxcnc.org apt source for 2.9, then upgrade LinuxCNC.

The details will depend on which platform you're running on. Open a terminal then type `lsb_release -ic` to find this information out:

```
lsb_release -ic
Distributor ID: Debian
Codename:       Trixie
```

You should be running on Debian Bullseye, Bookworm or Trixie or Ubuntu 20.04 "Focal Fossa" or newer. LinuxCNC 2.9.y will not run on older distributions than these.

You will also need to check which realtime kernel is being used:

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

Si ve (como arriba) `-rt-` en el nombre del kernel, entonces está ejecutando el preempt-rt kernel y debería instalar la versión "uspace" de LinuxCNC. También debe instalar uspace para configuraciones "sim" en kernels que no son en tiempo real.

Si ve `-rtai-` en el nombre del kernel entonces está ejecutando RTAI tiempo real. Consulte a continuación la versión de LinuxCNC a instalar. Los paquetes RTAI están disponibles para Bookworm y Buster pero no actualmente para Bullseye.

1.5.1.1. Configuración de fuentes apt

- Abra la ventana `Fuentes de software`. El proceso para hacer esto se diferencia ligeramente en las tres plataformas soportadas:
  - Debian:
    - Haga clic en `Menú de aplicaciones`, luego en `Sistema`, luego en `Administrador de paquetes Synaptic`.
    - In Synaptic, click on the `Settings` menu, then click `Repositories` to open the `Software Sources` window.
  - Ubuntu Precise:
    - Click on the `Dash Home` icon in the top left.
    - In the `Search` field, type "software", then click on the `Ubuntu Software Center` icon.
    - In the Ubuntu Software Center window, click on the `Edit` menu, then click on `Software Sources...` to open the `Software Sources` window.

- Ubuntu Lucid:
  - Click the `System` menu, then `Administration`, then `Synaptic Package Manager`.
  - In Synaptic, click on the `Settings` menu, then click on `Repositories` to open the `Software Sources` window.

- In the `Software Sources` window, select the `Other Software` tab.
- Delete or un-check all the old linuxcnc.org entries (leave all non-linuxcnc.org lines as they are).
- Click the `Add` button and add a new apt line. The line will be slightly different on the different platforms:

Cuadro 1.2: Tabla resumen de las variantes de sistema operativo y su configuración de repositorio correspondiente. La configuración puede realizarse en la GUI del administrador de paquetes o en el archivo /etc/apt/sources.list.

SO / Versión tiempo real	**Repositorio**
Debian Bullseye - preempt	`deb https://linuxcnc.org bullseye base 2.9-uspace`
Debian Bookworm - preempt	`deb https://linuxcnc.org bookworm base 2.9-uspace`
Debian Bookworm - RTAI	`deb https://linuxcnc.org bookworm base 2.9-rt`
Debian Trixie - preempt	`deb https://linuxcnc.org trixie base 2.9-uspace`
Debian Trixie - RTAI	`deb https://linuxcnc.org trixie base 2.9-rt`

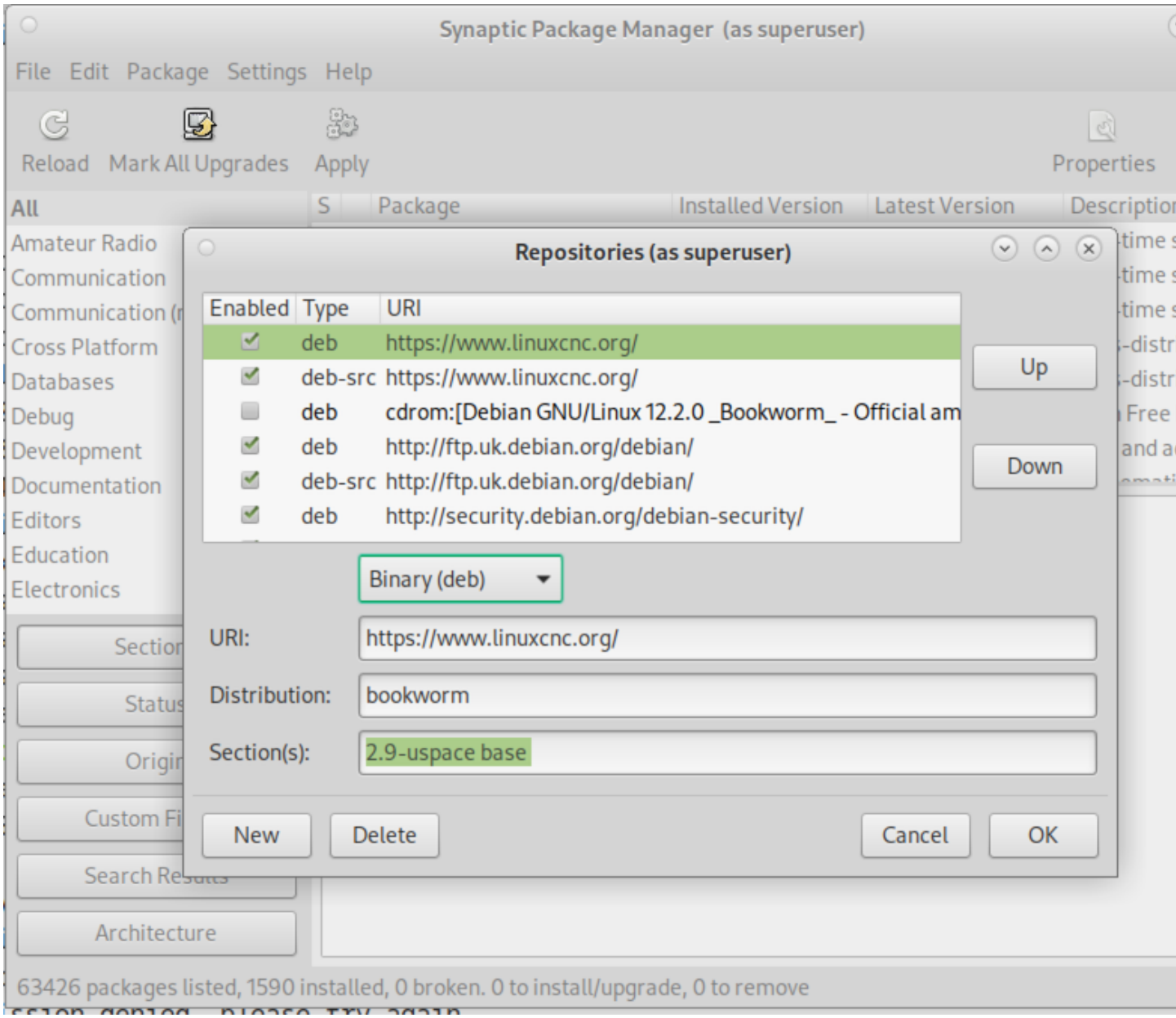


Figura 1.3: Captura de pantalla de una configuración de repositorio con el administrador de paquetes Synaptic.

- Click `Add Source`, then `Close` in the Software Sources window. If it pops up a window informing you that the information about available software is out-of-date, click the `Reload` button.

1.5.1.2. Upgrading to the new version

Ahora su computadora sabe dónde obtener la nueva versión del software. A continuación tenemos que instalarlo.

The process again differs depending on your platform.

Debian uses the Synaptic Package Manager.

- Open Synaptic using the instructions in Setting apt sources above.

- Click the `Reload` button.
- Use the Search function to search for `linuxcnc`.
- The package is called ”linuxcnc” for RTAI kernels and ”linuxcnc-uspace” for preempt-rt.
- Haga clic en la casilla de verificación para marcar los paquetes linuxcnc y linuxcnc-doc-* para actualización. El administrador de paquetes puede seleccionar un número de paquetes adicionales que también se instalarán, para satisfacer las dependencias que tiene el nuevo paquete linuxcnc.
- Haga clic en el botón `Aplicar` y deje que su computadora instale el nuevo paquete. El viejo paquete linuxcnc se actualizará automáticamente al nuevo.

1.5.1.3. Ubuntu

- Click on the `Dash Home` icon in the top left.
- In the `Search` field, type ”update”, then click on the `Update Manager` icon.
- Click the `Check` button to fetch the list of packages available.
- Click the `Install Updates` button to install the new versions of all packages.

1.5.2. Updating without Network

Para actualizar sin una conexión de red, necesita descargar un paquete .deb y luego instalarlo con dpkg. Los .deb se pueden encontrar en https://linuxcnc.org/dists/ .

Tiene que profundizar en el enlace anterior para encontrar el deb correcto para su instalación. Abra una terminal y escriba *lsb_release -ic* para encontrar el nombre de su sistema operativo.

```
> lsb_release -ic
Distributor ID: Debian
Codename:       trixie
```

Pick the OS from the list then pick the major version you want like 2.9-rt for RTAI or 2.9-uspace for preempt-rt.

Next pick the type of computer you have: binary-amd64 for 64-bit PC or binary-arm64 (64bit) for Raspberry Pi.

Next pick the version you want from the bottom of the list like *linuxcnc-uspace_2.9.8_amd64.deb* (choose the latest by date). Download the deb and copy it to your home directory. You can rename the file to something a bit shorter with the file manager like *linuxcnc_2.9.8.deb* then open a terminal and install it with the package manager with this command:

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

1.5.3. Updating Configuration Files for 2.9

1.5.3.1. Manejo estricto de intérpretes enchufables

Si sólo ejecutas código G normal y no sabes qué es un intérprete enchufable entonces esta sección no te afecta.

Una prestación de LinuxCNC que se usa rara vez es el soporte de intérpretes enchufables, controlados por la configuración INI no documentada `[TASK]INTERPRETER`.

Versiones de LinuxCNC anteriores a 2.9.0 manejaban una configuración incorrecta de `[TASK]INTERPRETER` que caía en automático de vuelta al uso del intérprete de código G predeterminado.

Desde 2.9.0, un valor incorrecto de `[TASK]INTERPRETER` provocará que LinuxCNC se niegue a arrancar. La situación se arregla eliminando dicha configuración del archivo INI, para que LinuxCNC use el intérprete de código G predeterminado.

1.5.3.2. Canterp

Si sólo ejecutas código G normal y no usas el intérprete enchufable `canterp` entonces esta sección no te afecta.

En el extremadamente improbable evento de usar `canterp`, debes saber que el módulo has sido movido de `/usr/lib/libcanterp.so` a `/usr/lib/linuxcnc/canterp.so`, y por lo tanto se necesita modificar la configuración correspondiente `[TASK]INTERPRETER` de `libcanterp.so` a `canterp.so`.

1.5.3.3. Spindle limits in the INI

Ahora es posible agregar configuraciones a la sección [SPINDLE] del archivo INI

MAX_FORWARD_VELOCITY = 20000 La velocidad máxima del husillo (en rpm)

MIN_FORWARD_VELOCITY = 3000 La velocidad de husillo mínima (en rpm)

MAX_REVERSE_VELOCITY = 20000 This setting will default to MAX_FORWARD_VELOCITY if omitted.

MIN_REVERSE_VELOCITY = 3000` This setting is equivalent to MIN_FORWARD_VELOCITY but for reverse spindle rotation. It will default to the MIN_FORWARD_VELOCITY if omitted.

INCREMENT = 200 Sets the step size for spindle speed increment / decrement commands. This can have a different value for each spindle. This setting is effective with AXIS and Touchy but note that some control screens may handle things differently.

HOME_SEARCH_VELOCITY = 100 - Accepted but currently does nothing

HOME_SEQUENCE = 0 - Accepted but currently does nothing

1.5.4. Updating Configuration Files for 2.10.y

Touchy: las entradas de Touchy MACRO ahora deben ser colocadas en la sección [MACROS] de INI, en vez de la sección [TOUCHY]. Esto es parte de un proceso de encomunar las configuraciones INI entre GUIs.

1.5.5. New HAL components

1.5.5.1. Non-Realtime

mdro mqtt-publisher pi500_vfd pmx485-test qtplasmac-cfg2prefs qtplasmac-materials qtplasmac-plasmac2c qtplasmac-setup sim-torch svd-ps_vfd

1.5.5.2. En tiempo real

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

1.5.6. New Drivers

A framework for controlling ModBus devices using the serial ports on many Mesa cards has been introduced. http://linuxcnc.org/docs/2.9/html/drivers/mesa_modbus.html

A new GPIO driver for any GPIO which is supported by the gpiod library is now included: http://linuxcnc.org/-docs/2.9/html/drivers/hal_gpio.html

1.6. Preguntas y respuestas frecuentes de Linux

Estos son algunos comandos básicos y técnicas para nuevos usuarios de Linux. Información más completa se puede encontrar en la web o mediante las páginas del manual con el comando man.

1.6.1. Inicio de sesión automático

1.6.1.1. Debian

Debian Stretch usa por defecto el entorno de escritorio Xfce, con el gestor de pantallas lightDM. Para obtener acceso automatico con Stretch:

- En una terminal, use el comando:

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

- Anote la ruta absoluta del archivo de configuración lightdm.conf.
- Edite ese archivo con un editor de texto puro (gedit, nano, etc), como root.
- Busque y descomente las lineas:

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

- Haga autologin-user=su_nombre_usuario
- Guarde y reinicie.

1.6.1.2. Ubuntu

Al instalar LinuxCNC con el CD de Ubuntu por defecto se tiene que iniciar sesión cada vez que encienda el ordenador. Para activar autentificacion automática vaya a *System > Administration > Login Window*. Si se trata de una instalación nueva, la ventana de inicio de sesión puede tardar unos segundo en aparecer. Usted tiene que tener la contraseña que utilizo para la instalación para acceder a la ventana de configuración de inicio. En la pestaña seguridad marque Habilitar Acceso Automático y elija un nombre de usuario de la lista (seleccione su nombre de usuario).

1.6.2. Arranque automático

Para tener un inicio automático de LinuxCNC con su configuración después de encender el equipo vaya a *System > Preferences > Sessions > Startup Applications*, seleccione la opcion agregar nuevo. Vaya a su carpeta de configuración y seleccione el archivo .ini. Cuando el cuadro de dialogo selector de archivos se cierra, añadir linuxcnc y un espacio al frente de la ruta a su archivo .ini.

Ejemplo:

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

La documentación se refiere a tu respectivo archivo .ini como archivo INI.

1.6.3. Terminal

Hay que hacer muchas cosas desde la terminal, como verificar el búfer de mensajes del núcleo con *dmesg*. Ubuntu y Linux Mint tienen un atajo de teclado Ctrl + Alt + t. Debian Stretch no tiene definido ningún atajo de teclado. Se puede crear facilmente con el *Administrador de configuración*. La mayoría de los administradores de archivos modernos admiten el botón derecho para abrir una terminal, solo asegúrese de hacer clic derecho en un área en blanco o en un directorio y no en un nombre de archivo. La mayoría de los sistemas operativos tienen la terminal como elemento de menú, generalmente en *Accesorios*.

1.6.4. Páginas del manual

Una página de manual es una forma de documentación de software que se encuentra comúnmente en un sistema operativo UNIX o similar a UNIX, como Linux.

Para ver una página del manual abra una terminal, para encontrar algo acerca del comando de búsqueda ”find” teclee:

```
man find
```

Utilice las teclas página-arriba y pagina-abajo para ver las páginas del manual y la tecla Q para salir de la visualización.

---

nota
Al ver la página de manual desde la terminal, es posible que no obtenga la página de manual esperada. Por ejemplo, si escribe man abs obtendrá el abs de C, no el abs de LinuxCNC. Lo mejor es ver las páginas de manual de LinuxCNC en los documentos HTML.

1.6.5. Lista de módulos

A veces, para soluciónar algún problema, usted necesita obtener una lista de los módulos que se encuentran cargados. En una ventana de terminal teclee:

```
lsmod
```

Si desea enviar la salida de lsmod a un archivo de texto teclee:

```
lsmod > mymod.txt
```

El archivo de texto resultante *mymod.txt*, se colocará en el directorio de inicio si usted no cambió de directorio cuando abrió la terminal, y será nombrado mymod.txt o como usted lo haya nombrado.

1.6.6. Edición de archivos de root

Cuando abra el explorador de archivos y vea que el propietario del archivo es el usuario root, se tienen que hacer algunos pasos adicionales para modificar ese archivo. La edición de algunos archivos de root puede traer malos resultados, tenga cuidado al hacerlo; generalmente usted puede ver y abrir la mayoría de los archivos root, pero se abrirán en modo de *solo lectura*.

1.6.6.1. Con la línea de comandos

Abra una ventana de terminal teclee

```
sudo gedit
```

Abrir el archivo con el menú File > Open > Edit

1.6.6.2. Usando la interfaz gráfica

1. Haga clic derecho sobre el escritorio y seleccione *Crear lanzador*.
2. Escriba un nombre como *editar sudo*.
3. Escriba *gksudo "gnome-open %u"* como el comando y guarde el lanzador en su escritorio.
4. Arrastre un archivo a su lanzador para abrir y editar.

1.6.6.3. Acceso tipo super usuario

En Ubuntu puede convertirse en root tecleando "sudo -i" en una ventana de terminal y luego tecleando su contraseña. Tenga cuidado porque usted puede dañar su instalación si no sabe lo que esta haciendo.

1.6.7. Comandos en la terminal

1.6.7.1. Directorio de trabajo

Para encontrar la ruta del directorio de trabajo actual en la terminal, teclee:

```
pwd
```

1.6.7.2. Cambiar directorios

Para cambiar el directorio trabajo a un nivel arriba, p. ej. el directorio padre, teclee:

```
cd ..
```

Para subir dos niveles en la terminal teclee:

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

Para moverte directamente a tu directorio principal de usuario usa en la terminal el comando cd sin argumentos:

```
cd ..
```

Para moverse hacia abajo al subdirectorio linuxcnc/configs en la terminal teclee:

```
cd linuxcnc/configs
```

1.6.7.3. Listar los archivos en un directorio

Para ver una lista de todos los archivos y subdirecciones en la terminal teclee:

```
dir
```

ó

```
ls
```

1.6.7.4. Encontrar un archivo

El comando *find* puede ser un poco confuso para un usuario nuevo de Linux. La sintaxis básica es:

```
find directorio-inicio parámetros acciones
```

Por ejemplo para encontrar todos los archivos .ini en su directorio de linuxcnc primero tiene que usar el comando pwd para ver el directorio.

Abra una nueva ventana de terminal y teclee:

```
pwd
```

y pwd podría devolver el siguiente resultado:

```
/home/joe
```

Con esta información se pondrá el comando conjunto de esta manera:

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

Aqui, -name es el nombre del archivo que se busca y -print hace que se muestre el resultado en la ventana de terminal. El nombre *.ini indica *devolver todos los archivos que tienen la extensión .ini*. La diagonal se requiere para escapar los metacaracteres de la consola. Si desea más información al respecto, vea las páginas del manual de *find*.

1.6.7.5. Búsqueda de texto

```
grep -irl 'texto a buscar' *
```

Este comando encuentra todos los archivos que contienen el *texto a buscar* en el directorio actual y todos los subdirectorios por debajo de este, sin tener en cuenta el uso de mayúsculas. La -i es para ignorar mayúsculas, la -r es para recursión (incluir todos los subdirectorios en la búsqueda) y la opcion -l retornará una lista de los nombres de archivo. Si no se usa -l también se obtendrá el texto donde fue encontrada cada ocurrencia de *texto a buscar*. El * es un comodín para buscar todos los archivos. Ver la página de manual de grep para más información.

1.6.7.6. Mensajes de diagnóstico

Para ver los mensajes de arranque usar "dmesg" en la ventana de comandos. Para guardar los mensajes de arranque en un archivo use el operador de redirección >, de esta manera:

```
dmesg > bootmsg.txt
```

El contenido de este archivo puede ser copiado y pegado en línea para compartir con la gente que le este intentando ayudar a diagnosticar un problema.

Para borrar el búfer de mensajes, teclee:

```
sudo dmesg -c
```

Esto puede ser útil justo antes del arranque de LinuxCNC, por lo que solo habrá un registro de información relacionada con el lanzamiento actual de LinuxCNC.

Para encontrar la dirección de un puerto paralelo integrado use grep para filtrar la información producida por dmesg.

Después del arranque abrir una terminal y escribir:

```
dmesg|grep parport
```

1.6.8. Elementos convenientes

1.6.8.1. Iniciador de terminal

Si quiere añadir un iniciador de terminal en la barra del panel en la parte superior de la pantalla normalmente puede hacer clic derecho en el panel en la parte superior de la pantalla y seleccionar ”añadir al panel”. Seleccione lanzador de aplicación personalizado y agregar. Dele un nombre y use el comando gnome-terminal en la caja de comando.

1.6.9. Problemas de hardware

1.6.9.1. Información del hardware

Para encontrar que hardware está conectado a la placa base, en una ventana de terminal teclee:

```
lspci -v
```

1.6.9.2. Resolución del monitor

Durante la instalación, Ubuntu intentará detectar la configuración del monitor. Si esto no funciona el sistema se instalará con una resolución máxima de 800x600.

Instrucciones para arreglar esto se encuentran aquí:

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

1.6.10. Rutas

Rutas relativas Las rutas relativas se basan en el directorio de inicio, que es el directorio que contiene el archivo INI. El uso de rutas relativas puede facilitar la reubicación de configuraciones, pero requiere una buena comprensión de los especificadores de ruta de Linux.

```
./f0        es lo mismo que f0, p. ej. un archivo llamado f0 en el directorio de ↩
   inicio
../f1       se refiere a un archivo llamado f1 en el directorio padre
../../f2    se refiere a un archivo llamado f2 en el directorio padre del padre
../../../f3 etc.
```

Capítulo 2

Información General para el Usuario

Prefacio

LinuxCNC is modular and flexible. These attributes lead many to see it as a confusing jumble of little things and wonder why it is the way it is. This page attempts to answer that question before you get into the thick of things.

LinuxCNC comenzó en el Instituto Nacional de Estándares y Tecnología (NIST) en EE.UU.. Creció usando Unix como su sistema operativo. Unix lo hizo diferente. Entre los primeros desarrolladores de Unix creció un conjunto de ideas de escritura de código que algunos llaman ”la manera Unix”. Estos primeros autores de LinuxCNC siguieron esas maneras.

Eric S. Raymond, en su libro The Art of Unix Programming, resume la filosofía Unix como una filosofía de ingeniería ampliamente utilizada, ”Mantenlo simple, estúpido” (”Keep it Simple, Stupid” o Principio KISS). Luego describe cómo cree que esta filosofía general se aplica como una norma cultural en Unix aunque, como era de esperar, no es difícil encontrar violaciones graves de la mayoría de las siguientes prácticas en la realidad de Unix:

- Rule of Modularity: Write simple parts connected by clean interfaces.
- Rule of Clarity: Clarity is better than cleverness.
- Rule of Composition: Design programs to be connected to other programs.
- Rule of Separation: Separate policy from mechanism; separate interfaces from engines.[1]

Mr. Raymond offered several more rules but these four describe essential characteristics of the LinuxCNC motion control system.

The **Modularity** rule is critical. Throughout these handbooks you will find talk of the interpreter or task planner or motion or HAL. Each of these is a module or collection of modules. It’s modularity that allows you to connect together just the parts you need to run your machine.

The **Clarity** rule is essential. LinuxCNC is a work in progress — it is not finished nor will it ever be. It is complete enough to run most of the machines we want it to run. Much of that progress is achieved because many users and code developers are able to look at the work of others and build on what they have done.

The **Composition** rule allows us to build a predictable control system from the many modules available by making them connectable. We achieve connectability by setting up standard interfaces to sets of modules and following those standards.

The **Separation** rule requires that we make distinct parts that do little things. By separating functions debugging is much easier and replacement modules can be dropped into the system and comparisons easily made.

¿Qué significa ”la manera Unix” para usted como usuario de LinuxCNC?. Significa que usted puede tomar decisiones sobre cómo usará el sistema. Muchas de estas elecciones se harán durante la integración de la máquina, pero muchas también afectaran a la forma en que se usará la máquina. Durante la lectura de la documentacion, encontrará muchos lugares donde tendrá que hacer comparaciones. Eventualmente se deberan tomar decisiones, ”Usaré esta interfaz en lugar de esa otra” o

[1] Found at link:https://en.wikipedia.org/wiki/Separation_of_mechanism_and_policy, 2022-11-13

”Voy a escribir offsets de piezas de esta manera en lugar de hacerlo de esa otra”. A lo largo de estos manuales describimos el rango de posibilidades disponibles actualmente.

As you begin your journey into using LinuxCNC we offer two cautionary notes:[2]

- Parafraseando las palabras de Doug Gwyn en UNIX: ”LinuxCNC no fue diseñado para evitar que sus usuarios hagan cosas estúpidas, ya que eso también les impediría hacer cosas ingeniosas.”
- También las palabras de Steven King: ”LinuxCNC es fácil de usar. Simplemente no se descuida con respecto a qué tipo de usuarios es amigable”.

A series of videos on YouTube provide plenty of evidence a transition to LinuxCNC is possible no matter what your regular computer operating system may be. That said, with the advent of additive manufacturing like 3D printing there is an increasing interest by the broader IT community in CNC machining and it should be possible to find someone with complementary skills/equipment near to you to jointly overcome the initial hurdles.

2.1. Introducción al usuario de LinuxCNC

2.1.1. Introducción

This document is focused on the use of LinuxCNC, it is intended for readers who have already installed and configured it. Some information on installation is given in the following chapters. The complete documentation on installation and configuration can be found in the integrator’s manual.

2.1.2. How LinuxCNC Works

LinuxCNC is a suite of highly-customisable applications for the control of a Computer Numerically Controlled (CNC) mills and lathes, 3D printers, robots, laser cutters, plasma cutters and other automated devices. It is capable of providing coordinated control of up to 9 axes of movement.

At its heart, LinuxCNC consists of several key components that are integrated together to form one complete system:

- a Graphical User Interface (GUI), which forms the basic interface between the operator, the software and the CNC machine itself;
- una capa de abstracción de hardware (HAL), que proporciona un método para vincular las diversas señales virtuales internas generadas y recibidas por LinuxCNC con el mundo exterior, y
- the high level controllers that coordinate the generation and execution of motion control of the CNC machine, namely the motion controller (EMCMOT), the discrete input/output controller (EMCIO) and the task executor (EMCTASK).

The below illustration is a simple block diagram showing what a typical 3-axis CNC mill with stepper motors might look like:

[2] Found at link:https://en.wikipedia.org/wiki/Unix_philosophy, 07/06/2008

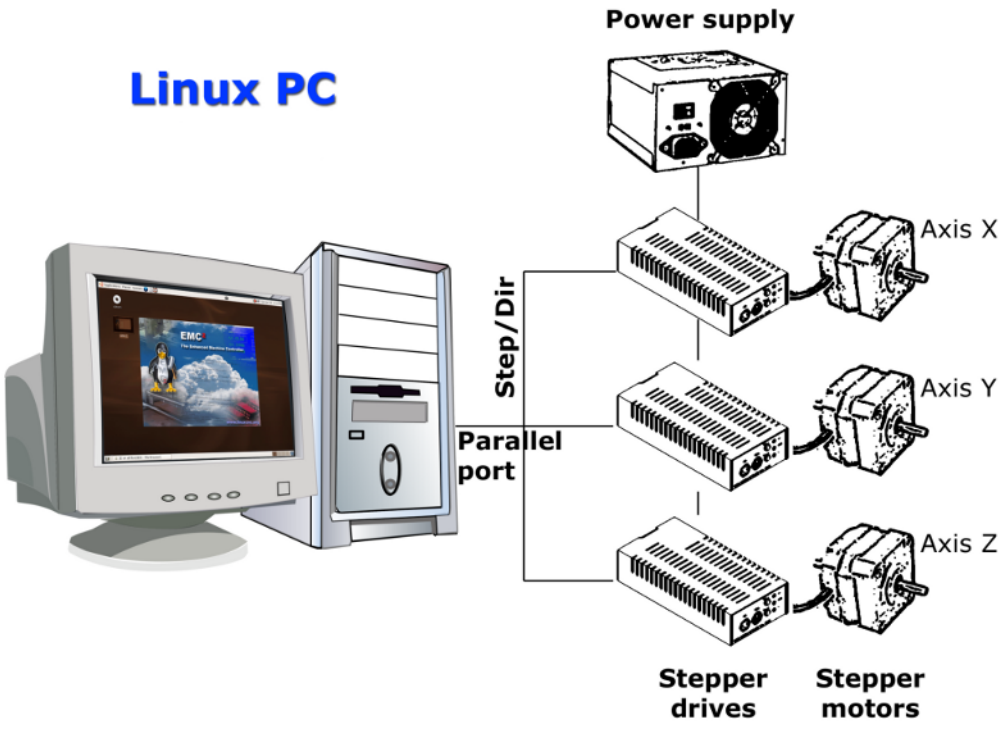


Figura 2.1: Simple LinuxCNC Controlled Machine

Una computadora que ejecuta LinuxCNC envía una secuencia de impulsos a través del puerto paralelo a las unidades paso a paso; de las cuales cada una tiene un motor paso a paso conectado. Cada unidad recibe dos señales independientes; una señal para ordenar al accionamiento que mueva su motor paso a paso asociado en sentido horario o antihorario (señal de direccion), y una segunda señal (tren de pulsos) que define la velocidad a la que gira el motor paso a paso.

Se ha ilustrado un sistema de motor paso a paso bajo control del puerto paralelo, pero un sistema LinuxCNC también puede conectarse a una amplia variedad de interfaces hardware dedicadas de control de movimiento para una mayor velocidad o capacidades de E/S. Se puede encontrar una lista completa de interfaces compatibles con LinuxCNC en la página Wiki Hardware soportado.

In most circumstances, users will create a configuration specific to their mill setup using either the Stepper Configuration Wizard (for CNC systems operating using the computers' parallel port) or the Mesa Hardware Wizard (for more advanced systems utilising a Mesa Anything I/O PCI card). Running either wizard will create several folders on the computers' hard drive containing a number of configuration files specific to that CNC machine, and an icon placed on the desktop to allow easy launching of LinuxCNC.

For example, if the Stepper Configuration Wizard was used to create a setup for the 3-axis CNC mill illustrated above entitled *My_CNC*, the folders created by the wizard would typically contain the following files:

- **Carpeta:** `Mi_CNC`
  - `Mi_CNC.ini`
    El archivo INI contiene toda la información básica del hardware relacionado con el funcionamiento de la fresadora CNC, tales como el número de pasos que debe realizar cada motor paso a paso para completar una revolución completa, la velocidad máxima en que cada paso a paso puede operar, los límites de recorrido de cada eje o la configuración y el comportamiento de interruptores de límite en cada eje.

- **`Mi_CNC.hal`**
  Este archivo HAL contiene información que le dice a LinuxCNC cómo vincular las señales virtuales internas a conexiones físicas exteriores a la computadora. Por ejemplo, especificando el pin 4 en el puerto paralelo para enviar la señal de dirección de paso del eje Z, o instruir a LinuxCNC para dejar de mover el motor del eje X cuando se activa el interruptor de límite conectado al pin 13 del puerto paralelo.
- **`custom.HAL`**
  Las personalizaciones para la configuración de la fresadora, más allá del alcance del asistente, pueden realizarse incluyendo enlaces adicionales a otros puntos virtuales dentro de LinuxCNC en este archivo HAL. Al iniciar una sesión de LinuxCNC, este archivo se lee y se procesa antes de cargar la GUI. Por ejemplo, puede incluir iniciar las comunicaciones Modbus con el driver del husillo para que se confirme como operativo antes de que se muestre la GUI.
- **`custom_postgui.hal`**
  El archivo HAL custom_postgui permite una mayor personalización de LinuxCNC, pero difiere de custom.HAL en que se procesa después de que se muestre la GUI. Por ejemplo, después de establecer comunicaciones Modbus al driver del motor del husillo en custom.hal, LinuxCNC puede usar el archivo custom_postgui para vincular la lectura de la velocidad del husillo desde el driver del motor con un gráfico de barras que se muestra en la GUI.
- **`postgui_backup.hal`**
  Proporciona una copia de seguridad del archivo custom_postgui.hal para permitir al usuario restaurar rápidamente una configuración HAL postgui que funcionaba anteriormente. Esto es especialmente útil si el usuario desea ejecutar el Asistente de configuración nuevamente bajo el mismo nombre, *Mi_CNC*, para modificar algunos parámetros. Al guardar la nueva configuración con el asistente, se sobrescribirá el archivo custom_postgui existente mientras que el archivo postgui_backup queda intacto.
- **`tool.tbl`**
  Un archivo de tabla de herramientas contiene una lista parametrizada de cualquier herramienta de corte utilizada. Estos parámetros puede incluir el diámetro y la longitud del cortador, y se utiliza para proporcionar un catálogo de datos que le dice a LinuxCNC cómo compensar su movimiento para herramientas de diferentes tamaños dentro de una operación de fresado.

- **Carpeta: `nc_files`**
  La carpeta nc_files se proporciona como la ubicación predeterminada para almacenar los programas de código G utilizados en la máquina. También incluye varias subcarpetas con ejemplos de código G.

2.1.3. Interfaces de usuario gráficas

A graphical user interface is the part of the LinuxCNC that the machine tool operator interacts with. LinuxCNC comes with several types of user interfaces which may be chosen from by editing certain fields contained in the INI file:

AXIS
AXIS, the standard keyboard GUI interface. This is also the default GUI launched when a Configuration Wizard is used to create a desktop icon launcher:

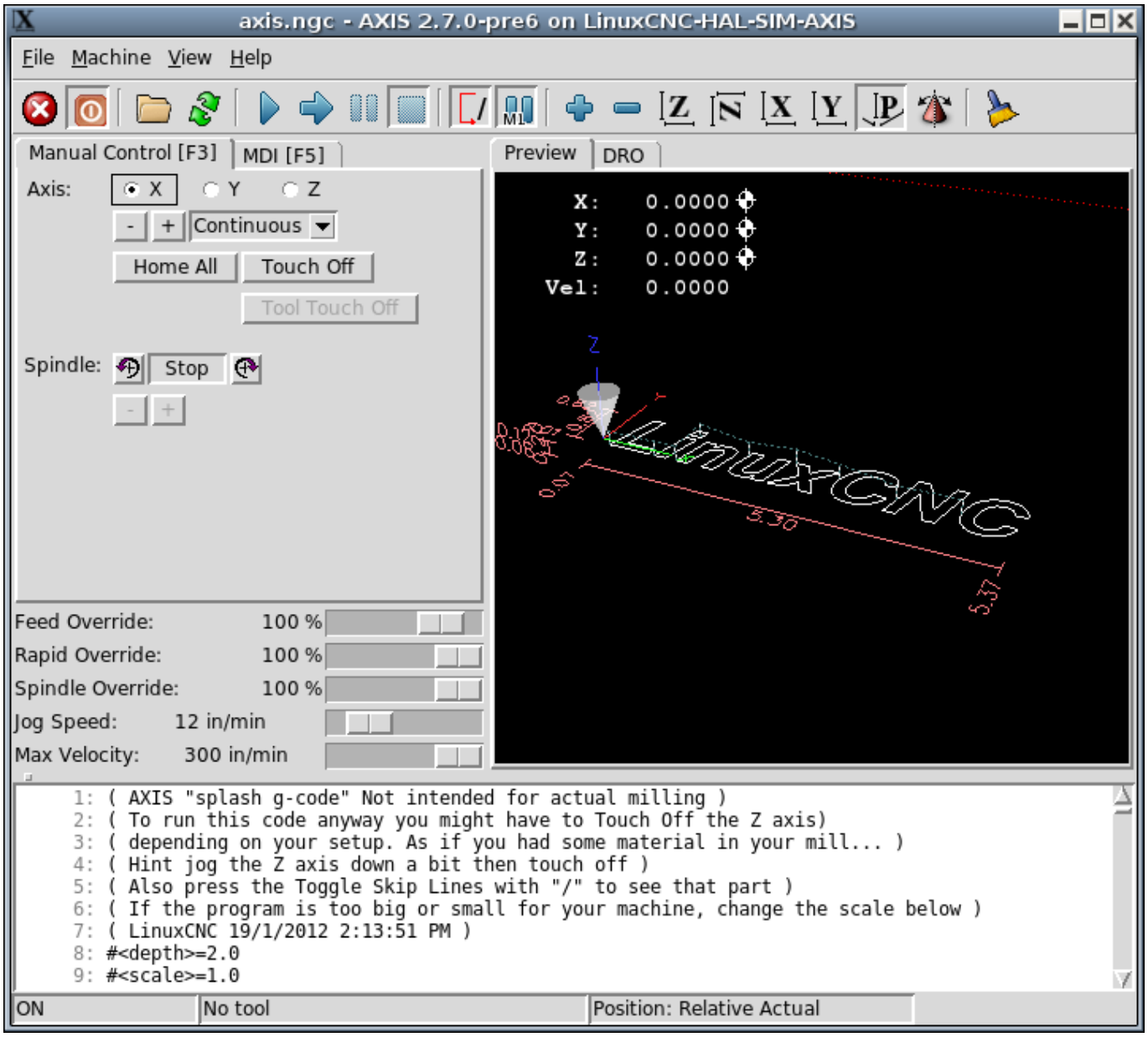


Figura 2.2: AXIS, the standard keyboard GUI interface

Touchy

Touchy, a touch screens GUI:

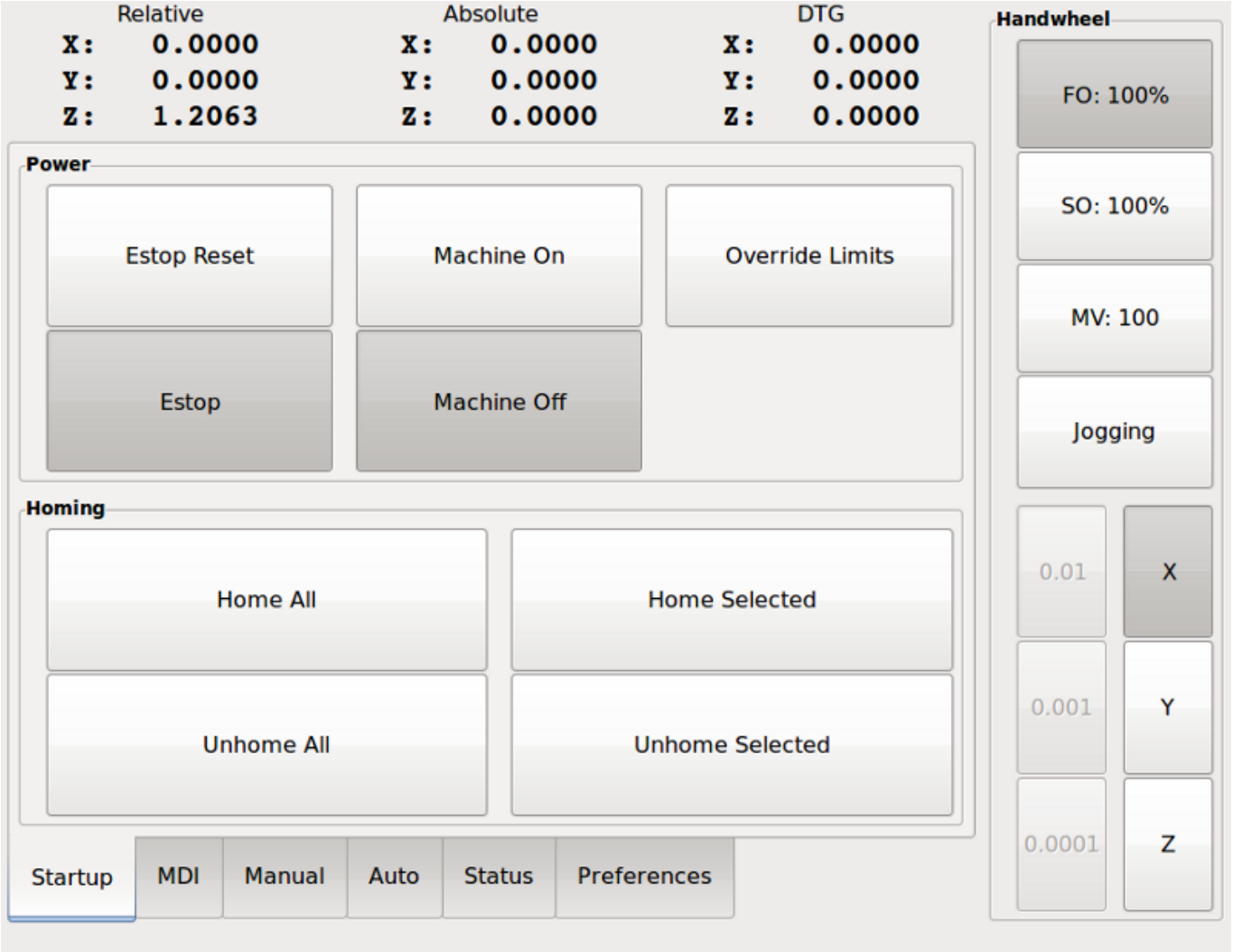


Figura 2.3: Touchy, a touch screen GUI

Gscreen

Gscreen, a user-configurable touch screen GUI:

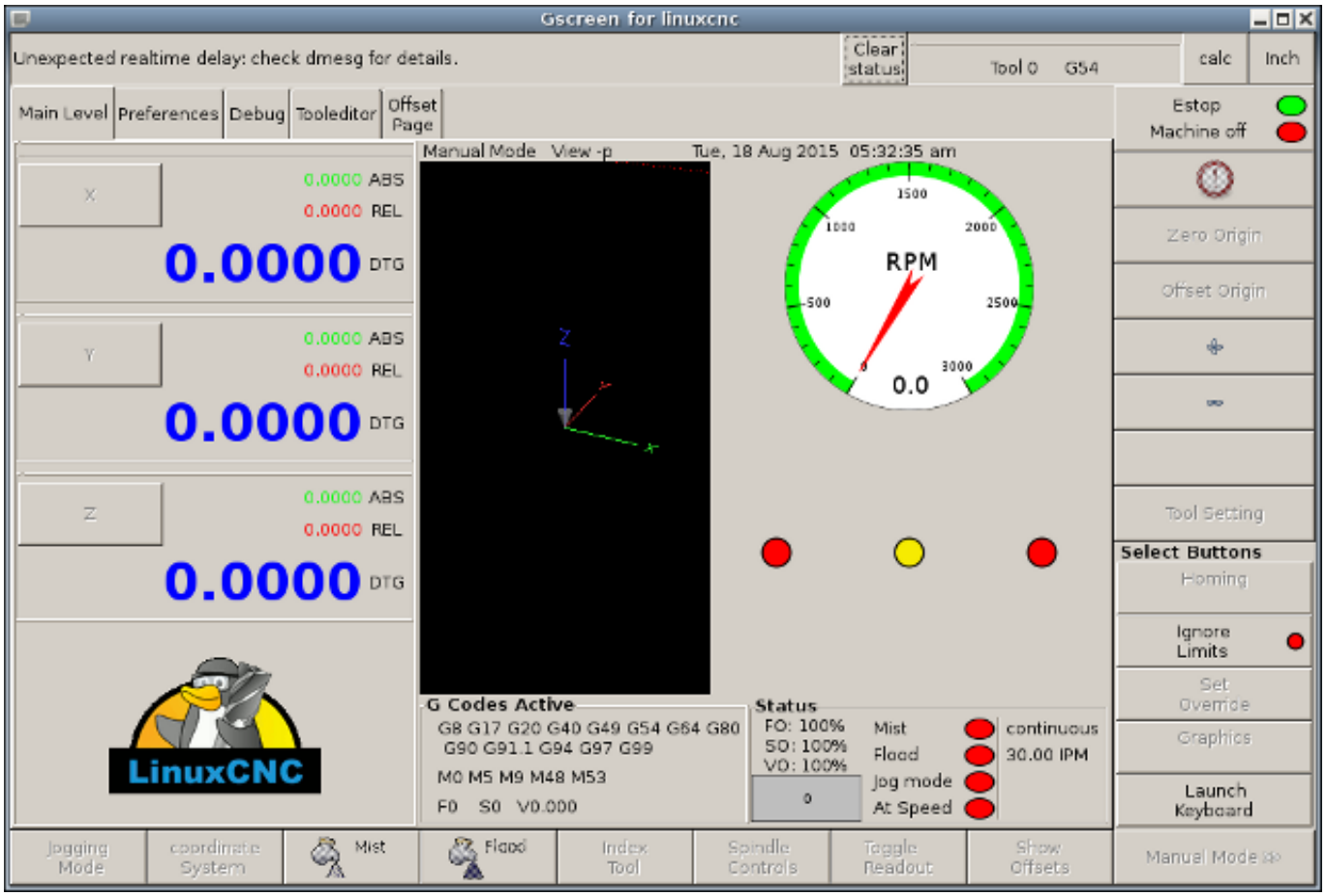


Figura 2.4: Gscreen, a configurable base touch screen GUI

GMOCCAPY

GMOCCAPY, a touch screen GUI based on Gscreen. GMOCCAPY is also designed to work equally well in applications where a keyboard and mouse are the preferred methods of controlling the GUI:

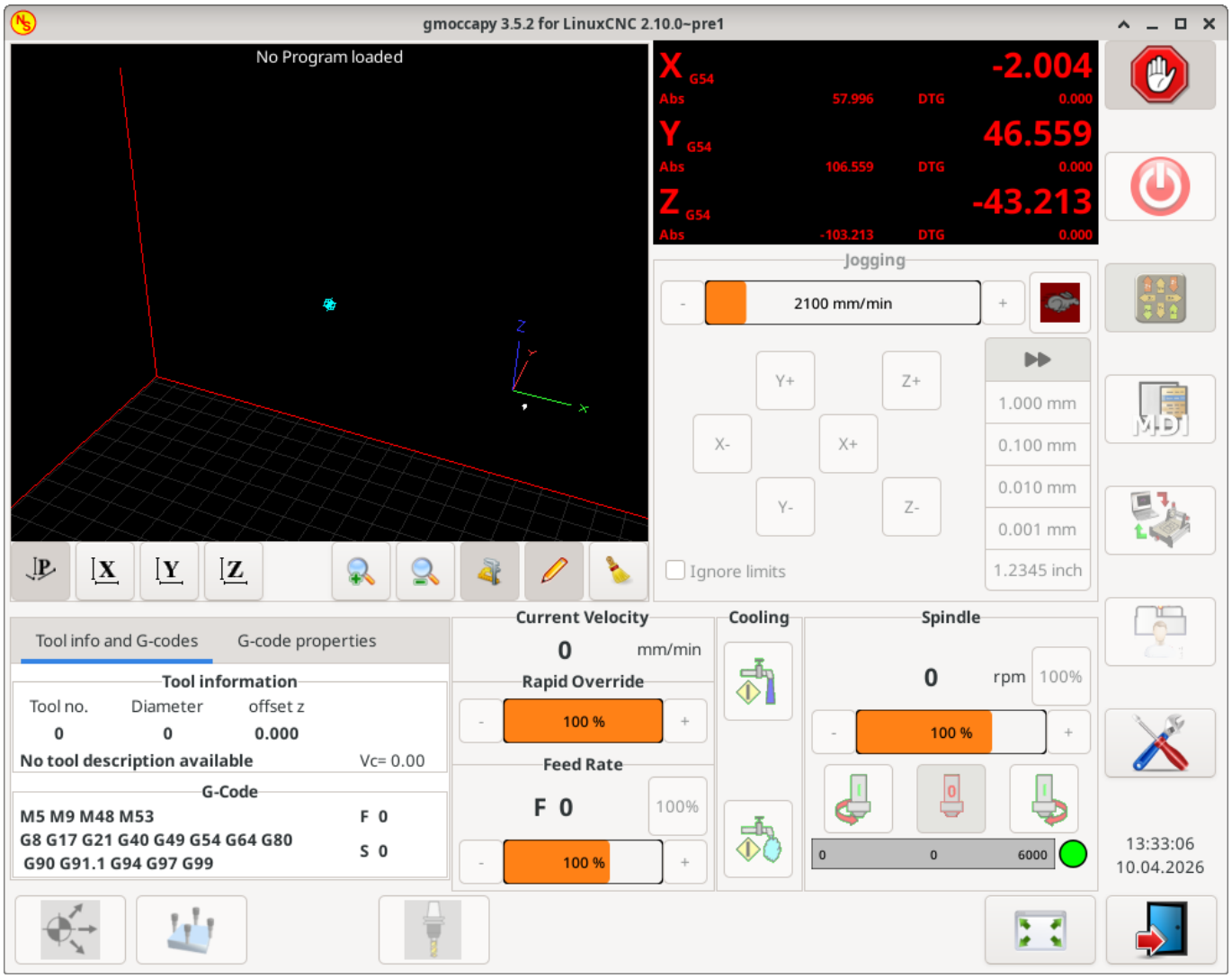


Figura 2.5: GMOCCAPY, a touch screen GUI based on Gscreen

NGCGUI

NGCGUI, a subroutine GUI that provides wizard-style programming of G code. NGCGUI may be run as a standalone program or embedded into another GUI as a series of tabs. The following screenshot shows NGCGUI embedded into AXIS:

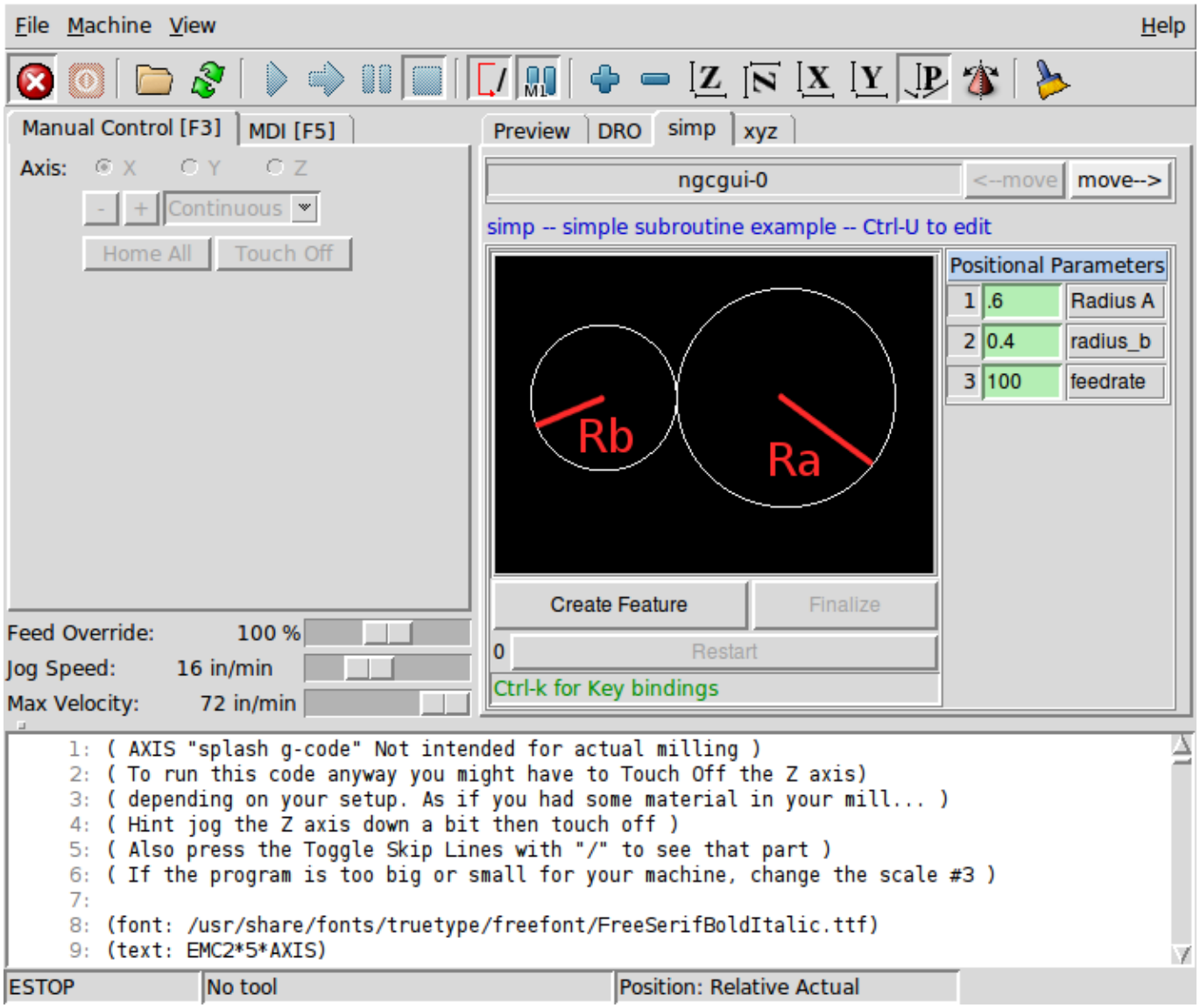


Figura 2.6: NGCGUI, a graphical interface integrated into AXIS

TkLinuxCNC

TkLinuxCNC, another interface based on Tcl/Tk. Once the most popular interface after AXIS.

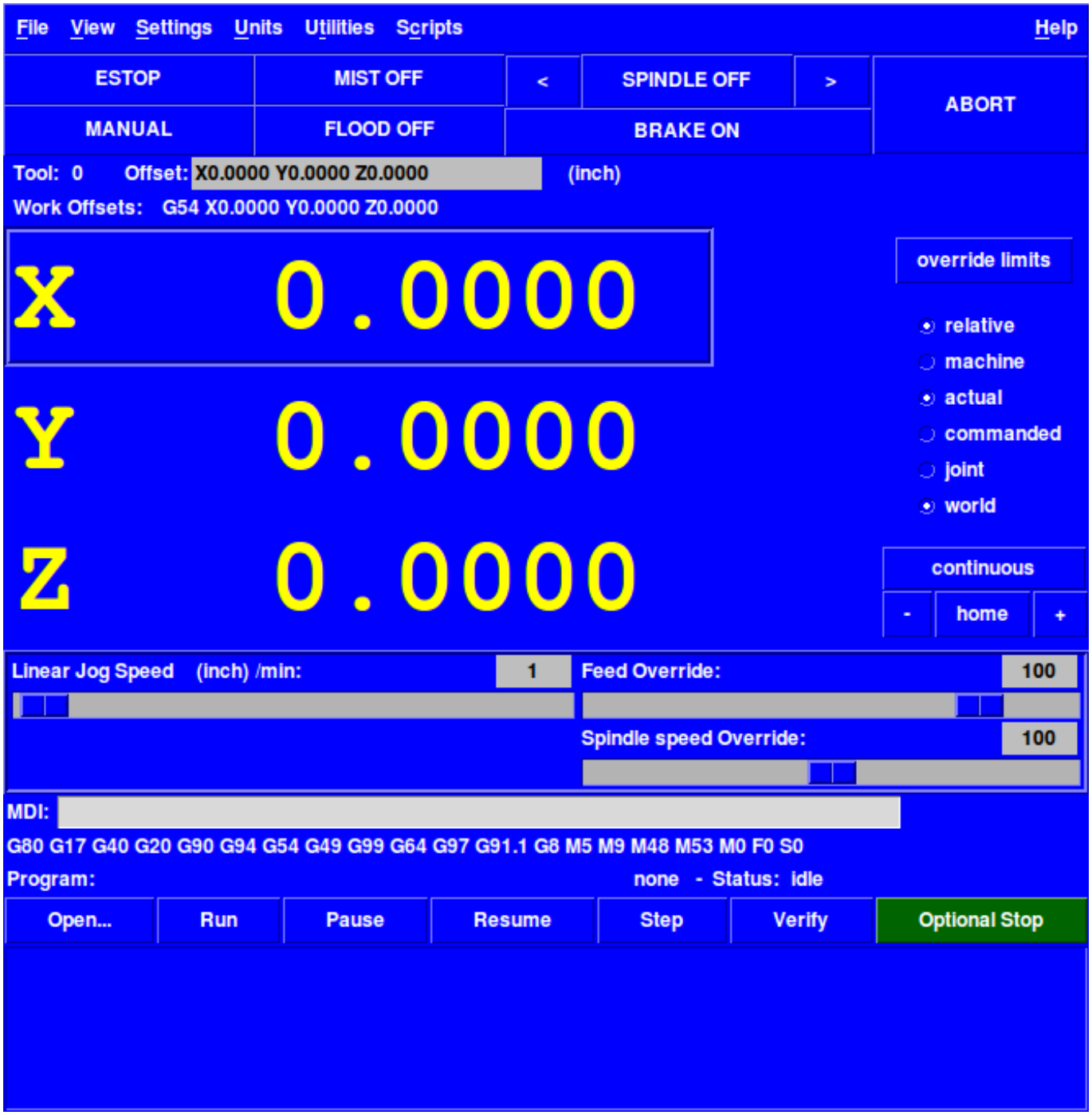


Figura 2.7: TkLinuxCNC graphical interface

QtDragon

QtDragon, una GUI de pantalla táctil basada en QtVCP usando la librería PyQt5. Viene en dos versiones *QtDragon* y *QtDragon_hd*. Son muy similares en cuanto a características, pero QtDragon_hd esta hecha para monitores más grandes.

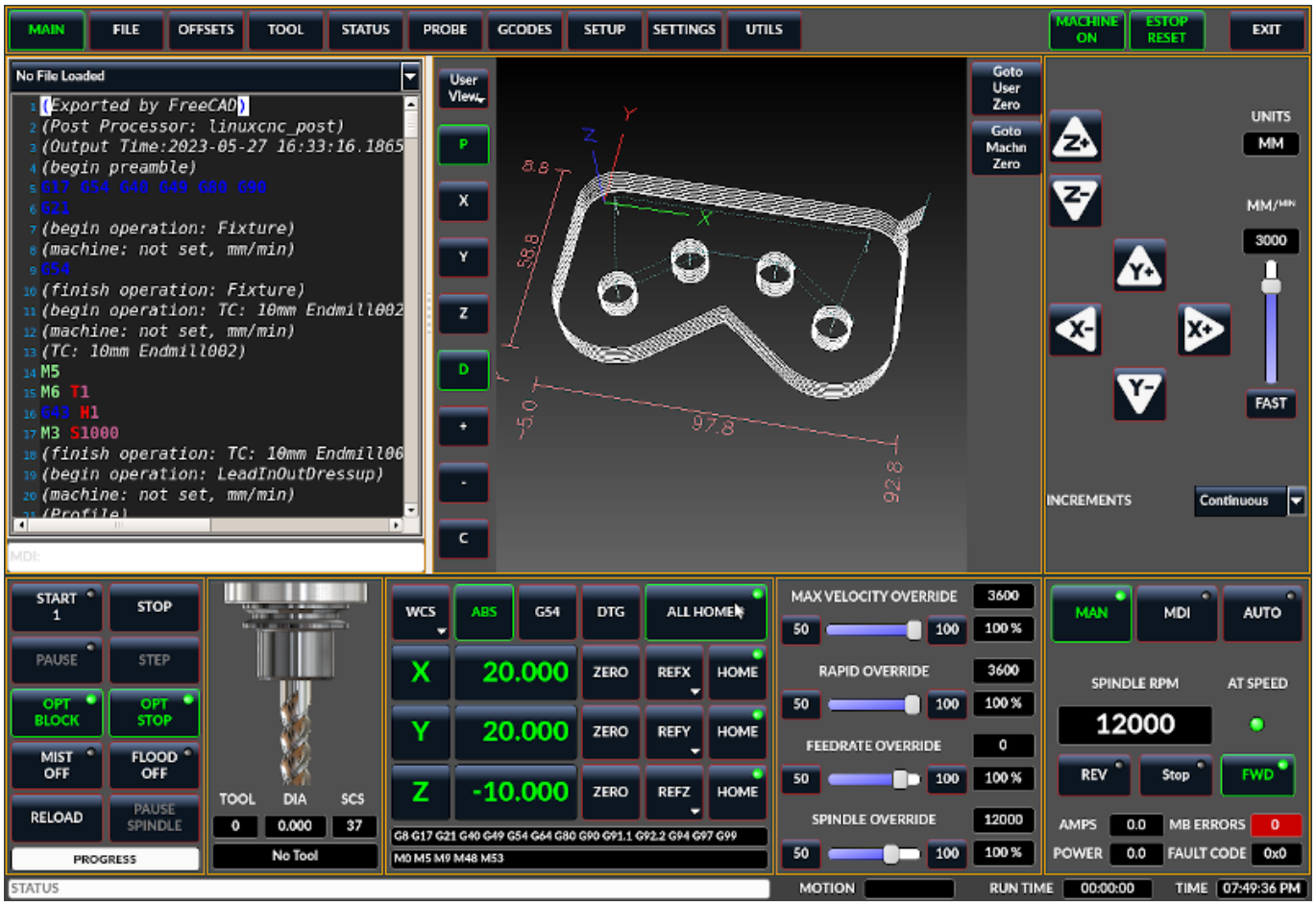


Figura 2.8: QtDragon, a touch screen GUI based on QtVCP

QtPlasmaC

QtPlasmaC, una GUI de pantalla táctil para corte por plasma basada en QtVCP usando la librería PyQt5. Viene en tres relaciones de aspecto: 16:9, 4:3 y 9:16.

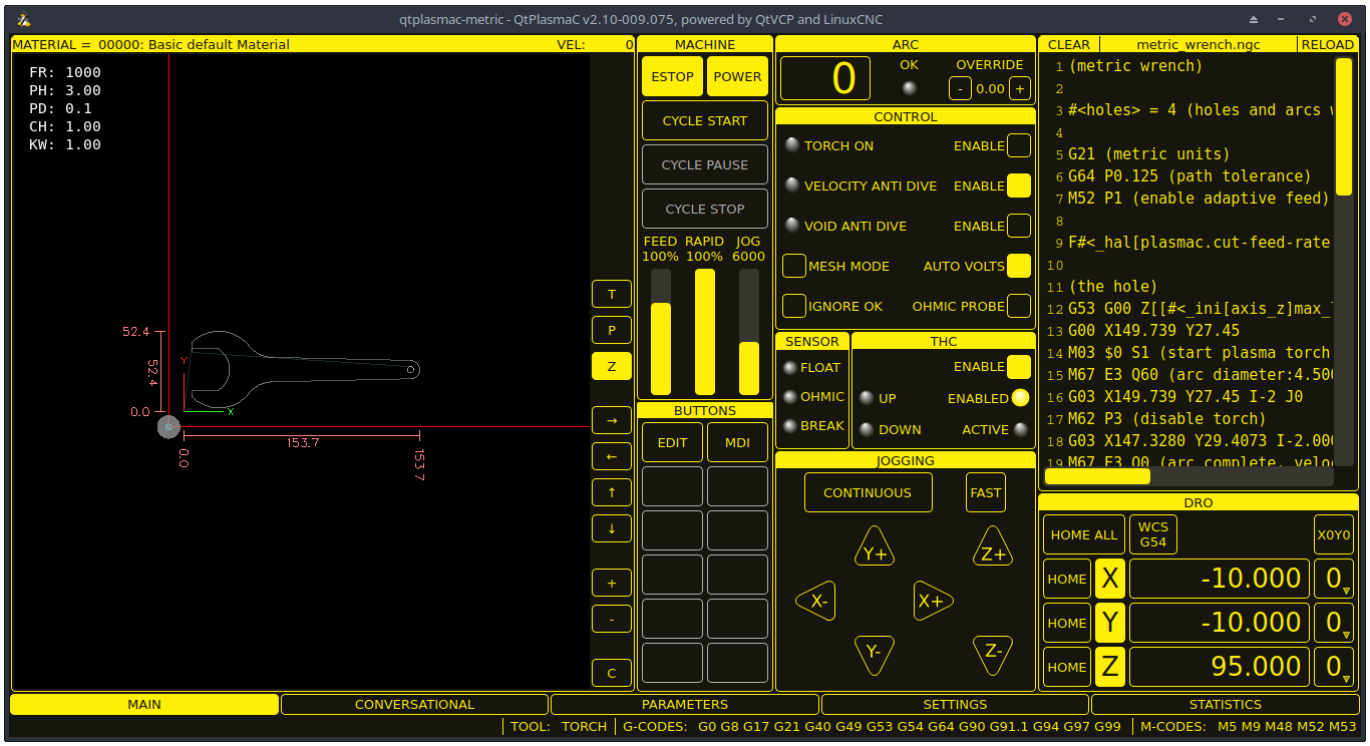


Figura 2.9: QtPlasmaC, a touch screen plasma cutting GUI based on QtVCP

2.1.4. Interfaces de usuario

Estas interfaces de usuario son una manera de interactuar con LinuxCNC fuera de las interfaces de usuario gráficas.

halui
A HAL based user interface allowing to control LinuxCNC using buttons and switches

linuxcncrsh
A telnet based user interface allowing to send commands from remote computers.

2.1.5. Paneles Virtuales de Control

As mentioned above, many of LinuxCNC's GUIs may be customized by the user. This may be done to add indicators, readouts, switches or sliders to the basic appearance of one of the GUIs for increased flexibility or functionality. Two styles of Virtual Control Panel are offered in LinuxCNC:

PyVCP
PyVCP, a Python-based virtual control panel that can be added to the AXIS GUI. PyVCP only utilises virtual signals contained within the Hardware Abstraction Layer, such as the spindle-at-speed indicator or the Emergency Stop output signal, and has a simple no-frills appearance. This makes it an excellent choice if the user wants to add a Virtual Control Panel with minimal fuss.

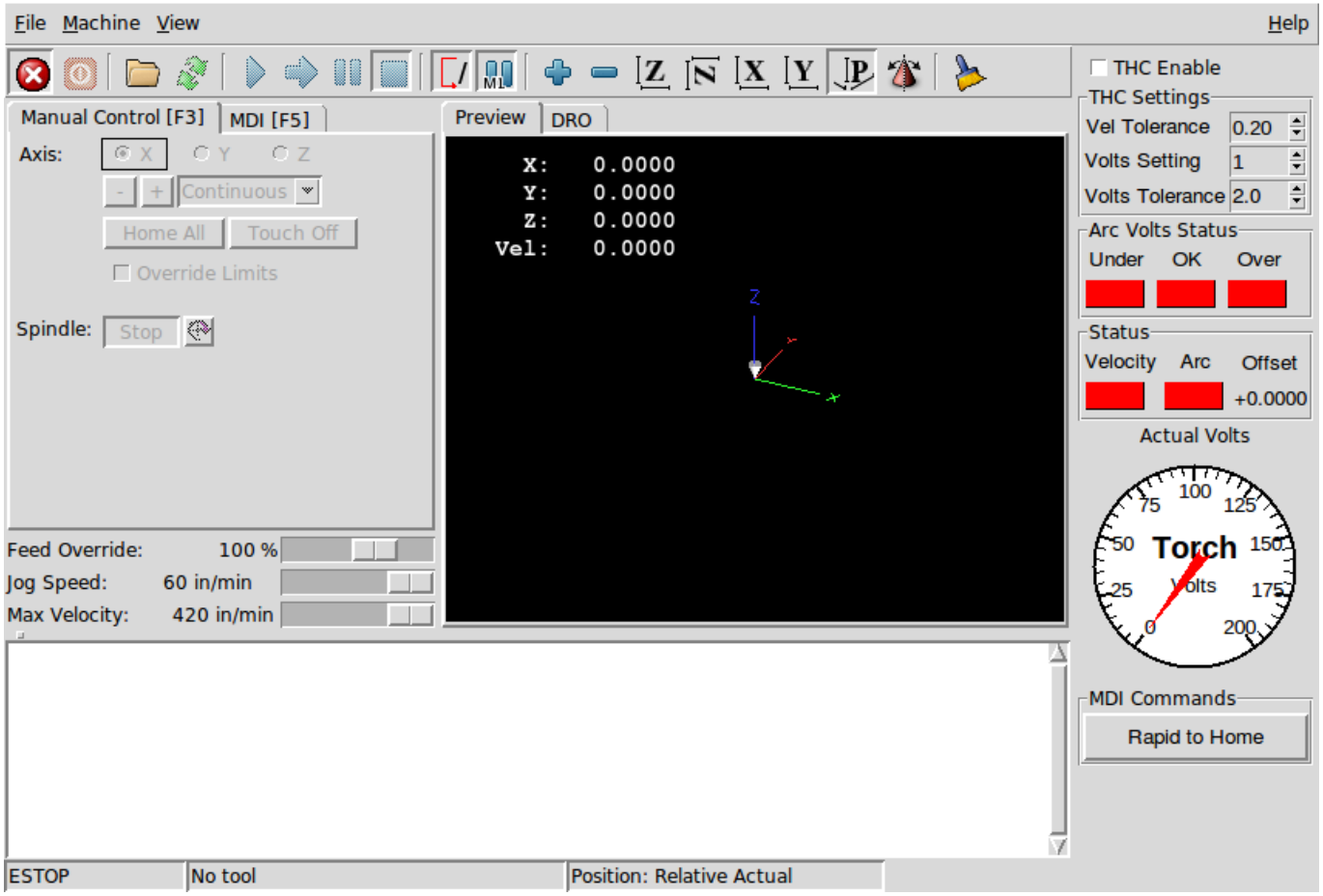


Figura 2.10: PyVCP Example Embedded Into AXIS GUI

GladeVCP

GladeVCP, a Glade-based virtual control panel that can be added to the AXIS or Touchy GUIs. GladeVCP has the advantage over PyVCP in that it is not limited to the display or control of HAL virtual signals, but can include other external interfaces outside LinuxCNC such as window or network events. GladeVCP is also more flexible in how it may be configured to appear on the GUI:

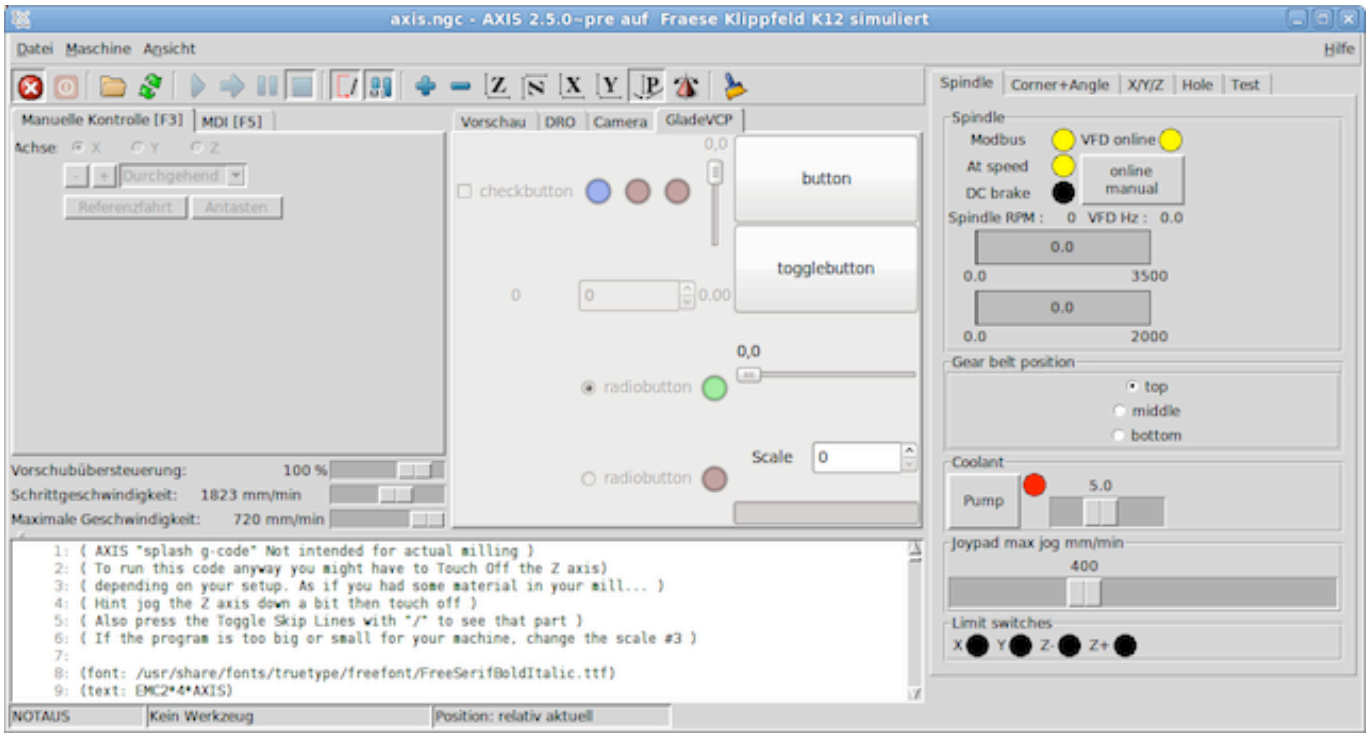


Figura 2.11: GladeVCP Example Embedded Into AXIS GUI

QtVCP

QtVCP, a PyQt5-based virtual control panel that can be added to most GUIs or run as a standalone panel. QtVCP has the advantage over PyVCP in that it is not limited to the display or control of HAL virtual signals, but can include other external interfaces outside LinuxCNC such as window or network events by extending with python code. QtVCP is also more flexible in how it may be configured to appear on the GUI with many special widgets:

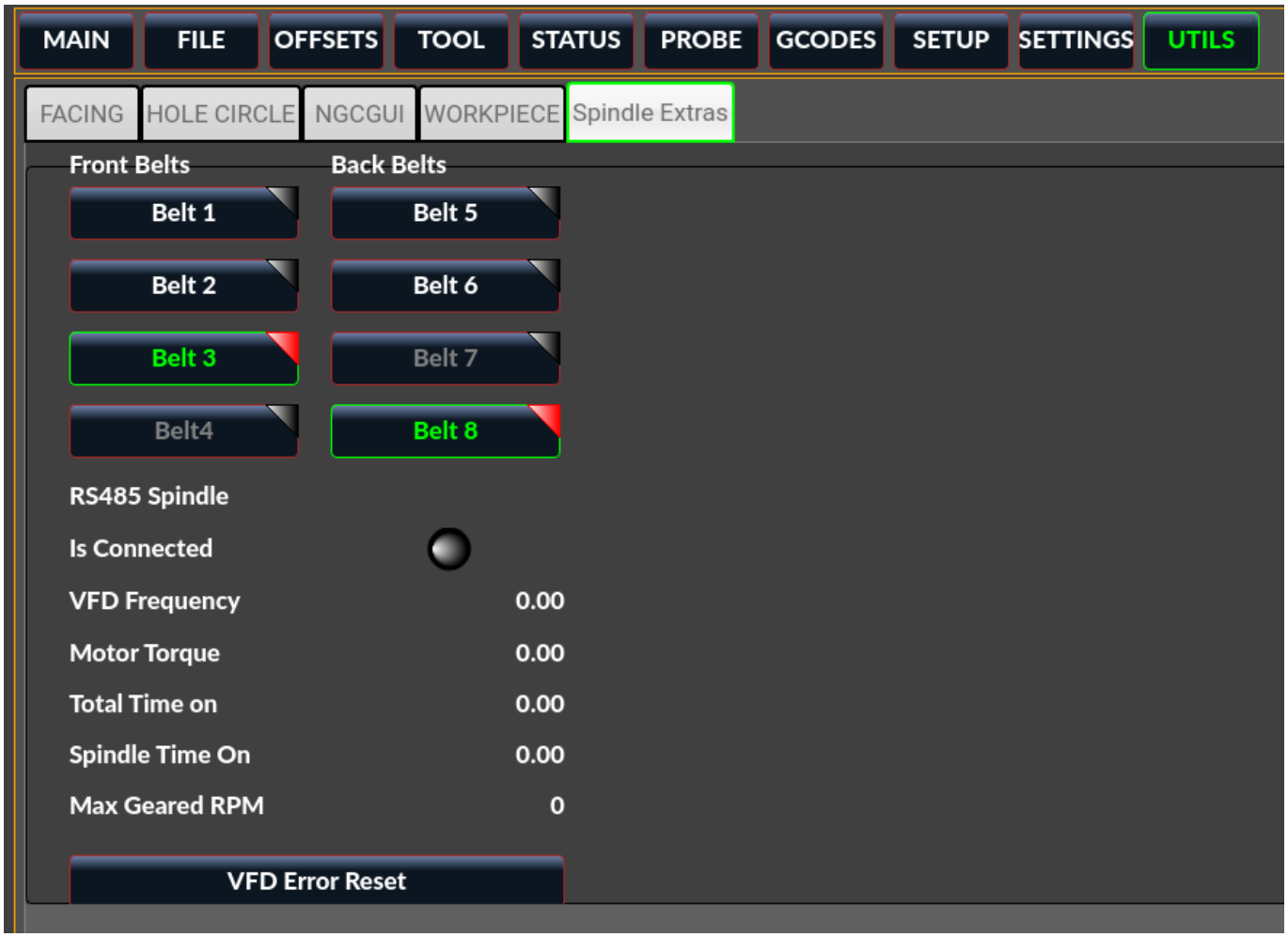


Figura 2.12: QtVCP Example Embedded Into QtDragon GUI

2.1.6. Languages

LinuxCNC uses translation files to translate LinuxCNC User Interfaces into many languages including French, German, Italian, Finnish, Russian, Romanian, Portuguese and Chinese. Assuming a translation has been created, LinuxCNC will automatically use whatever native language you log in with when starting the Linux operating system. If your language has not been translated, contact a developer on IRC, the mailing list or the User Forum for assistance.

2.1.7. Think Like a CNC Operator

This manual does not pretend to teach you how to use a lathe or a milling machine. Becoming an experienced operator takes a lot of time and requires a lot of work. An author once said, *We learn by experience, if one possesses it all*. Broken tools, vices attacked and the scars are evidence of the lessons learned. A beautiful finish, tight tolerances and caution during the work are evidence of lessons learned. No machine nor program can replace human experience.

Now that you start working with the LinuxCNC software, you have to put yourself in the shoes of an operator. You must be in the role of someone in charge of a machine. It’s a machine that will wait for your commands and then execute the orders that you will give it. In these pages, we will give the explanations which will help you to become a good CNC operator with LinuxCNC.

2.1.8. Modes of Operation

When LinuxCNC is running, there are three different major modes used for inputting commands. These are Manual, Auto, and Manual Data Input (MDI). Changing from one mode to another makes a big difference in the way that the LinuxCNC control behaves. There are specific things that can be done in one mode that cannot be done in another. An operator can home an axis in manual mode but not in auto or MDI modes. An operator can cause the machine to execute a whole file full of G-codes in the auto mode but not in manual or MDI.

In manual mode, each command is entered separately. In human terms a manual command might be ”turn on coolant” or ”jog X at 25 inches per minute”. These are roughly equivalent to flipping a switch or turning the hand wheel for an axis. These commands are normally handled on one of the graphical interfaces by pressing a button with the mouse or holding down a key on the keyboard. In auto mode, a similar button or key press might be used to load or start the running of a whole program of G-code that is stored in a file. In the MDI mode the operator might type in a block of code and tell the machine to execute it by pressing the <return> or <enter> key on the keyboard.

Some motion control commands are available concurrently and will cause the same changes in motion in all modes. These include Abort, Emergency Stop, and Feed Rate Override. Commands like these should be self explanatory.

The AXIS user interface hides some of the distinctions between Auto and the other modes by making auto-commands available at most times. It also blurs the distinction between Manual and MDI, because some Manual commands like Touch Off are actually implemented by sending MDI commands. It does this by automatically changing to the mode that is needed for the action the user has requested.

2.2. Conceptos importantes

Este capítulo cubre conceptos importantes que deben ser entendidos por el usuario antes de intentar manejar una máquina CNC con código G.

2.2.1. Control de trayectoria

2.2.1.1. Planificación de trayectoria

La planificación de trayectorias, en general, es el medio por el cual LinuxCNC sigue la ruta especificada por su programa de código G, mientras opera dentro de los límites de la máquina.

Un programa de código G nunca puede ser completamente obedecido. Por ejemplo, imagine que especifica un programa de solo de línea con el siguiente movimiento:

```
G1 X1 F10 (G1 es movimiento lineal, X1 es el destino, F10 es la velocidad)
```

En realidad, todo el movimiento no se puede hacer a F10, ya que la máquina debe acelerar desde el reposo, avanzar hacia X=1, y luego desacelerar para parar de nuevo. A veces, parte del movimiento se realiza a F10, pero para muchos movimientos, especialmente los cortos, nunca se alcanza en absoluto la velocidad de avance especificada. Tener movimientos cortos en su código G puede hacer que su máquina se ralentice y luego acelere para los movimientos mas largos si no se emplea *naive cam detector* con G64 P*n*.

La aceleración y desaceleración básica descrita anteriormente no es compleja y no hay ningún compromiso que tomar. En el archivo INI se han especificado las restricciones de la máquina, como la velocidad y aceleración máxima de ejes, que deben ser obedecidas por el planificador de trayectoria.

Para obtener mas información sobre las opciones INI del planificador vea la sección Trayectoria en el capítulo INI.

2.2.1.2. Seguimiento de ruta

Un problema menos sencillo es el seguimiento de ruta. Cuando se programa una esquina en código G, el planificador de trayectoria puede hacer varias cosas, todas ellas correctas según el caso:

- Puede desacelerar hasta detenerse exactamente en las coordenadas de la esquina, y luego acelerar en la nueva dirección.
- También puede hacer lo que se llama fusión (blending), que consiste en mantener la velocidad de avance mientras pasa por la esquina, por lo que es necesario redondear dicha esquina para obedecer las restricciones de la máquina.

Puede ver que aquí si hay un compromiso; se puede reducir la velocidad para obtener un mejor seguimiento del camino, o mantener la velocidad con peor seguimiento. Dependiendo del corte particular, el material, las herramientas, etc, el programador puede querer comprometerse de una manera u otra.

Los movimientos rápidos también obedecen el control de trayectoria actual. Con movimientos suficientemente largos para alcanzar la velocidad máxima en una máquina con baja aceleración y si no se especifica la tolerancia de ruta, puede obtener una esquina bastante redonda.

2.2.1.3. Programando el planificador

Los comandos de control de trayectoria son los siguientes:

`G61`
(Modo de ruta exacta) `G61` visita el punto programado exactamente, aunque eso signifique que la máquina podría detenerse por completo temporalmente para cambiar de dirección hacia el punto siguiente programado.

`G61.1`
(Modo de parada exacta) `G61.1` le dice al planificador que se detenga exactamente en cada final de segmento. La ruta será seguida exactamente pero parar por completo la alimentación puede ser destructivo para la pieza o la herramienta, dependiendo de las especificaciones de la máquina.

`G64`
(Fusión sin modo de tolerancia) `G64` es la configuración predeterminada cuando se inicia LinuxCNC. G64 es simplemente fusión sin *naive cam detector* habilitado. G64 y G64 P0 le dicen al planificador que sacrifique la precisión de seguimiento de ruta para mantener la velocidad de avance. Esto es necesario para algunos tipos de materiales o herramientas donde las paradas exactas son dañinas y puede funciona bien siempre que el programador no olvide que la ruta de la herramienta será algo más curvilínea que lo que especifica el programa. Al usar movimientos G0 (rápidos) con G64, tenga cuidado con los movimientos a puntos seguros (clearance) y permita suficiente distancia para evitar obstáculos en función de las capacidades de aceleración de su máquina.

`G64 P- Q-`
(Fusión con modo de tolerancia) Esto habilita *naive cam detector* y permite fusiones con tolerancia. Si se programa, por ejemplo, G64 P0.05, se le dice al planificador que se desea alimentación continua, pero en esquinas programadas quiere que desacelere lo suficiente para que la ruta de la herramienta pueda mantenerse a no mas de 0.05 unidades de usuario de la ruta programada. La cantidad exacta de desaceleración depende de la geometría de la esquina programada y las limitaciones de la máquina, pero lo único de lo que el programador necesita preocuparse es de la tolerancia. Esto le da al programador control total sobre el compromiso de seguimiento de camino. La tolerancia de la fusión se puede cambiar a lo largo de todo el programa según sea necesario. Tenga en cuenta que una especificación G64 P0 tiene el mismo efecto que G64 solo, necesario para compatibilidad con los viejos programas de código G. Vea la sección G64 del capítulo de código G.

Fusión sin tolerancia
: El punto controlado tocará cada movimiento especificado en al menos un punto. La máquina nunca se moverá a una velocidad tal que no pueda llegar a una parada exacta al final del movimiento actual (o el próximo movimiento, si hace una pausa cuando la fusión ya ha comenzado). La distancia desde el punto final del movimiento será tan grande como deba ser para mantener la mejor alimentación para el contorno.

Naive CAM Detector
: Varios movimientos G1 sucesivos que involucran solo a los ejes XYZ y que se desvíen menos de una cantidad Q- de una línea recta que los contenga, se fusionaran en una única línea recta. Este movimiento combinado reemplaza los movimientos individuales G1 cuando se quiere fusionar con tolerancia. Entre movimientos sucesivos, el punto controlado no se desviará mas de P- desde los puntos finales reales de los movimientos. El punto controlado tocará al menos un punto en cada movimiento. La máquina nunca se moverá a una velocidad tal que no puede llegar a una parada exacta al final del movimiento actual (o siguiente movimiento, si hace una pausa cuando la fusión ya ha comenzado). En movimientos G2/3, que se muevan en el plano G17 (XY), cuando la desviación máxima de un arco desde un línea recta es menor que la tolerancia Q- de G64, el arco se divide en dos líneas (desde el inicio del arco hasta el punto medio y desde el punto medio hasta el final). Esas líneas están entonces sujetas al algoritmo naive cam detector para líneas. Por lo tanto, casos de línea a arco, arco a arco y arco a línea, así como línea a línea, podrán beneficiarse de *naive cam detector*. Esto mejora el rendimiento de contorneado simplificando el camino.

En la siguiente figura, la línea azul representa la velocidad real de la máquina. Las líneas rojas son la capacidad de aceleración de la máquina. Las líneas horizontales debajo de cada gráfico son el movimiento planificado. La parte superior de la gráfica muestra cómo el planificador de trayectoria ralentizará la máquina cuando se encuentren movimientos cortos para mantenerse dentro de los límites de los ajuste de aceleración de la máquina para poder detenerse exactamente al final del próximo movimiento. La gráfica inferior muestra el efecto de Naive Cam Detector al combinar los movimientos y hacer un mejor trabajo manteniendo el velocidad según lo planeado.

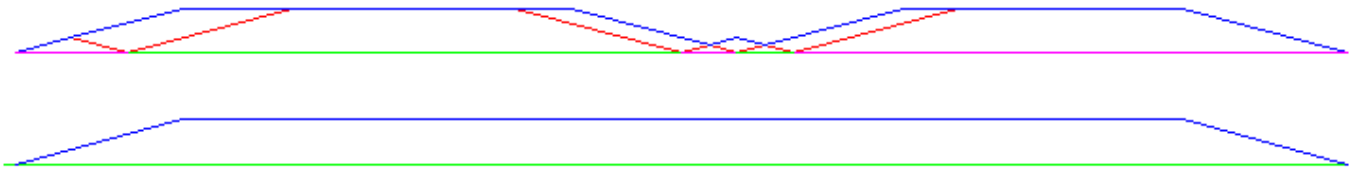


Figura 2.13: Naive CAM Detector

2.2.1.4. Planificando movimientos

Asegúrese de que los movimientos sean *lo suficientemente largos* para adaptarse a su máquina/material. Principalmente debido a la regla de que la máquina nunca se moverá a una velocidad tal que no pueda detenerse por completo al final del movimiento actual, hay una longitud de movimiento mínima que permitirá a la máquina mantener una velocidad de avance solicitada para un ajuste de aceleración dado.

Las fases de aceleración y desaceleración utilizan cada una la mitad de MAX_ACCELERATION del archivo INI. En una combinación que sea una inversión exacta, esto causará que la aceleración total del eje iguale el valor MAX_ACCELERATION del archivo INI. En otros casos, la aceleración real de la máquina es algo menor que esta aceleración INI.

Para mantener la velocidad de alimentación, el movimiento debe ser mas largo que la distancia necesaria para acelerar de cero a la velocidad de avance deseada y luego detenerse de nuevo. Usando

A como **1/2** del valor de MAX_ACCELERATION del archivo INI, y F como velocidad de avance **en unidades por segundo**, el tiempo de aceleración es $\mathbf{t_a = F/A}$ y la distancia de aceleración es $\mathbf{d_a = F*t_a/2}$. El tiempo de desaceleración y distancia son las mismas, haciendo que la distancia crítica sea $\mathbf{d = d_a + d_d = 2 * d_a = F^2/A}$.

Por ejemplo, para una velocidad de avance de 1 pulgada por segundo y una aceleración de **10 pulgadas/seg^2** la distancia crítica es $\mathbf{1^2/10 = 1/10 = 0.1}$ **pulgadas**.

Para una velocidad de avance de 0.5 pulgadas por segundo, la distancia crítica es $\mathbf{5^2/100 = 25/100 = 0.025}$ pulgadas.

2.2.2. Códigos G

2.2.2.1. Valores predeterminados

Cuando LinuxCNC se arranca, se cargan muchos códigos G y M de manera predeterminada. Los códigos actuales activos G y M se pueden ver en la pestaña MDI de la ventana *Códigos G activos:* en la interfaz AXIS. Estos códigos G y M definen el comportamiento de LinuxCNC y es importante que entienda qué hace cada uno de ellos antes de ejecutar LinuxCNC. Los valores predeterminados se pueden cambiar cuando se ejecuta un archivo de código G y pueden quedar en un estado diferente al del comienzo de sesión de LinuxCNC. La mejor práctica es establecer los valores predeterminados necesarios para el trabajo en un preámbulo de su archivo de código G, sin suponer que los valores predeterminados no han cambiado. Imprimir la página de Referencia rápida de códigos G puede ayudarle a recordar qué es cada uno.

2.2.2.2. Velocidad de alimentación

La forma en la que se aplica la velocidad de alimentación depende de si un eje involucrado con el movimiento es un eje giratorio. Lea y comprenda la sección Velocidad de alimentación si tiene un eje giratorio o un torno.

2.2.2.3. Offset del radio de la herramienta

El offset del radio de la herramienta (G41/G42) requiere que la herramienta sea capaz de tocar en algún lugar a lo largo de cada movimiento programado sin estropear los dos movimientos adyacentes. Si eso no es posible con el diámetro actual de la herramienta, se obtendrá un error. Una herramienta de menor diámetro puede trabajar sin error en el mismo camino. Esto significa que puede programar una herramienta para pasar por un camino que es más estrecho que la herramienta sin ningún error. Para más información, ver la sección Compensación del cortador.

2.2.3. Homing

Después arrancar LinuxCNC cada eje debe ser llevado a casa antes de ejecutar un programa o comando MDI. Si su máquina no tiene interruptores home, una marca de coincidencia en cada eje puede ayudar a poner las coordenadas de la máquina en casa en el mismo lugar cada vez. Una vez en casa, se usarán tus límites suaves configurados en el archivo INI.

Si desea desviarse del comportamiento predeterminado, o desea usar el interfaz Mini, necesitará configurar la opción `NO_FORCE_HOMING = 1` en la sección `[TRAJ]` de su archivo INI. Se puede obtener más información sobre homing en el Manual del integrador.

2.2.4. Cambios de herramientas

Hay varias opciones al hacer cambios de herramientas manuales. Ver la Sección [EMCIO] para obtener información sobre la configuración de estas opciones. Ver también las secciones G28 y G30 del capítulo de Código G.

2.2.5. Sistemas de coordenadas

Los sistemas de coordenadas pueden ser confusos al principio. Antes de manejar una máquina CNC debe comprender los conceptos básicos de los sistemas de coordenadas utilizados por LinuxCNC. La información detallada sobre los sistemas de coordenadas de LinuxCNC está en la sección Sistema de coordenadas de este manual.

2.2.5.1. Coordenadas de máquina G53

Al hacer una secuencia homing en LinuxCNC, usted configura el sistema de coordenadas de la máquina G53 a 0 para cada uno de los ejes en la secuencia.

Ningún otro sistema de coordenadas u offsets de herramientas se cambian por la secuencia homing.

La única forma de moverse en el sistema de coordenadas de la máquina es cuando se programa un G53 en la misma línea que el movimiento. Normalmente se está en el sistema de coordenadas G54.

2.2.5.2. Coordenadas de usuario G54-59.3

Normalmente se usa el sistema de coordenadas G54. Cuando se aplica un offset al actual sistema de coordenadas de usuario, una pequeña bola azul con líneas estará en el orígen de máquina cuando su DRO esté mostrando *Posición: Relativa Actual* en AXIS. Si sus offsets son temporales, use Zero Coordinate System del menú de máquina o programe *G10 L2 P1 X0 Y0 Z0* al final de su archivo de código G. Cambie el número *P* para que se adapte al sistema de coordenadas en el que desea borrar el offset.

- Los offsets almacenados en un sistema de coordenadas de usuario se conservan cuando LinuxCNC se apaga.
- Usando el botón *Touch Off* en AXIS, se establece un offset para el Sistema de coordenadas elegido de usuario.

2.2.5.3. Cuando este perdido

Si tiene problemas para obtener 0,0,0 en el DRO cuando piense que debería mostrarse, puede que tenga algunos offsets programados y necesita eliminarlos.

- Muévase al origen de la máquina con G53 G0 X0 Y0 Z0
- Borre cualquier offset de G92 con G92.1
- Use el sistema de coordenadas G54
- Establezca el sistema de coordenadas G54 para que sea el mismo que el sistema de coordenadas máquina con *G10 L2 P1 X0 Y0 Z0 R0*.
- Desactivar los offsets de herramienta con G49
- Activar la pantalla de coordenadas relativas desde el menú

Ahora debe estar en el origen de la máquina X0 Y0 Z0 y el sistema de coordenadas relativo debe ser el mismo que el sistema de coordenadas de máquina.

2.2.6. Configuraciones de la máquina

El siguiente diagrama muestra una fresadora típica que muestra la dirección del recorrido de la herramienta, la mesa y los interruptores de límite. Observe cómo la mesa se mueve en la dirección opuesta a las flechas del sistema de coordenadas cartesianas mostrado por la imagen *Dirección Herramienta*. Esto hace que la herramienta *se mueva* en la dirección correcta en relación con el material.

Note también la posición de los interruptores de límite y la dirección de activación de sus levas. Diversas combinaciones son posibles, por ejemplo, es posible (contrario al dibujo) colocar un interruptor simple fijo en medio de la mesa y dos levas móviles para activarlo. En esta caso los límites estarán invertidos, +X será a al derecha de la mesa y -X a la izquierda. Esta inversión no cambia nada desde el punto de vista de la dirección del movimiento de la herramienta.

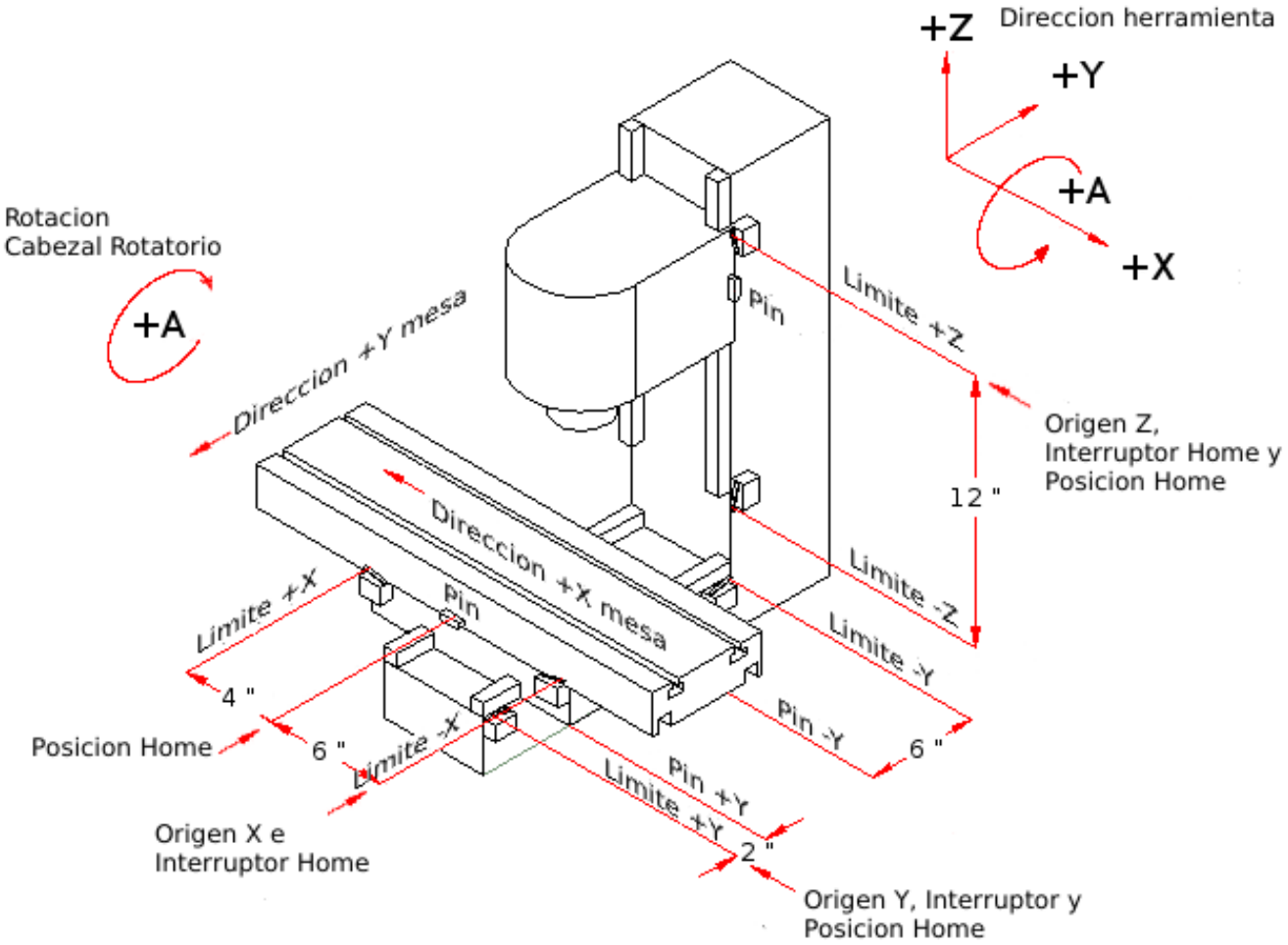


Figura 2.14: Configuración típica de fresadora

El siguiente diagrama muestra un torno típico que muestra la dirección del recorrido de la herramienta y los interruptores de límite.

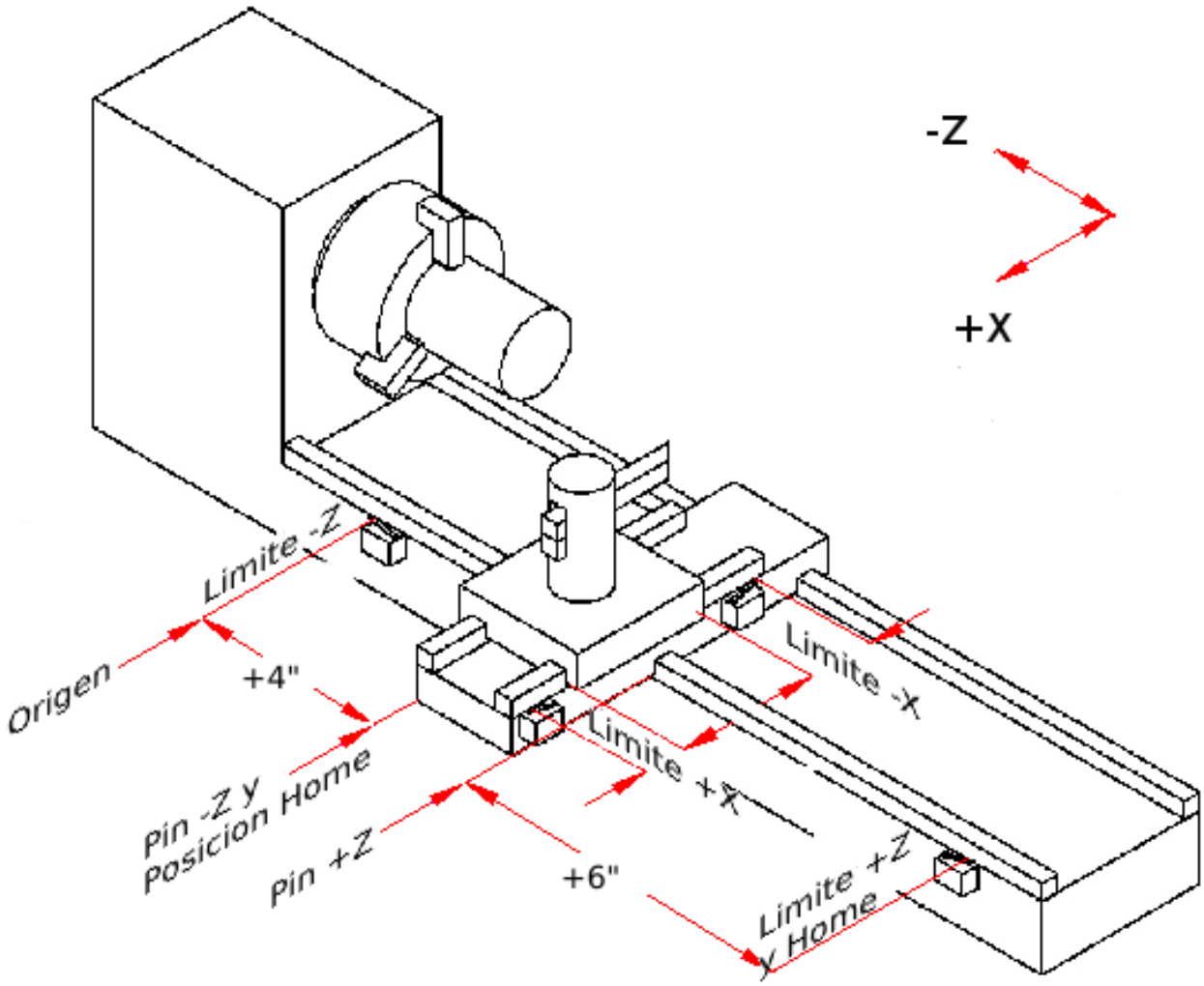


Figura 2.15: Configuración típica de torno

2.3. Iniciando LinuxCNC

2.3.1. Ejecutando LinuxCNC

LinuxCNC se inicia con el archivo script *linuxcnc*.

```
linuxcnc [opciones] [<archivo-ini>]
```

Opciones del script linuxcnc

```
linuxcnc: Ejecuta LinuxCNC

Uso:
  $ linuxcnc -h
    Esta ayuda

  $ linuxcnc [Opciones]
```

```
    Elegir el archivo de configuración INI gráficamente

  $ linuxcnc [Opciones] ruta/a/tu_archivo_ini
    Proporciona el archivo de configuración INI usando su ruta

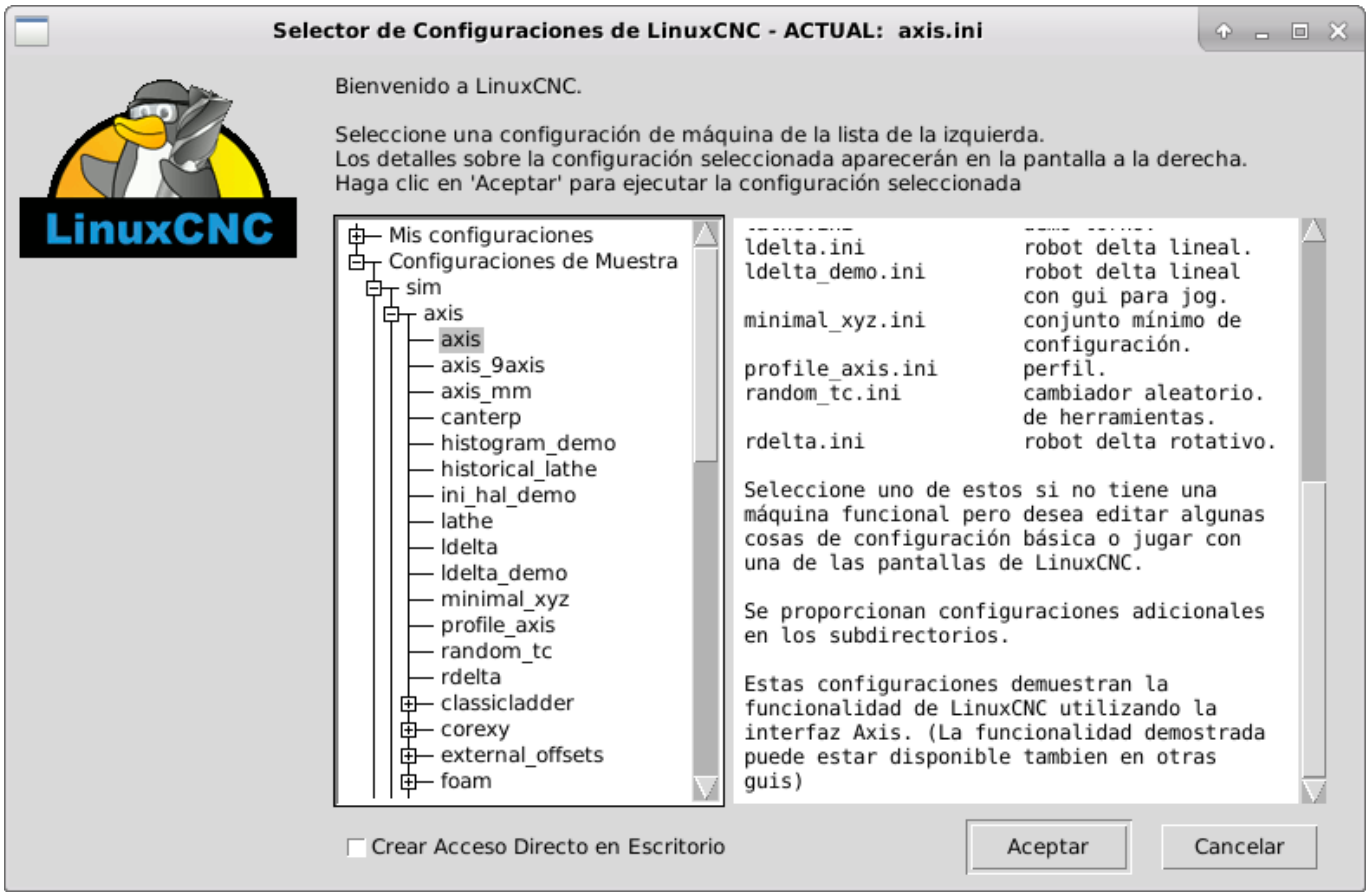
  $ linuxcnc [Opciones] -l
    Usar el archivo de configuración INI usado previamente

Opciones:
    -d: Enciende el modo ”debug”
    -v: Enciende el modo ”verbose”
    -r: Deshabilita la redirección de stdout y stderr a ~/linuxcnc_print.txt y
        ~/linuxcnc_debug.txt cuando stdin no es una tty.
        Usado al correr pruebas de linuxcnc no-interactivamente.
    -l: Usa el último archivo INI usado
    -k: Continúa ante la presencia de errores en archivos HAL
    -t ”nombredemodulotp [parámetros]”
             especifica módulo de planificación de trayectoria personalizado
             anula la configuración opcional INI [TRAJ]TPMOD
    -m ”nombredemodulohome [parámetros]”
             especifica un módulo de homing personalizado
             anula la configuración opcional INI [EMCMOT]HOMEMOD
    -H ”nombredirectorio”: busca archivos HAL en nombredirectorio antes de buscar
                   en directorio INI y librería de sistema:
                   /home/git/linuxcnc-dev/lib/hallib
Nota:
    La opción -H ”nombredirectorio” se puede especificar varias veces
```

Si al script *linuxcnc* se le pasa un archivo .ini como argumento, lo lee y se inicia LinuxCNC. La sección [HAL] del archivo .ini especifica el orden de carga de archivos HAL, si se usa mas de uno. Una vez que se cargan los archivos HAL=xxx.hal, se carga la GUI y luego se carga el archivo POSTGUI=.xxx.hal. Si se han creado objetos PyVCP o GladeVCP con pines HAL, debe usar el archivo HAL postgui para hacer cualquier conexión a esos pines. Consulte la sección [HAL] de la configuración INI para mas información.

2.3.1.1. Selector de configuración

Si no se pasa ningún archivo INI al script linuxcnc, se carga el selector de configuración para que pueda elegir y guardar una configuración de ejemplo. Una vez que una configuración de ejemplo se ha guardado, se puede modificar para adaptarse a su aplicación. Los archivos de configuración se guardan en el directorio linuxcnc/configs.



2.4. Descripción general de una máquina CNC

Esta sección proporciona una breve descripción de una máquina CNC desde los puntos de vista de entrada y salida del intérprete.

2.4.1. Componentes mecánicos

Una máquina CNC tiene muchos componentes mecánicos que pueden ser controlados o que pueden afectar a la forma en que se ejerce el control. Esta sección describe el subconjunto de aquellos componentes que interactúan con el intérprete. Los componentes mecánicos que no interactúan directamente con el intérprete, como los botones de jog, no se describen aquí, incluso si afectan al control.

2.4.1.1. Ejes

Cualquier máquina CNC tiene uno o más ejes. Diferentes tipos de máquinas CNC pueden tener diferentes combinaciones. Por ejemplo, una *fresadora de 4 ejes* puede tener ejes XYZA o XYZB. Un torno normalmente tiene los ejes XZ. Una máquina de corte de espuma puede tener ejes XYUV. En LinuxCNC, el caso de una máquina *pórtico* XYYZ, con dos motores para un eje, se maneja mejor con la cinemática en lugar de con un segundo eje lineal.

nota
Si el movimiento de los componentes mecánicos no es independiente, como en máquinas hexapod, el lenguaje RS274/NGC y las funciones de mecanizado canónicas seguirán siendo utilizables, siempre que los niveles inferiores de control sepan cómo controlar los mecanismos reales para producir el mismo movimiento relativo de herramienta y pieza de trabajo como el que se produciría por ejes independientes. Esto se llama *cinemática*.

nota
Con LinuxCNC, el caso de máquina pórtico XYYZ con dos motores para un eje se maneja mejor con la cinemática en lugar de con un segundo eje lineal.

.Primary Linear Axes The X, Y, and Z axes produce linear motion in three mutually orthogonal directions.

.Secondary Linear Axes The U, V, and W axes produce linear motion in three mutually orthogonal directions. Typically, X and U are parallel, Y and V are parallel, and Z and W are parallel.

.Rotational Axes The A, B and C axes produce angular motion (rotation). Typically, A rotates around a line parallel to X, B rotates around a line parallel to Y, and C rotates around a line parallel to Z.

2.4.1.2. Husillo

Una máquina CNC tiene generalmente un husillo que sostiene una herramienta de corte, una sonda, o el material en el caso de un torno. El husillo puede ser controlado por el software CNC. LinuxCNC soporta hasta 8 husillos, los cuales pueden ser controlados individualmente y pueden trabajar simultáneamente a diferentes velocidades y direcciones.

2.4.1.3. Refrigerante

Cada refrigerante por niebla e inundación puede funcionar independientemente. El lenguaje RS274/NGC los apaga en conjunto, vea la sección M7 M8 M9.

2.4.1.4. Porcentaje de alimentación y velocidad

Una máquina CNC puede tener controles separados de imposición de velocidad y avance, que permiten al operador especificar la velocidad de alimentación real, o la velocidad utilizada en el husillo durante el maquinado, en función de un porcentaje de la velocidad programada.

2.4.1.5. Interruptor de borrado de bloque

Una máquina CNC puede tener un interruptor de eliminación de bloques de código G. Vea la sección Eliminar Bloque.

2.4.1.6. Interruptor de parada de programa opcional

Una máquina CNC puede tener un interruptor de parada opcional de programa. Ver la sección Parada opcional de programa.

2.4.2. Componentes de control y datos

2.4.2.1. Ejes lineales

Los ejes X, Y y Z forman un sistema de coordenadas estándar de mano derecha de ejes lineales ortogonales. Las posiciones de los tres mecanismos de movimiento lineal se expresan usando coordenadas en estos ejes.

Los ejes U, V y W también forman un sistema de coordenadas diestro estándar. X y U son paralelos, Y y V son paralelos, y Z y W son paralelos (cuando A, B y C se rotan a cero).

2.4.2.2. Ejes de rotación

Los ejes de rotación se miden en grados como ejes lineales de revolución en los que la dirección de la rotación positiva es en sentido antihorario cuando se ven desde el extremo positivo del eje X, Y o Z correspondiente. Por *eje lineal de revolucion*, nos referimos a uno en el que la posición angular puede aumentar sin límite (va hacia más-infinito) a medida que el eje gira en sentido antihorario y decrece sin límite (va hacia menos-infinito) a medida que el eje gira en el sentido de las agujas del reloj. Se utilizan ejes lineales de revolucion independientemente de si hay o no un límite mecánico en la rotación.

El sentido horario o antihorario se toma desde el punto de vista de la pieza de trabajo. Si la pieza de trabajo está sujeta a un plato giratorio que gira en un eje de rotación, un giro en sentido antihorario desde el punto de vista de la pieza de trabajo se logra girando el plato giratorio en una dirección que (para la mayoría de las configuraciones de máquina comunes) se ve en el sentido de las agujas del reloj desde el punto de vista de alguien parado al lado de la máquina. [3]

2.4.2.3. Punto controlado

El punto controlado es el punto cuya posición y velocidad de movimiento están controlados. Cuando el offset de la longitud de la herramienta es cero (el valor predeterminado), este es un punto en el eje del husillo (a menudo llamado punto calibrado) que es una distancia fija mas allá del final del husillo, generalmente cerca del extremo del portaherramientas que encaja en el husillo. La ubicación del punto controlado se puede mover a lo largo del eje del husillo especificando una cantidad positiva para el offset de la longitud de la herramienta. Esta cantidad es normalmente la longitud de la herramienta de corte en uso, por lo que el punto controlado está al final de la herramienta de corte. En un torno, los offsets de longitud de herramienta se pueden especificar para los ejes X y Z, y el punto controlado esta en la punta de la herramienta o ligeramente fuera de ella (donde se intersectan las líneas perpendiculares al eje que tocan el *frente* y *flanco* de la herramienta).

2.4.2.4. Movimiento lineal coordinado

Para manejar una herramienta a lo largo de una ruta especifica, un centro de mecanizado debe coordinar el movimiento de varios ejes. Usamos el término *movimiento lineal coordinado* para describir la situación en la que, nominalmente, cada eje se mueve a velocidad constante y todos los ejes se mueven desde sus posiciones iniciales a sus posiciones finales al mismo tiempo. Si solo los ejes X, Y y Z (o uno o dos de ellos) se mueven, se produce movimiento en una línea recta, de ahí la palabra *lineal* en el término. En movimientos reales, a menudo no es posible mantener la velocidad constante porque se requiere aceleración o desaceleración al comienzo y/o al final del movimiento. Sin embargo, es factible controlar los ejes para que, en todo momento, cada eje haya completado la misma fracción del movimiento requerido que los otros ejes. Esto mueve la herramienta a lo largo de la misma ruta, y también llamamos a este tipo de movimiento *movimiento lineal coordinado*.

[3] si se viola el requisito de paralelismo, el creador del sistema tiene que decir cómo distinguir entre sentido horario y antihorario.

El movimiento lineal coordinado se puede realizar a la velocidad de avance prevalente, o a la velocidad de cruce, o puede estar sincronizado con la rotación del husillo. Si los límites físicos en la velocidad del eje hacen que la tasa deseada sea inalcanzable, todos los ejes se ralentizan para mantener el camino deseado.

2.4.2.5. Velocidad de alimentación

La velocidad a la que se mueve el punto controlado es, nominalmente, la velocidad estable que puede establecer el usuario. En el intérprete, la tasa de alimentación se interpreta de la siguiente manera (a menos que se usen los modos *alimentación inversa al tiempo* o *alimentación por revolución*, en cuyo caso, consulte la sección G93-G94-G95-Mode).

1. Si X, Y o Z se mueven, F está en unidades por minuto en el sistema cartesiano XYZ, y todos los demás ejes (ABCUVW) se mueven para arrancar y parar de manera coordinada.
2. De lo contrario, si U, V o W se mueven, F está en unidades por minuto en el sistema cartesiano UVW y todos los demás ejes (ABC) se mueven para arrancar y parar de manera coordinada.
3. De lo contrario, el movimiento es giratorio puro y la palabra F está en unidades de rotación en el sistema *pseudo-cartesiano* ABC.

2.4.2.6. Enfriamiento

El refrigerante de inundación y el refrigerante de niebla pueden encenderse independientemente. El lenguaje RS274/NGC los apaga juntos. Ver sección control de refrigerante.

2.4.2.7. Espera

Se puede ordenar que un centro de mecanizado haga dwell (es decir, mantenga todos los ejes inmóviles) durante una cantidad específica de tiempo. El uso más común de dwell es romper y despejar las virutas, por lo que el husillo suele girar durante un dwell. Independientemente del modo de control de ruta (ver la sección control de ruta) la máquina se detendrá exactamente al final del movimiento programado anterior, como si estuviera en modo de ruta exacta.

2.4.2.8. Unidades

Las unidades utilizadas para distancias a lo largo de los ejes X, Y y Z pueden medirse en milímetros o pulgadas. Las unidades para todas las demás cantidades involucradas en el control de la máquina no puede ser cambiadas. Diferentes cantidades usan diferentes unidades específicas. La velocidad del husillo se mide en revoluciones por minuto. Las posiciones de los ejes de rotación se miden en grados. La velocidad de alimentación se expresan en unidades actuales de longitud por minuto, o grados por minuto, o unidades de longitud por revolución del husillo, como se describe en la sección G93 G94 G95.

2.4.2.9. Posición actual

El lugar donde en cualquier momento se encuentra el punto controlado se llama *posición actual*, y el controlador siempre conoce dónde está ese punto. Los números que representan la posición actual deben ajustarse si, en ausencia de cualquier movimiento del eje, ocurre alguno de estos eventos:

1. Se cambian las unidades de longitud.
2. El offset de la longitud de la herramienta ha cambiado.
3. Se modifican los offsets del sistema de coordenadas.

2.4.2.10. Plano seleccionado

Siempre hay un *plano seleccionado*, que debe ser el plano XY, el YZ, o el XZ del centro de mecanizado. El eje Z es, por supuesto, perpendicular al plano XY, el eje X al plano YZ, y el eje Y al plano XZ.

2.4.2.11. Carrusel de herramientas

Se asigna cero o una herramienta a cada ranura en el carrusel de herramientas.

2.4.2.12. Cambio de herramienta

Se puede ordenar a un centro de mecanizado que cambie las herramientas.

2.4.2.13. Pallet Shuttle

Los dos palets pueden intercambiarse por comando.

2.4.2.14. Porcentaje de velocidad

Los botones de porcentaje de velocidad pueden ser activados (como funcionan normalmente) o volverlos inoperantes (ya no tendrán efecto alguno). El lenguaje RS274/NGC tiene un comando para activar todos los botones y otro para desactivarlos. Ver inhibición y activación de correctores de velocidad. Ver también aquí para mas detalles.

2.4.2.15. Modo de control de ruta

El centro de mecanizado puede colocarse en cualquier modo de control de ruta entre estos tres:

modo de parada exacta
: En el modo de parada exacta, la máquina se detiene brevemente al final de cada movimiento programado.

modo de ruta exacta
: En modo de ruta exacta, la máquina sigue la ruta programada lo más exactamente posible, ralentizandose o deteniendose, si es necesario en las esquinas agudas del camino.

modo continuo
: En modo continuo, las esquinas de la ruta pueden ser redondeadas ligeramente para que la velocidad de alimentación pueda mantenerse actualizada (pero no mas que la tolerancia, si se ha especificado).

Ver las secciones G61 y G64.

2.4.3. Interacción del intérprete con interruptores

El intérprete interactúa con varios interruptores. Esta sección describe las interacciones con mas detalle. En ningún caso el intérprete sabe cuál es la configuración de cualquiera de estos interruptores.

2.4.3.1. Cancelación de interruptores de avance y velocidad

El intérprete procesará los comandos RS274/NGC que habilitan *M48* y deshabilitan *M49* los interruptores de imposición de alimentación y velocidad. Para ciertos movimientos, como el cruce de salida al final de un hilo durante un ciclo de roscado, los interruptores se deshabilitan automáticamente.

LinuxCNC reacciona a la anulación de la configuración de alimentación y velocidad cuando estos interruptores están habilitados.

Consulte la sección Interruptores M48-M49 para obtener mas información.

2.4.3.2. Interruptor de borrado de bloque

Si el interruptor de borrado de bloque está activado, las líneas de código G que comienzan con una barra inclinada (el carácter de *borrar bloque*) no se interpretan. Si el interruptor está apagado, tales líneas sí son interpretadas. Normalmente, este interruptor debe activarse antes de iniciar el programa NGC.

2.4.3.3. Interruptor de parada de programa opcional

Si este interruptor está activado y se encuentra un código M1, la ejecución del programa entra en pausa.

2.4.4. Tabla de herramientas

Para usar el intérprete se requiere una tabla de herramientas. El archivo dice que herramientas están en qué ranuras de un cambiador de herramientas y cuál es el tamaño y tipo de cada herramienta. El nombre de la tabla de herramientas se define en el archivo INI:

```
[EMCIO]
# archivo de tabla de herramientas
TOOL_TABLE = tooltable.tbl
```

El nombre de archivo predeterminado probablemente se parezca a lo anterior, pero es posible que prefiera darle a su máquina su propia tabla de herramientas, utilizando el mismo nombre en su archivo INI, pero con una extensión tbl:

```
TOOL_TABLE = acme_300.tbl
```

o:

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

Para obtener mas información sobre los detalles del formato de la tabla de herramientas, vea la sección Formato de la tabla de herramientas.

2.4.5. Parámetros

Bajo el punto de vista del lenguaje RS274/NGC, un centro de mecanizado mantiene una matriz de parámetros numéricos definida por el valor de sistema (RS274NGC_MAX_PARAMETERS). Muchos de ellos tienen usos específicos, especialmente en la definición de sistemas de coordenadas. La cantidad de parámetros numéricos puede aumentar a medida que el desarrollo agrege soporte para nuevos parámetros. La matriz persiste con el tiempo, incluso si el centro de mecanizado está apagado. LinuxCNC usa un archivo de parámetros para asegurar la persistencia y le da al intérprete la responsabilidad de mantener el archivo. El intérprete lee el archivo cuando se inicia y lo escribe cuando se cierra.

Todos los parámetros están disponibles para su uso en programas de código G.

El formato de un archivo de parámetros se muestra en la siguiente tabla. El archivo consiste en cualquier cantidad de líneas de encabezado, seguidas por una línea en blanco, seguidas por cualquier cantidad de líneas de datos. El intérprete omite las líneas de encabezado. Es importante que haya exactamente una línea en blanco (sin espacios ni tabuladores) antes de los datos. La línea de encabezado que se muestra en la siguiente tabla describe las columnas de datos, por lo que se sugiere (pero no es obligatorio) que esa línea siempre se incluya en el encabezamiento.

El intérprete solo lee las dos primeras columnas de la tabla. La tercer columna, *Comentario*, no es leída por el intérprete.

Cada línea del archivo contiene el número de índice del parámetro en la primera columna y, en la segunda columna, el valor al que ese parámetro debe establecerse. El valor se representa como un número flotante de doble precisión dentro del intérprete, pero el punto decimal no es obligatorio en el archivo. Todos los parámetros que se muestran en la siguiente tabla son parámetros requeridos y deben ser incluidos en cualquier archivo de parámetros, excepto cualquier parámetro que represente un valor de eje de rotación para un eje no utilizado, que puede omitirse. Se señalará un error si falta algún parámetro requerido. El archivo puede incluir cualquier otro parámetro, siempre que su número esté en el rango de 1 a 5400. Los números de los parámetros se deben organizar en orden ascendente; si no lo estan, se señalará un error. Cualquier parámetro incluido en el archivo leído por el intérprete se incluirá en el archivo que se escriba cuando se cierre. El archivo original se guarda como un archivo de copia de seguridad cuando se escribe el nuevo archivo. Los comentarios no se conservan cuando se escribe el archivo.

Cuadro 2.1: Formato de archivo de parámetros

Numeración de parámetros	**Valor del parámetro**	**Comentarios**
5161	0.0	G28 Home X
5162	0.0	G28 Home Y

Vea la sección Parámetros para mas información.

2.5. Información para usuarios de torno

Este capítulo reunirá información específica para tornos.

2.5.1. Modo torno

Si su máquina CNC es un torno, hay algunos cambios específicos que probablemente querrá hacer a su archivo INI para obtener los mejores resultados de LinuxCNC.

Si está utilizando la GUI AXIS, haga que AXIS muestre sus herramientas de torno correctamente. Consulte la sección configuración INI para obtener más detalles.

Para configurar AXIS para el modo torno.

```
[DISPLAY]

# Informar a Axis que nuestra máquina es un torno.
LATHE = TRUE
```

El modo torno en AXIS no establece su plano predeterminado en G18 (XZ). Debe programarlo en el preámbulo de cada archivo de código G o (mejor) agregarlo a su archivo INI, como:

```
[RS274NGC]

# códigos G modales (modos) con los que se inicializa el intérprete
# en el arranque
RS274NGC_STARTUP_CODE = G18 G20 G90
```

Si está usando GMOCCAPY entonces vea la sección del torno de GMOCAPPY.

2.5.2. La tabla de herramientas

La ”Tabla de herramientas” es un archivo de texto que contiene información sobre cada herramienta. El archivo se encuentra en el mismo directorio que su configuración y de forma predeterminada se denomina ”tool.tbl”. Las herramientas pueden estar en un cambiador de herramientas o ser cambiadas manualmente. El archivo puede editarse con un editor de texto o actualizarse con G10 L1, L10, L11. También hay un editor de tablas de herramientas incorporado en la pantalla AXIS. El número máximo de entradas en la tabla de herramientas es 56. El número máximo de herramienta y de ranura es 99999.

Las versiones anteriores de LinuxCNC tenían dos formatos de tablas de herramientas diferentes para fresadoras y tornos, pero desde la versión 2.4.x, se utiliza un formato de tabla de herramientas para todas las máquinas. Simplemente ignore las partes de la tabla de herramientas que no pertenecen a su máquina, o que no necesite usar. Para obtener más información sobre los detalles del formato de tabla de herramientas, vea la sección Tabla de herramientas.

2.5.3. Orientaciones de herramienta

La siguiente figura muestra las orientaciones de la herramienta de torno con el ángulo de la línea central (l.c.) de cada orientación e información sobre FRONTANGLE y BACKANGLE.

FRONTANGLE y BACKANGLE están en el sentido de las agujas del reloj comenzando en una línea paralela a Z+.

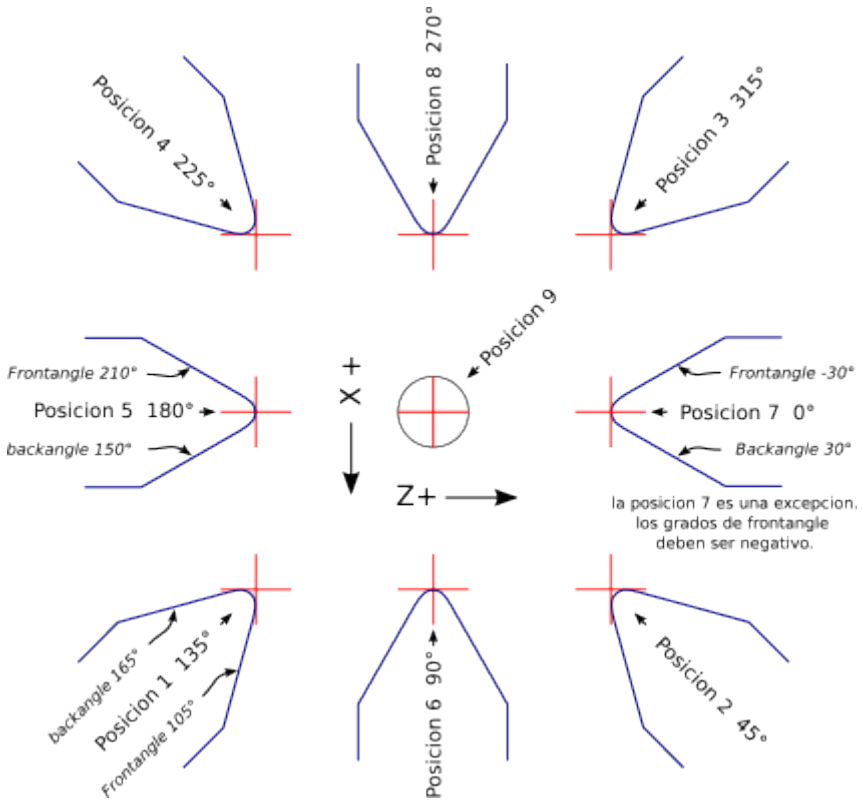
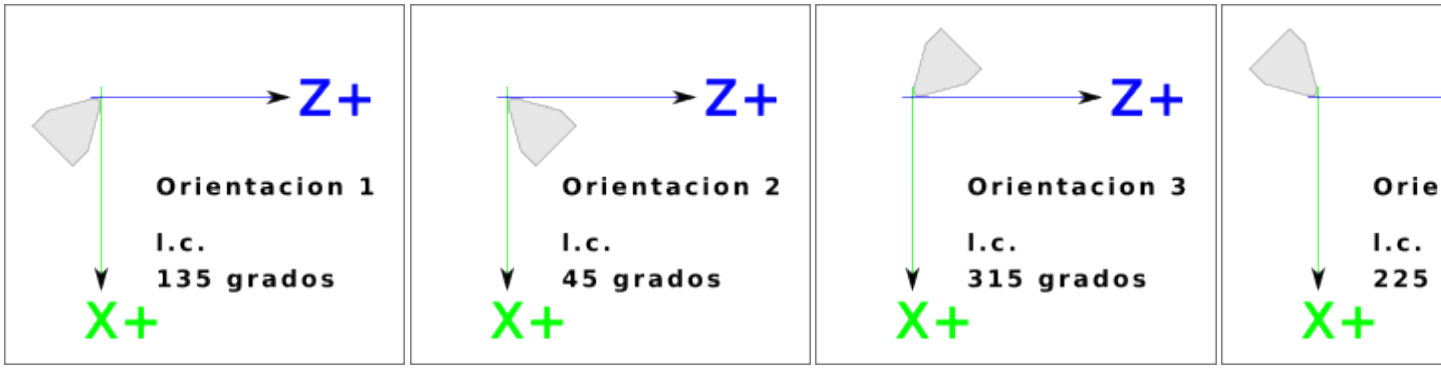


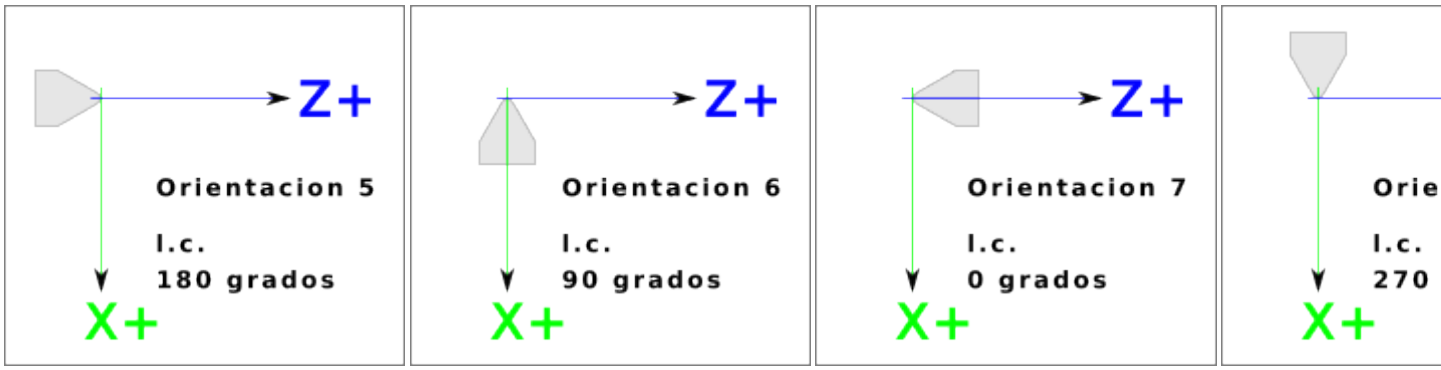
Figura 2.16: Orientaciones de herramienta de torno

En AXIS, las figuras siguientes muestran cómo se ven las posiciones de las herramientas, tal como se ingresaron en la tabla de herramientas.

Tool Positions 1, 2, 3 & 4



Tool Positions 5, 6, 7 & 8



2.5.4. Toque inicial de la herramienta

Cuando AXIS se ejecuta en modo de torno, puede configurar X y Z en la tabla de herramientas mediante la ventana Touch Off. Si tiene una torreta de herramientas normalmente tendrá seleccionado *Touch off to fixture* al configurar su torreta. Cuando configura Z cero del material tendrá seleccionado *Touch off to material*. Para obtener más información sobre los códigos G utilizados para las herramientas, consulte M6, Tn, y G43. Para obtener más información sobre las opciones touch off de la herramienta en AXIS, consulte Tool Touch Off.

2.5.4.1. Toque inicial X

El offset del eje X para cada herramienta es normalmente un desplazamiento desde la línea central del husillo.

Un método es tomar su herramienta de torneado normal y preparar algunas piezas a un diámetro conocido. Usando la ventana Touch Off, ingrese el diámetro medido (o radio si está en modo radio) para esa herramienta. Luego, use liquido de marcado o un rotulador para cubrir la pieza. Lleve cada herramienta hasta que toque solo el tinte y establezca su offset X con touch off en el diámetro de la pieza utilizada. Asegúrese de que las herramientas en los cuadrantes de las esquinas tengan el radio de la punta establecido correctamente en la tabla de herramientas para que el punto de control sea correcto. Tool touch off automaticamente agrega un G43 por lo que la herramienta actual es el offset actual.

Una sesión típica podría ser:

1. Homing de cada eje si no lo están.
2. Configurar la herramienta actual con *T*n *M6 G43*, donde *n* es el número de la herramienta.
3. Seleccionar el eje X en la *ventana de Control Manual*.
4. Mover X a una posición conocida o realizar un corte de prueba y medir el diámetro.
5. Seleccionar Touch Off y seleccionar Tabla de herramientas, luego ingrese la posición o el diámetro.
6. Siga la misma secuencia para corregir el eje Z.

Nota: si está en el modo Radio, debe ingresar el radio, no el diámetro.

2.5.4.2. Toque inicial Z

Los offsets del eje Z pueden ser un poco confusos al principio porque hay dos elementos para el desplazamiento Z. Existe el desplazamiento de la tabla de herramientas y el desplazamiento de las coordenadas de la máquina. Primero veremos los offsets de la tabla de herramientas. Un método es usar un punto fijo en su torno y establecer el desplazamiento Z para todas las herramientas desde este punto. Algunos usan la nariz del husillo o la cara del mandril. Esto le da la posibilidad de cambiar a una nueva herramienta y establece su offset Z sin tener que reiniciar todas las herramientas.

Una sesión típica podría ser:

1. Homing de cada eje si no lo están.
2. Asegurar que no haya offsets en vigor para el sistema de coordenadas actual.
3. Configurar la herramienta actual con *T*n *M6 G43*, donde *n* es el número de la herramienta.
4. Seleccionar el eje Z en la ventana de Control manual.
5. Acerque la herramienta a la superficie de control.
6. Usando un cilindro mueva el Z lejos de la superficie de control hasta que el cilindro pase justo entre la herramienta y la superficie de control.
7. Seleccione Touch Off y seleccione la Tabla de herramientas y establezca la posición en 0.0.
8. Repita para cada herramienta usando el mismo cilindro.

Ahora todas las herramientas están desplazadas a la misma distancia de una posición estándar. Si cambia una herramienta como una broca, repita lo anterior y ahora estará sincronizado con el resto de las herramientas para el desplazamiento Z. Algunas herramientas pueden requerir un poco más para determinar el punto de control desde el punto touch off. Por ejemplo, si tiene una herramienta de tronzado de 0.125” de ancho y toca con el lado izquierdo, pero quiere que el derecho sea el Z0, ingrese 0.125” en la ventana touch off.

2.5.4.3. El offset Z de la máquina

Una vez que todas las herramientas tienen el offset Z ingresado en la tabla de herramientas, puede utilizar cualquier herramienta para configurar el offset de la máquina utilizando el sistema de coordenadas de máquina.

Una sesión típica podría ser:

1. Homing de cada eje si no lo están.
2. Configurar la herramienta actual con *T*n *M6*, donde *n* es el número de la herramienta.
3. Emitir un G43 de modo que el offset de la herramienta actual esté vigente.
4. Lleve la herramienta a la pieza de trabajo y configure el offset Z de la máquina.

Si olvida configurar el G43 para la herramienta actual al configurar el sistema de coordenadas de la máquina compensado, no obtendrá lo que espera, ya que el offset de la herramienta se agregará al offset actual cuando la herramienta se utilize en su programa.

2.5.5. Movimiento sincronizado del husillo

El movimiento sincronizado del husillo requiere un codificador de cuadratura conectado al husillo con un pulso índice por revolución. Ver la página man de motion y el Ejemplo de control de husillo para más información.

Roscado El ciclo de roscado G76 se utiliza para roscas internas y externas. Para obtener más información, consulte la sección G76.

Velocidad constante de superficie CSS (Constant Surface Speed) utiliza el origen de la máquina X modificado por el offset X de la herramienta para calcular la velocidad del husillo en RPM. CSS hará un seguimiento de los cambios en los offsets de herramienta. El origen máquina X debe ser donde la herramienta de referencia (la que tiene cero desplazamiento) está en el centro de rotación. Para obtener más información, consulte la sección G96.

Alimentación por revolución La alimentación por revolución moverá el eje Z en la cantidad de F por cada revolución. Esto no es para roscar; use G76 para roscado. Para obtener más información, consulte la sección G95.

2.5.6. Arcos

Calcular arcos puede ser desafiante si no se considera el modo de radio y diámetro en tornos así como la orientación del sistema de coordenadas de la máquina. Lo siguiente se aplica a los arcos de formato centro. En un torno, debe incluir G18 en su preámbulo ya que el valor predeterminado es G17, incluso si está en modo torno, en la interfaz de usuario AXIS. Los arcos en el plano XZ de G18 utilizan los desplazamientos I (eje X) y K (eje Z).

2.5.6.1. Diseño de arcos en torno

Un torno típico tiene el husillo a la izquierda del operador y la herramienta en el lado del operador desde línea central del husillo. Esto normalmente se configura con el eje Y imaginario (+) apuntando al suelo.

Lo siguiente será cierto en este tipo de configuración:

- El eje Z (+) apunta hacia la derecha, alejándose del husillo.
- El eje X (+) apunta hacia el operador, y cuando está en el lado del operador, los valores de X son positivos.

Algunos tornos con herramientas en la parte posterior tienen el eje Y imaginario (+) apuntando hacia arriba.

Las direcciones de arco G2/G3 se basan en el eje alrededor del cual giran. En el caso de los tornos, se trata del eje Y imaginario. Si el eje Y (+) señala hacia el piso, hay que mirar hacia arriba para que aparezca el arco que va en la dirección correcta. Así que mirando desde (no hacia) arriba, invierta G2/G3 para que el arco aparezca en la dirección correcta.

2.5.6.2. Modo de radio y diámetro

Al calcular arcos en modo radio solo tiene que recordar la dirección de rotación que se aplica a su torno.

Al calcular arcos en el modo de diámetro, X es el diámetro y el offset X (I) es el radio incluso si está en el modo de diámetro G7.

2.5.7. Ruta de la herramienta

2.5.7.1. Punto de control

El punto de control de la herramienta sigue la ruta programada. El punto de control está en la intersección de las líneas paralelas a X y Z, tangentes al diámetro de la punta de la herramienta, tal como define los ejes X y Z un touch off para esa herramienta. Al tornear o refrentar piezas con caras rectas, el camino de corte y el filo de la herramienta siguen el mismo recorrido. Al tornear radios y ángulos, el filo de la herramienta no seguirá la ruta programada a menos que esté activada la *compensacion del cortador*. En la siguientes figuras se puede ver cómo el punto de control no sigue el filo de la herramienta, como se podría suponer.

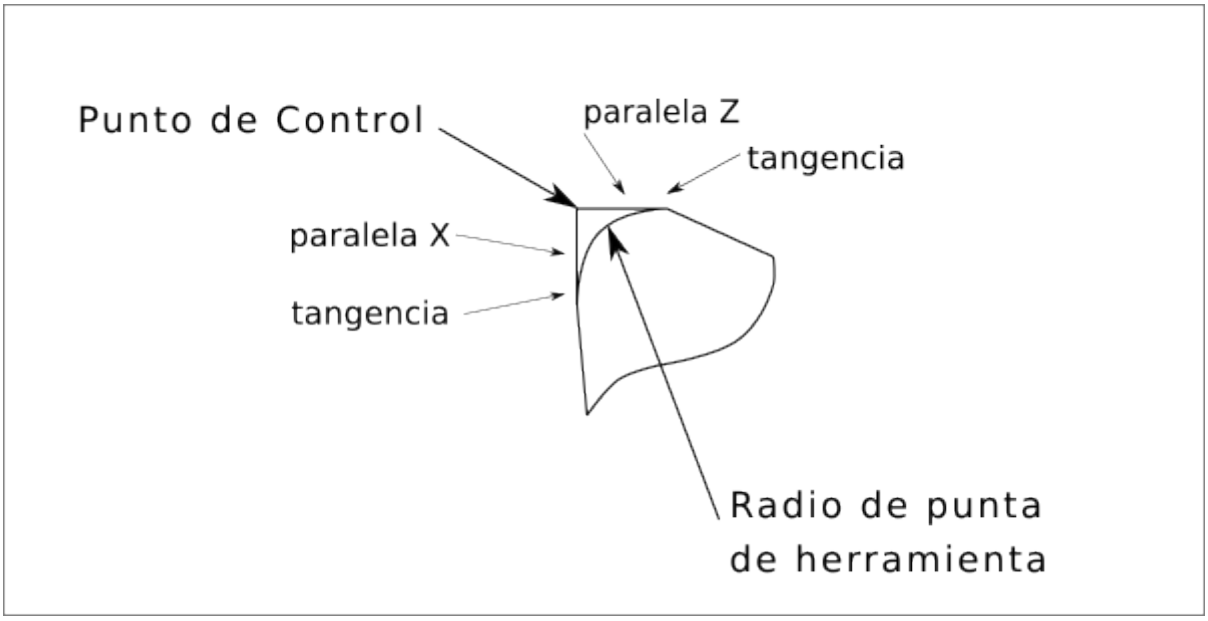


Figura 2.17: Punto de control

2.5.7.2. Corte de ángulos sin compensación

Ahora imagine que programamos una rampa sin compensación. El camino programado se muestra en la siguiente figura. Como se puede ver, la ruta programada y la ruta de corte deseada son una y la misma, siempre que nos estemos moviendo en la dirección X o Z solamente.

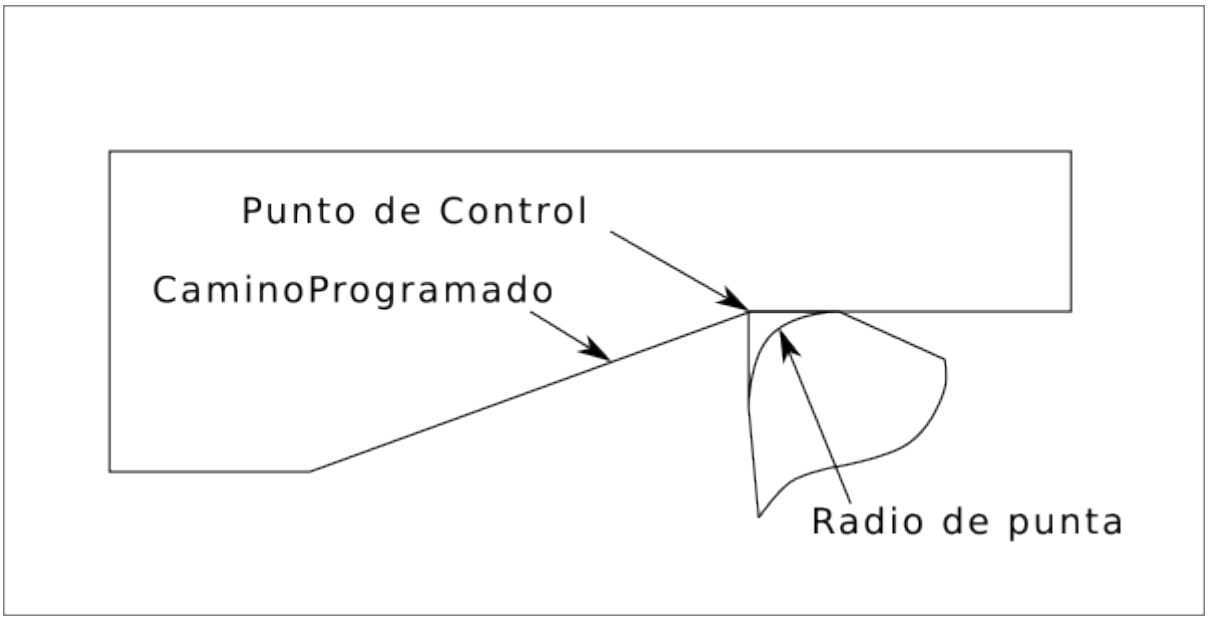


Figura 2.18: Entrada a la rampa

Ahora, a medida que el punto de control avanza a lo largo del camino programado, el borde del cortador real no sigue ese camino como se muestra en la siguiente figura. Hay dos maneras de resolver esto; compensacion del cortador o ajustar la ruta programada para compensar el radio de la punta.

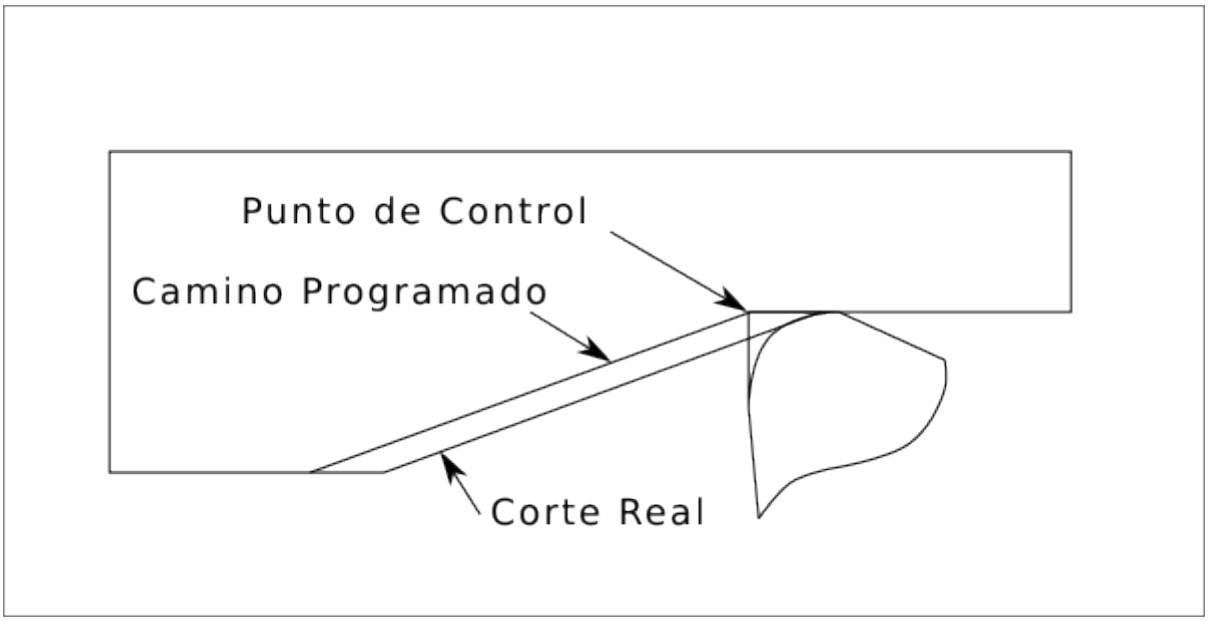


Figura 2.19: Ruta en rampa

En el ejemplo anterior, es un ejercicio simple ajustar la ruta programada a la ruta real deseada moviendo la ruta programada hacia la izquierda del radio de la punta de la herramienta.

2.5.7.3. Cortando un radio

En este ejemplo examinaremos lo que sucede durante el corte de un radio sin compensación. En la siguiente figura se ve la herramienta torneando el diámetro exterior de la pieza. El punto de control de la herramienta sigue el camino programado y la herramienta toca el diámetro exterior de la pieza.

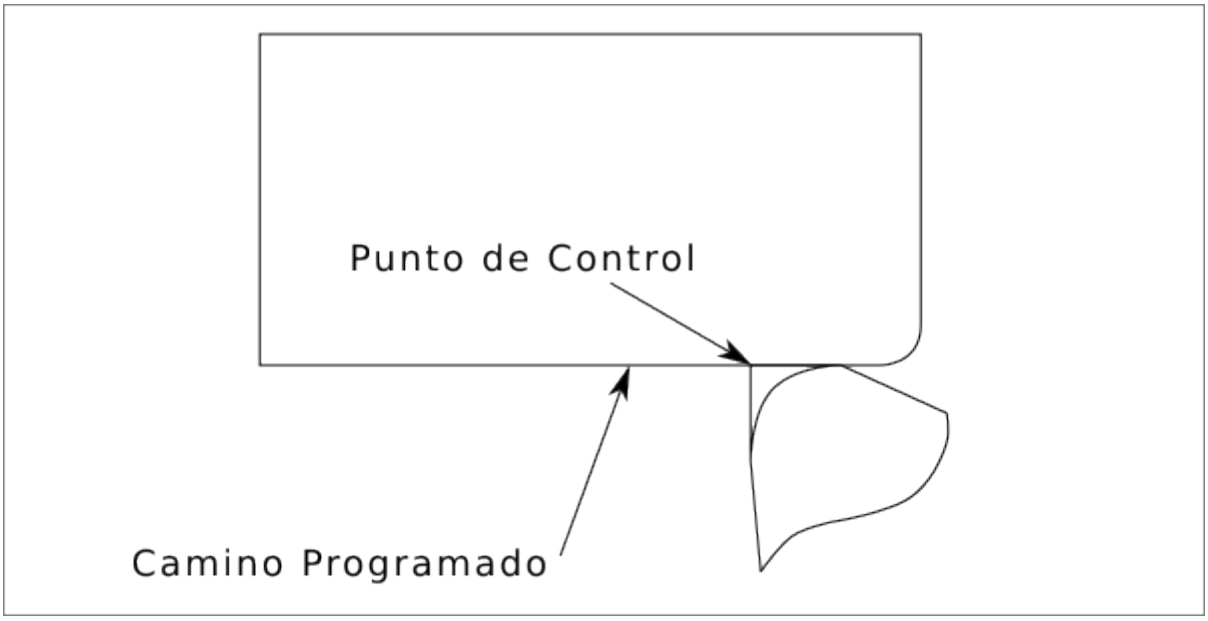


Figura 2.20: Torneado exterior

En la siguiente figura puede ver que a medida que la herramienta se acerca al final de la pieza, el punto de control todavía sigue el camino pero la punta de la herramienta ha dejado la pieza y está al aire. También puede ver que aunque ha sido programado un radio, la pieza terminará con una esquina cuadrada.

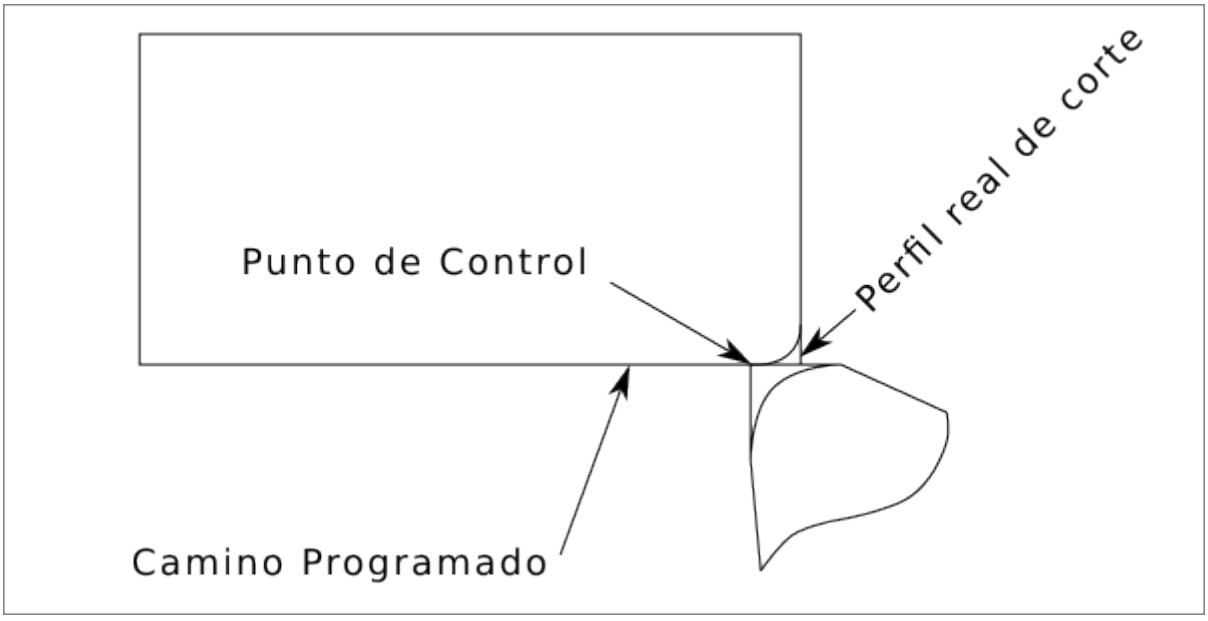


Figura 2.21: Torneado del radio

Ahora puede ver que mientras el punto de control sigue al radio programado, la punta de la herramienta ha dejado la pieza y ahora está al aire.

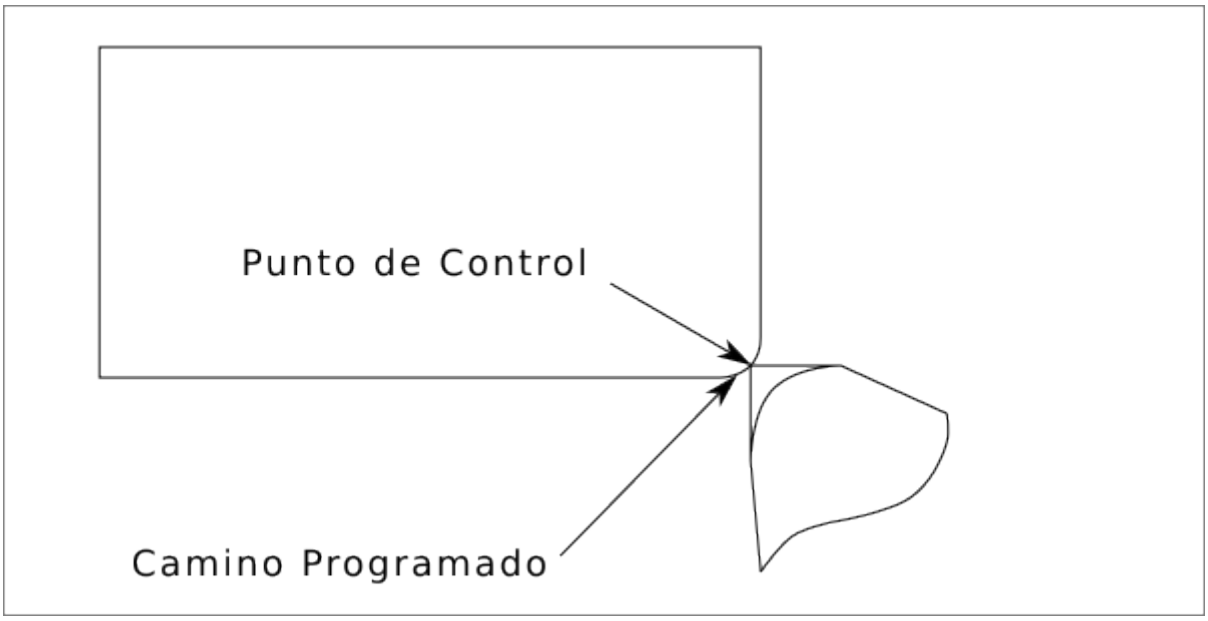


Figura 2.22: Torneado del radio

En la figura final podemos ver que la punta de la herramienta terminará de cortar la cara pero deja una esquina cuadrada en lugar de un radio. Nótese también que si se programa el corte para que finalice en el centro de la pieza, una pequeña cantidad de material no se cortará debido al radio de la herramienta. Para terminar una cara cortada en el centro de una parte, tiene que programar la herramienta para ir más allá del centro, como mínimo, el radio de la punta de la herramienta.

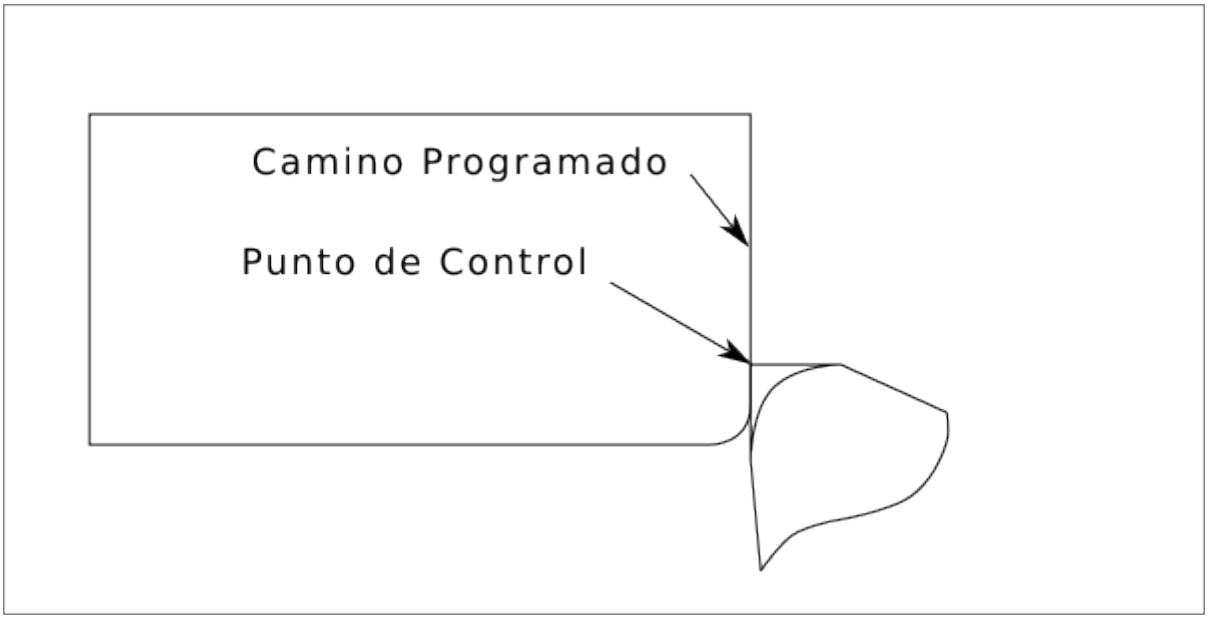


Figura 2.23: Corte de la cara

2.5.7.4. Usando compensación del cortador

- Cuando use compensación de cortador en un torno, piense en el radio de la punta de la herramienta como el radio de un cortador cilíndrico (fresa).
- Cuando se usa compensación de cortador, el camino debe ser lo suficientemente grande para que una herramienta cilíndrica no interfiera en la línea siguiente.
- Cuando se cortan líneas rectas en el torno es posible que no desee utilizar compensacion del cortador. Por ejemplo, mandrinar un agujero con una barra demasiado ajustada puede que no le permita suficiente espacio para realizar el movimiento de salida.
- El movimiento de entrada en un arco con compensación es importante para obtener los resultados correctos.

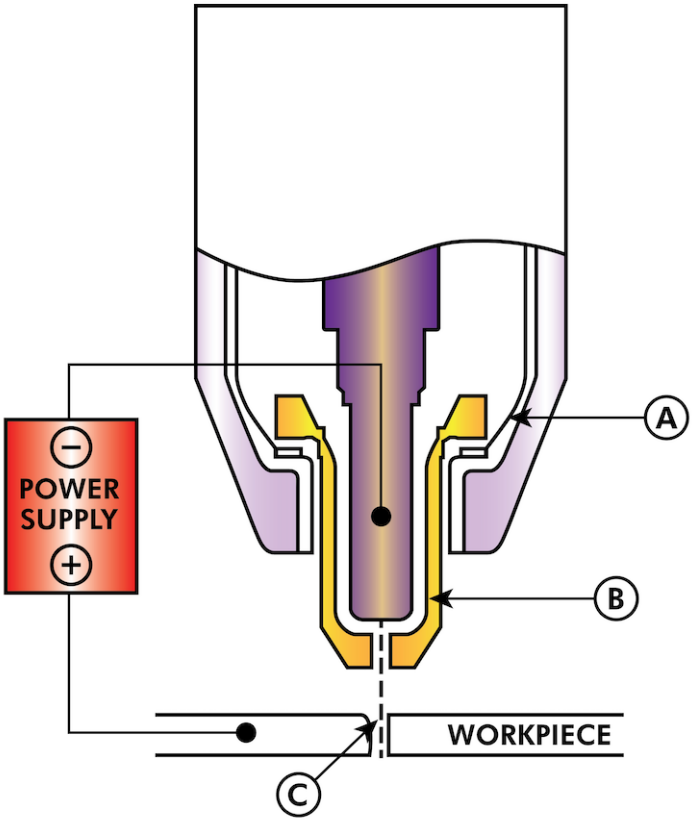
2.6. Básicos del corte por plasma para usuarios de LinuxCNC

2.6.1. ¿Qué es el plasma?

El plasma es un cuarto estado de la materia, un gas que se ha calentado a una temperatura extremadamente alta e ionizado para que se vuelva eléctricamente conductor. Los procesos de corte y acanalado por arco de plasma utilizan este plasma para transferir un arco eléctrico a la pieza de trabajo. El metal a cortar o quitar se derrite por el calor del arco y se sopla. Si bien el objetivo del corte con arco de plasma es la separación del material, el acanalado con arco de plasma se utiliza para eliminar metales a una profundidad y ancho controlados.

Las antorchas de plasma son similares en diseño a las bujías de automocion. Consisten en secciones negativas y positivas separadas por un aislante central. Dentro de la antorcha, el arco piloto comienza en el espacio entre el electrodo cargado negativamente y la punta cargada positivamente. Una vez que el arco piloto ha ionizado el gas, la columna de plasma sobrecalentada fluye a través del pequeño orificio en la punta de la antorcha, que se enfoca en el metal a cortar.

En una antorcha de corte por plasma, un gas frío ingresa a la Zona B, donde un arco piloto entre el electrodo y la punta de la antorcha calienta e ioniza el gas. El arco de corte principal se transfiere luego a la pieza de trabajo a través de la columna de gas de plasma en la Zona C. Al forzar el gas de plasma y el arco eléctrico a través de un pequeño orificio, la antorcha suministra una alta concentración de calor a un área pequeña. El arco de plasma rígido y restringido se muestra en la Zona C. Se usa polaridad directa de corriente continua (CC) para el corte por plasma, como se muestra en la ilustración. La zona A canaliza un gas secundario que enfría la antorcha. Este gas también ayuda al gas de plasma de alta velocidad a expulsar el metal fundido del corte permitiendo un corte rápido y sin escoria.



TYPICAL TORCH HEAD DETAIL

2.6.2. Inicialización de arco

Hay dos métodos principales para la inicialización del arco para cortadores de plasma que están diseñados para funcionar con CNC. Si bien se utilizan otros métodos en algunas máquinas (como el inicio por contacto, donde se requiere el contacto físico de la antorcha con el material), no son adecuados para aplicaciones CNC.

2.6.2.1. Inicio por alta frecuencia

Este tipo de inicio es ampliamente empleado. Aunque es una tecnología más antigua, funciona bien y se inicia rápidamente. Pero, debido a la alta frecuencia y al alto voltaje que se requiere para ionizar el aire, tiene algunos inconvenientes. A menudo interfiere con los circuitos electrónicos circundantes e incluso puede dañar los componentes. También se necesita un circuito especial para crear el arco piloto. Los modelos de bajo costo no tendrán un arco piloto, y requieren tocar el consumible para comenzar el trabajo. El empleo de un circuito de HF también puede aumentar los problemas de mantenimiento, ya que generalmente hay puntos ajustables que deben limpiarse y reajustarse de vez en cuando.

2.6.2.2. Blowback Start

Este tipo de arranque utiliza la presión de aire suministrada a la cortadora para forzar un pequeño pistón o cartucho dentro del cabezal de la antorcha hacia atrás para crear una pequeña chispa entre

la superficie interior del consumible, ionizando el aire y creando una pequeña llama de plasma. Esto también crea un ”arco piloto” que proporciona una llama de plasma que permanece encendida, tanto si está en contacto con el metal como si no. Este es un muy buen tipo de inicio que ahora utilizan varios fabricantes. Su ventaja es que requiere algo menos de circuiteria, es bastante confiable y genera mucho menos ruido eléctrico.

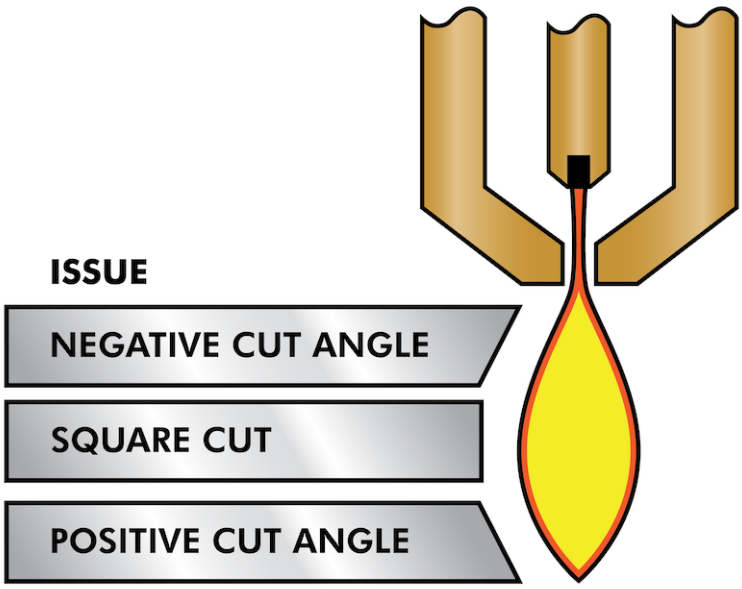
Para los sistemas CNC de plasma de aire de nivel básico, se prefiere el estilo blowback para minimizar la interferencia eléctrica con la electrónica y las PC estándar, pero el arranque de alta frecuencia sigue siendo el más importante en máquinas grandes de 200 amperios o más. Estos requieren PC’s y dispositivos electrónicos de nivel industrial e incluso los fabricantes comerciales han tenido problemas con fallos porque no han tenido en cuenta el ruido eléctrico en sus diseños.

2.6.3. Plasma y CNC

Las operaciones de plasma en máquinas CNC son bastante especiales en comparación con el fresado o el torneado y es un proceso un tanto huérfano. El calentamiento desigual del material por el arco de plasma hará que la placa se pandee y se doble. La mayoría de las placas de metal no salen de fábrica en un estado muy uniforme o plano. Las placas gruesas (más de 30 mm) pueden estar fuera de plano de 50 a 100 mm. La mayoría de las operaciones de código G en CNC comenzarán a partir de una referencia conocida o una pieza de stock que tiene un tamaño y forma conocidos, y el código G se escribe para desbastar el exceso y finalmente cortar la parte terminada. Con el plasma, el estado desconocido de la placa hace que sea imposible generar un código G que satisfaga estas variaciones en el material.

Un arco de plasma tiene forma ovalada y la altura de corte debe controlarse para minimizar bordes biselados. Si la antorcha está demasiado alta o demasiado baja, los bordes pueden quedar excesivamente biselados. También es crítico que la antorcha se mantenga perpendicular a la superficie.

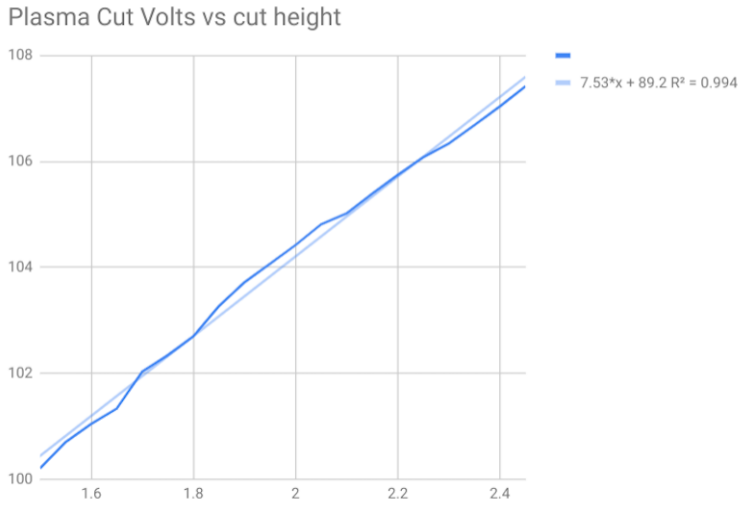
- **La distancia de trabajo de la antorcha puede afectar el bisel del borde**



- **Ángulo de corte negativo:** antorcha demasiado baja, aumente la distancia de trabajo de la antorcha.
- **Ángulo de corte positivo:** antorcha demasiado alta, disminuir la distancia de trabajo de la antorcha.

nota
Una ligera variación en los ángulos de corte puede ser normal, siempre que esté dentro de tolerancia.

La capacidad de controlar con precisión la altura de corte en un entorno tan hostil y siempre cambiante es un desafío muy difícil. Afortunadamente, existe una relación muy lineal entre la altura de la antorcha (longitud del arco) y el voltaje del arco, como muestra este gráfico.



Este gráfico se preparó a partir de una muestra de aproximadamente 16,000 lecturas a diferentes alturas de corte y el análisis de regresión muestra 7.53 V/mm con un 99.4 % de confianza. En este caso particular, esta muestra fue tomada de una máquina Everlast de 50 A controlada por LinuxCNC.

El voltaje de la antorcha se convierte en una variable de control de proceso ideal para usarse en el ajuste de la altura de corte. Supongamos, por simplicidad, que el voltaje cambia en 10 voltios por mm. Esto se puede restablecer a 1 voltio por 0.1 mm (0.04”). Los principales fabricantes de máquinas de plasma (por ejemplo, Hypertherm, Thermal Dynamics y ESAB) producen tablas de corte que especifican la altura de corte recomendada y el voltaje de arco estimado a esta altura, así como algunos datos adicionales. Entonces, si el voltaje del arco es 1 voltio más alto que la especificación del fabricante, el controlador simplemente necesita bajar la antorcha en 0.1 mm (0.04 ”) para regresar a la altura de corte deseada. Tradicionalmente se usa una unidad de control de altura de la antorcha (THC) para administrar este proceso.

2.6.4. Elegir una máquina de plasma para operaciones CNC

Hay una gran cantidad de máquinas de plasma disponibles en el mercado hoy en día y no todas son aptas para el uso de CNC. El corte por plasma CNC es una operación compleja y se recomienda que los integradores elijan una máquina de plasma adecuada. Si no se hace, es probable que provoque horas y horas infructuosas en solución de problemas tratando de remediar la falta de lo que muchos considerarían características obligatorias.

Si bien las reglas pueden romperse si se comprende completamente los motivos por los que se aplica la regla, consideramos que un constructor de mesas de plasma debería seleccionar una máquina con las siguientes características:

- Inicio blowback, para minimizar el ruido eléctrico y simplificar la construcción
- Se prefiere una antorcha para máquina, pero muchos han usado antorchas manuales.

- Una punta de antorcha totalmente protegida para permitir la detección óhmica

Si tiene presupuesto, una máquina de gama alta le proporcionará:

- Tablas de corte del fabricante, que ahorrarán muchas horas y desperdicio de material al calibrar los parámetros de corte
- Contactos secos para ArcOK
- Terminales para el interruptor Arc On
- Voltaje de arco sin procesar o salida de voltaje de arco dividido
- Opcionalmente, una interfaz RS485 si utiliza un cortador de plasma Hypertherm y desea controlarlo desde la consola LinuxCNC.
- Ciclos de trabajo altos

En los últimos tiempos, una clase de máquina que incluye algunas de estas características está disponible por alrededor de $550 USD. Un ejemplo es la Herocut55i disponible en Amazon, pero todavía no hay comentarios de los usuarios. Esta máquina cuenta con una antorcha blowback, salida ArcOK, contactos de inicio de antorcha y voltaje bruto de arco.

2.6.5. Tipos de control de altura de antorcha

La mayoría de las unidades de THC son dispositivos externos y muchas tienen un método de ajuste "bit bang" bastante burdo. Proporcionan dos señales al controlador LinuxCNC. Una se enciende si el eje Z debe moverse hacia arriba y la otra si el eje Z debe moverse hacia abajo. Ninguna señal es verdadera si la antorcha está a la altura correcta. El popular Proma 150 THC es un ejemplo de este tipo de THC. El componente LinuxCNC THCUD está diseñado para funcionar con este tipo de THC.

Con el lanzamiento de la interfaz de voltaje a frecuencia Mesa THCAD, LinuxCNC pudo decodificar el voltaje real de la antorcha a través de una entrada de encoder. Esto permitió que LinuxCNC controlara el eje Z y eliminara el hardware externo. Las primeras implementaciones que utilizan THCAD replicaron el enfoque "bit bang". El componente LinuxCNC THC es un ejemplo de este enfoque.

Jim Colt, de Hypertherm, dice oficialmente que los mejores controladores de THC estaban completamente integrados en el controlador CNC. Por supuesto, se refería a los sistemas de gama alta fabricados por Hypertherm, Esab, Thermal Dynamics y otros, como Advanced Robotic Technology en Australia, sin soñar que el código abierto podría producir sistemas utilizando este enfoque que rivaliza con los sistemas de gama alta.

La inclusión de offsets externos en LinuxCNC V2.8 permitió que el control de plasma en LinuxCNC se elevara a un nivel completamente nuevo. Los offsets externos se refieren a la capacidad de aplicar un offset a la posición ordenada del eje, externa al controlador de movimiento. Esto es perfecto para el control de THC por plasma como un método para ajustar la altura de la antorcha en tiempo real según nuestra metodología de control de proceso elegida. Después de una serie de compilaciones experimentales, la configuración Plasmac se incorporó a LinuxCNC 2.8, y fue reemplazada por QtPlasmaC en LinuxCNC 2.9. Este ha sido un proyecto extremadamente ambicioso y muchas personas en todo el mundo han participado en las pruebas y la mejora del conjunto de características. QtPlasmaC es único en el sentido de que su objetivo de diseño era admitir todos los THC, incluidos los de bit bang, hasta el sofisticado control de voltaje de la antorcha, si el voltaje está disponible para LinuxCNC a través de un THCAD o algún otro sensor de voltaje. Además, QtPlasmaC está diseñado para ser un sistema independiente que no necesita ninguna subrutina adicional de código G y permite al usuario definir sus propias tablas de corte que están almacenadas en el sistema y accesibles mediante un menú desplegable.

2.6.6. Señal Arc OK

Las máquinas de plasma que tienen una interfaz CNC contienen un conjunto de contactos (por ejemplo, un relé) que se cierran cuando se establece un arco válido y cada lado de estos contactos serán pines en la interfaz CNC. Un integrador de mesas de plasma debe conectar un lado de estos pines a la alimentación de campo y el otro a un pin de entrada. Esto permite que el controlador CNC sepa cuándo se establece un arco válido y también cuándo se pierde un arco inesperadamente. Aquí hay una trampa potencial cuando la entrada es un circuito de alta impedancia, como una tarjeta Mesa. Si los contactos son un relé simple, existe una alta probabilidad de que la corriente que pasa a través del relé sea menor que la especificación de corriente mínima. En estas condiciones, los contactos del relé pueden sufrir una acumulación de óxido que con el tiempo puede dar como resultado una operación de contacto intermitente. Para evitar que esto suceda, se debe instalar una resistencia pull-down en el pin de entrada del controlador. Se debe tener cuidado para garantizar que esta resistencia se seleccione para asegurar que la corriente mínima pase a través del relé y tenga la potencia suficiente para manejar la potencia en el circuito. Finalmente, la resistencia debe montarse de tal manera que el calor generado no dañe nada mientras esté en funcionamiento.

Si tiene una señal ArcOK, se recomienda que se use por encima de cualquier señal sintetizada para eliminar posibles problemas de construccion. Una señal sintetizada disponible desde un THC externo o el Modo 0 de QtPlasmaC no puede reemplazar completamente los circuitos ArcOK en un inverter de plasma. Se han observado algunos problemas de construcción en los que se ha producido una mala configuración o incompatibilidad con el inverter de plasma a partir de una señal sintetizada de ArcOK. Sin embargo, en general, una señal ArcOK sintetizada correctamente configurada está bien.

Se puede lograr una señal de arco de arco simple y efectiva con un simple relé reed. Envuelva 3 vueltas de uno de los cables gruesos de la cortadora de plasma (por ejemplo, el cable de sujeción del material) a su alrededor. Coloque el relé en un tubo para protección y conecte un lado del relé a la alimentación de campo y el otro extremo a su pin de entrada ArcOK.

2.6.7. Detección de altura inicial

Debido a que la altura de corte es un parámetro tan crítico del sistema y la superficie del material es inherentemente desigual, un mecanismo del eje Z necesita un método para detectar la superficie del material. Hay tres métodos que pueden lograr esto:

1. Detección de corriente para detectar torque de motor incrementado,
2. un interruptor ”flotante” y uno eléctrico o
3. un circuito de detección ”óhmico” que se cierra cuando el protector de la antorcha hace contacto con el material.

La detección de corriente no es una técnica viable para mesas caseras, pero los interruptores flotantes y la detección óhmica se discuten a continuación:

2.6.7.1. Interruptores flotantes

La antorcha se monta en una plataforma deslizante que puede moverse hacia arriba cuando la punta de la antorcha hace contacto con la superficie del material y activa un interruptor o sensor. A menudo, esto se logra bajo el control de código G utilizando los comandos G38. Si este es el caso, luego de la prueba inicial, se recomienda sondear lejos de la superficie hasta que la señal de la sonda se pierda a una velocidad más lenta. Además, asegúrese de que se tenga en cuenta la histéresis del interruptor.

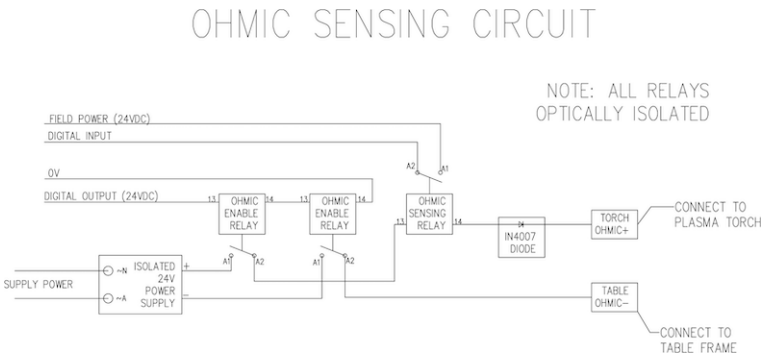
Independientemente del método de sondeo utilizado, se recomienda encarecidamente que se implemente el interruptor flotante para que haya una señal fallback o secundaria para evitar daños a la antorcha por un choque.

2.6.7.2. Detección óhmica

La detección óhmica se basa en el contacto entre la antorcha y el material que actúa como un interruptor para activar una señal eléctrica que es detectada por el controlador CNC. Siempre que el material esté limpio, este puede ser un método mucho más preciso para detectar el material que un interruptor flotante que puede causar la desviación de la superficie del material. Este circuito de detección óhmico opera en un entorno extremadamente hostil, por lo que es necesario implementar una serie de medidas de seguridad para garantizar la seguridad tanto de la electrónica del CNC como del operador. En el corte por plasma, la abrazadera de tierra unida al material es positiva y la antorcha es negativa. Se recomienda que:

1. La detección óhmica solo se implementará cuando la antorcha tenga una protección aislada de la punta de la antorcha que transporta el arco de corte.
2. El circuito óhmico utiliza una fuente de alimentación aislada totalmente separada que activa un relé optoaislado para permitir que la señal de prueba se transmita al controlador CNC.
3. El lado positivo del circuito debe estar en la antorcha
4. Ambos lados del circuito deben aislarse mediante relés optoaislados hasta que se realice el sondeo
5. Se deben usar diodos de bloqueo para evitar que el voltaje de arco ingrese al circuito de detección óhmico.

El siguiente es un circuito de ejemplo que se ha demostrado que funciona y es compatible con la configuración de QtPlasmaC en LinuxCNC .



2.6.7.3. Hipersensibilidad con MESA THCAD-5

Un método más sofisticado de detección de material que elimina los relés y diodos es usar otro THCAD-5 para monitorear el voltaje del circuito de detección de material desde una fuente de alimentación aislada. La ventaja que tiene es que el THCAD está diseñado para el hostil entorno eléctrico de plasma y aísla totalmente y con seguridad el lado lógico del lado de alto voltaje.

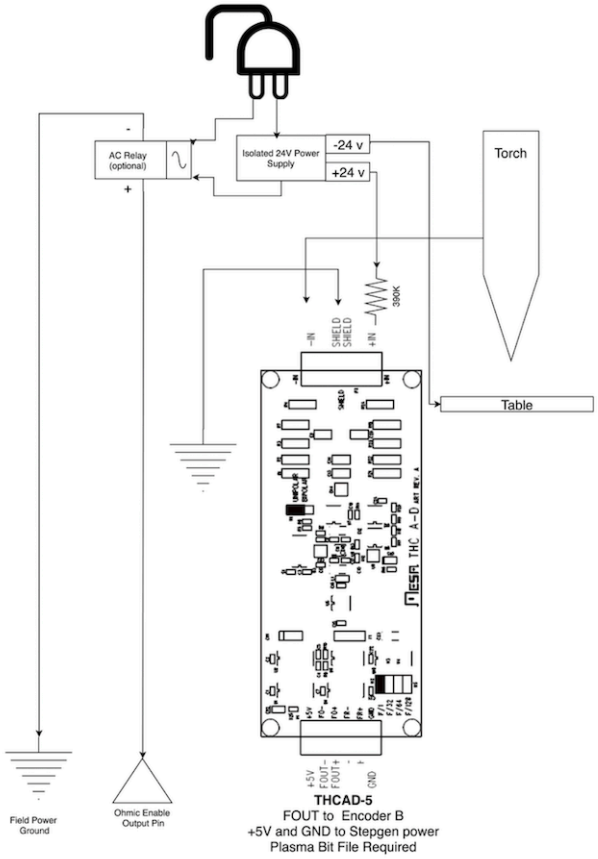
Para implementar este método, se requiere una segunda entrada de encoder.

Si usa una tarjeta Mesa, hay disponible un firmware diferente para proporcionar 2 entradas adicionales del codificador A en los pines del codificador B y del codificador. Este firmware está disponible para descargar para las placas 7i76e y 7i96 desde el sitio web de Mesa en las páginas del producto.

El THCAD es lo suficientemente sensible como para ver el aumento en el voltaje del circuito a medida que aumenta la presión de contacto. El componente ohmic.comp incluido en LinuxCNC puede monitorear el voltaje de detección y establecer un umbral de voltaje por encima del cual se considera que se hace contacto y se habilita una salida. Al monitorear el voltaje, se puede establecer un umbral inferior de ”circuito de interrupción” para construir una fuerte histéresis de interruptor. Esto minimiza

los disparos en falso. En nuestras pruebas, descubrimos que la detección de material mediante este método era más sensible y robusta, además que la implementación del cableado es más simple. Una ventaja adicional es que el uso de salidas de software, en lugar de pines físicos de E/S, libera pines para usarse con otros fines. Esta ventaja es útil para aprovechar al máximo la Mesa 7i96, que tiene pines de E/S limitados.

El siguiente diagrama de circuito muestra cómo implementar un circuito de hipersensibilidad.



Utilizamos una fuente de alimentación aislada de riel DIN Mean Well HDR-15 ultradelgada de 15 vatios basada en riel DIN de 24 V . Este es un dispositivo de doble aislamiento de clase II que resistirá cualquier voltaje de arco que pueda aplicarse a los terminales.

2.6.7.4. Ejemplo de código HAL para hipersensibilidad

El siguiente código HAL se puede pegar en su custom.hal de QtPlasmaC para habilitar la detección óhmica en el codificador 2 de un 7i76e. Instale el archivo de bits correcto y conecte el THCAD a IDX + e IDX-. Asegúrese de cambiar la configuración de calibración para estar de acuerdo con su THCAD-5.

```
# --- Cargue el componente ---
loadrt ohmic names=ohmicsense
addf ohmicsense servo-thread
```

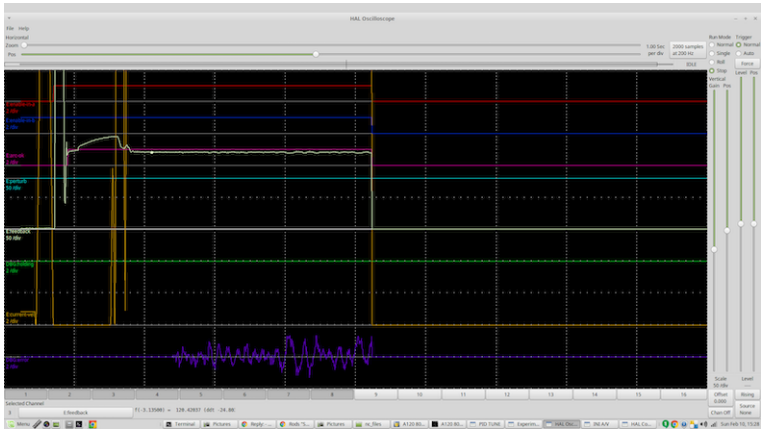
```
# --- 7i76e CONFIGURACIÓN DEL CODIFICADOR 2 PARA SENSADO OHMICO ---
setp hm2_7i76e.0.encoder.02.scale -1
setp hm2_7i76e.0.encoder.02.counter-mode 1

# --- Configure el componente ---
setp ohmicsense.thcad-0-volt-freq    140200
setp ohmicsense.thcad-max-volt-freq  988300
setp ohmicsense.thcad-divide         32
setp ohmicsense.thcad-fullscale      5
setp ohmicsense.volt-divider         4.9
setp ohmicsense.ohmic-threshold      22.0
setp ohmicsense.ohmic-low            1.0
net ohmic-vel ohmicsense.velocity-in <= hm2_7i76e.0.encoder.02.velocity

# --- Reemplace la señal de detección óhmica de QtPlasmaC ---
unlinkp db_ohmic.in
net ohmic-true ohmicsense.ohmic-on => db_ohmic.in
net plasmac:ohmic-enable    =>  ohmicsense.is-probing
```

2.6.8. Retraso THC

Cuando se establece un arco, el voltaje del arco alcanza un pico significativo y luego se estabiliza a un voltaje estable a la altura de corte. Tal como se muestra con la línea verde en la imagen a continuación.



Es importante que el controlador de plasma ”espere” antes de muestrear automáticamente el voltaje de la antorcha y comenzar el control de THC. Si se habilita demasiado pronto, el voltaje estará por encima de los voltios de corte deseados y la antorcha se bajará en un intento de abordar una condición de sobre-altura percibida.

En nuestras pruebas, esto varía entre máquinas y material de 0.5 a 1.5 segundos. Por lo tanto, una demora de 1.5 segundos después de que se recibe una señal válida de ArcOK antes de habilitar el control de THC es una configuración inicial segura. Si desea acortar esto para un material determinado, el Halscope de LinuxCNC le permitirá trazar el voltaje de la antorcha y tomar decisiones informadas sobre el menor retraso seguro a utilizar.

---

nota
Si la velocidad de corte no está cerca de la velocidad de corte deseada al final de este retraso, el controlador debe esperar hasta que esto se logre antes de habilitar el THC.

2.6.9. Muestreo de voltaje de antorcha

En lugar de confiar en las tablas de corte del fabricante para establecer el voltaje de la antorcha deseado, muchas personas (incluido el escritor) prefieren probar el voltaje a medida que el THC está habilitado y usarlo como un punto de ajuste.

2.6.10. Ruptura de antorcha

Se recomienda que se proporcione un mecanismo para permitir que la antorcha se ”rompa” o se caiga en caso de impacto con el material o una parte cortada que se haya volcado. Debe instalarse un sensor para permitir que el controlador CNC detecte si esto ha ocurrido y pause el programa en ejecución. Por lo general, esto se implementa usando imanes para asegurar la antorcha a la etapa del eje Z.

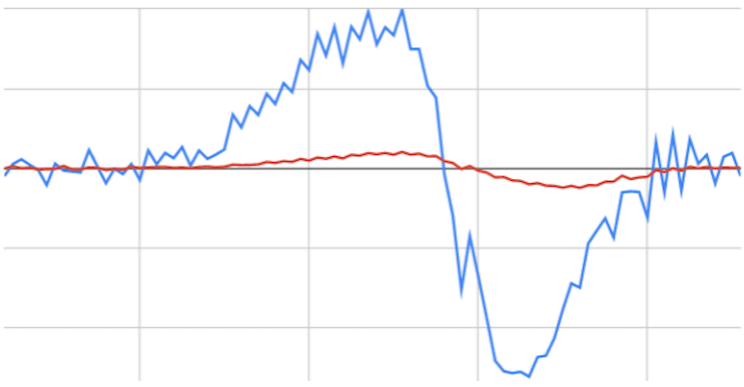
2.6.11. Bloquear esquinas / Velocidad anti-inmersión

El planificador de trayectoria LinuxCNC es responsable de traducir los comandos de velocidad y aceleración en movimiento que obedece las leyes de la física. Por ejemplo, el movimiento se ralentizará al negociar una esquina. Si bien esto no es un problema con las fresadoras o enrutadores, esto plantea un problema particular para el corte por plasma ya que el voltaje del arco aumenta a medida que el movimiento disminuye. Esto hará que el THC baje la antorcha. Una de las enormes ventajas de un control THC integrado en el controlador de movimiento LinuxCNC es que sabe lo que está sucediendo en todo momento. Por lo tanto, se convierte en una cuestión trivial monitorear la velocidad actual (`motion.current-speed`) y mantener la operación de THC si cae por debajo de un umbral establecido (por ejemplo, 10 % por debajo de la velocidad de avance deseada).

2.6.12. Cruce de vacío / Kerf

Si la antorcha de plasma pasa sobre un vacío durante el corte, el voltaje del arco aumenta rápidamente y el THC responde con un movimiento violento hacia abajo que puede aplastar la antorcha en el material y dañarla. Esta es una situación que es difícil de detectar y manejar. Hasta cierto punto, puede mitigarse con buenas técnicas de anidación, pero aún puede ocurrir en material más grueso cuando un retal cae. Este es el único problema que aún no se ha resuelto dentro del movimiento de código abierto LinuxCNC.

Una técnica sugerida es monitorear la tasa de cambio en los voltios de la antorcha a lo largo del tiempo (dv/dt) porque este parámetro es de orden de magnitud mayor al cruzar un vacío que lo que ocurre debido a la deformación normal del material. El siguiente gráfico muestra una gráfica de baja resolución de dv/dt (en azul) mientras cruza un vacío. La curva roja es un promedio móvil de voltios de antorcha.



Por lo tanto, debería ser posible comparar el promedio móvil con dv/dt y detener la operación del THC una vez que dv/dt exceda el rango normal esperado debido a la deformación. Se necesita más trabajo en esta área para llegar a una solución funcional en LinuxCNC.

2.6.13. Agujero y corte de forma pequeña

Se recomienda reducir la velocidad de corte al cortar agujeros y formas pequeñas.

John Moore dice: “Si desea detalles sobre el corte preciso de agujeros pequeños, consulte las hojas de ventas de Hypertherm *True Hole Technology* y también en PlasmaSpider, el usuario seanp ha publicado extensamente sobre su trabajo utilizando plasma de aire simple.

El método generalmente aceptado para obtener buenos agujeros de 37 mm de diámetro y hasta el grosor del material con un mínimo cono usando un plasma de aire es:

1. Utilice la corriente de corte recomendada para los consumibles.
2. Utilice la altura de corte recomendada fija (sin THC) para los consumibles.
3. Corte del 60 % al 70 % de la tasa de alimentación recomendada de consumibles y materiales.
4. Comience en o cerca del centro del agujero.
5. Use entrada perpendicular.
6. Sin salida, ya sea una ligera quemadura o un apagado temprano de antorcha dependiendo de lo que funcione mejor para usted.

Tendrá que experimentar para obtener el tamaño exacto del agujero porque el corte con este método será mas ancho que su corte recto habitual .”

Esta ralentización se puede lograr manipulando la velocidad de alimentación directamente en su post-procesador o utilizando alimentación adaptativa y un pin analógico como entrada. Esto le permite usar M67/M68 para establecer el porcentaje de alimentación deseada para cortar.

- Conociendo el avance

De la discusión anterior es evidente que el controlador de plasma necesita conocer la velocidad de alimentación establecida por el usuario. Esto plantea un problema con LinuxCNC porque LinuxCNC no guarda la velocidad de avance después de que el código G se almacena y analiza. Hay dos enfoques para solucionar esto:

1. Vuelva a asignar el comando F y guarde la velocidad de avance ordenada establecida en G-Code a través de un comando M67/M68.
2. Almacenar las tablas de corte en el controlador de plasma y permitir que el programa G-Code consulte la velocidad de avance actual (como lo hace QtPlasmaC).

Las etiquetas de estado es una característica recién añadida a LinuxCNC 2.9 que es útil para el corte por plasma. Esto agrega una ”etiqueta” que está disponible para el movimiento que contiene las velocidades de avance y velocidad actuales para todos los comandos de movimiento activos.

2.6.14. Pines de E/S para controladores de plasma

Los cortadores de plasma requieren varios pines adicionales. En LinuxCNC, no hay reglas estrictas sobre qué hace cada pin. En esta discusión asumiremos que el inversor de plasma tiene una interfaz CNC y que la tarjeta controladora tiene entradas activas altas en uso (por ejemplo, Mesa 7i76e).

Las mesas de plasma pueden ser máquinas grandes y le recomendamos que se tome el tiempo de instalar interruptores de límite máximo / mínimo e interruptores home separados para cada articulación. La excepción podría ser el límite inferior del eje Z. Cuando se activa un interruptor home, la articulación se desacelera bastante lentamente para obtener la máxima precisión. Esto significa que si desea utilizar velocidades de homing que sean proporcionales al tamaño de la mesa, puede sobrepasar el punto de activación inicial en 50-100 mm. Si utiliza un interruptor de límite/home compartido, debe mover el sensor fuera del punto de activación con HOME_OFFSET final o activará una falla del interruptor de límite cuando la máquina salga de la referencia. Esto significa que podría perder 50 mm o más de recorrido del eje con interruptores home/límite compartidos. Esto no sucede si se usan interruptores home y límite separados.

Por lo general, se requieren los siguientes pines (tenga en cuenta que las conexiones sugeridas pueden no ser apropiadas para una configuración de QtPlasmaC):

2.6.14.1. Arco OK (entrada)

- El inverter cierra los contactos cuando se establece un arco válido
- Conecte la alimentación de campo a un terminal ArcOK del inverter.
- Conecte otro terminal inverter OK al pin de entrada.
- Usualmente conectado a uno de los pines``motion.digital-``<*nn*> para usar desde código G con M66

2.6.14.2. Antorcha ON (salida)

- Activa un relé para cerrar la antorcha en el interruptor en el inverter.
- Conecte la antorcha en los terminales del inverter a los terminales de salida del relé.
- Conecte un lado de la bobina del relé al pin de salida.
- Conecte el otro lado de la bobina del relé a tierra de la alimentación de campo.
- Si se utiliza un relé mecánico, conecte un diodo flyback (por ejemplo, uno de la serie IN400x) a través de los terminales de la bobina con la banda del diodo apuntando hacia el pin de salida.
- Si se utiliza un relé de estado sólido, puede ser necesario observar la polaridad en las salidas.
- En algunas circunstancias, se puede usar el relé del husillo integrado en las tarjetas Mesa en lugar de un relé externo.
- Generalmente conectado a `spindle.0.on`.



aviso
Se recomienda encarecidamente que la antorcha no se pueda habilitar mientras este pin sea falso; de lo contrario, la antorcha no se apagará cuando se presione Estop.

2.6.14.3. Interruptor flotante (entrada)

- Utilizado para sondeo de superficie. Un sensor o interruptor que se activa si la antorcha se desliza hacia arriba cuando toca el material.
- Conecte la salida del sensor de proximidad al pin de entrada elegido. Si se utilizan interruptores mecánicos, conecte un lado del interruptor a la alimentación de campo y el otro lado a la entrada.
- Generalmente conectado a `motion.probe-input`.

2.6.14.4. Activación del sensor óhmico (salida)

- Vea el esquema de detección óhmica.
- Conecte el pin de salida a un lado de los relés de aislamiento y el otro lado a tierra de la alimentación de campo.
- En una configuración que no sea QtPlasmaC, generalmente activado por un ``motion.digital-out-``*<nn>* para que `M62` / `M63` / `M64` / `M65` pueda controlarlo en G-Code.

2.6.14.5. Detección óhmica (entrada)

- Tenga cuidado de seguir el esquema de detección óhmica mostrado anteriormente.
- Una fuente de alimentación aislada activa un relé cuando el protector de la antorcha hace contacto con el material.
- Conecte la alimentación de campo a un terminal de salida y el otro a la entrada.
- Tenga cuidado de observar la polaridad del relé si se utilizan relés de estado sólido optoacoplados.
- Por lo general está conectado a `motion.probe-input` y se puede hacer OR con el interruptor flotante.

Como se puede ver, las mesas de plasma son intensivas en pines y ya hemos consumido alrededor de 15 entradas antes de que se agreguen los estop normales. Otros tienen otros puntos de vista, pero es la opinión del escritor que la tarjeta Mesa 7i76e es preferible sobre la más barata 7i96 para permitir el MPG, interruptor de selección de escala y eje y otras características que puede agregar con el tiempo. Si su mesa usa servos, hay varias alternativas. Si bien hay otros proveedores, diseñar su máquina alrededor del ecosistema de Mesa simplificará el uso de su placa THCAD para leer el voltaje del arco.

2.6.14.6. Sensor de ruptura de la antorcha

- Como se mencionó anteriormente, se debe instalar un sensor de ruptura que se dispara si la antorcha se estrella y se cae.
- Por lo general, esto se conectaría a `halui.program-pause` para que la falla se pueda rectificar y se reanude el programa.

2.6.15. Código G para controladores de plasma

La mayoría de los controladores de plasma ofrecen un método para cambiar la configuración de código G. LinuxCNC admite esto a través de M67/M68 para comandos analógicos y M62-M65 para digital (comandos de encendido/apagado). Cómo se implemente esto es totalmente arbitrario. Veamos cómo la configuración de LinuxCNC QtPlasmaC hace esto:

Seleccione la configuración del material en QtPlasmaC y use la velocidad de alimentación para ese material.

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

nota
Los usuarios con una gran cantidad de entradas en la tabla de materiales de QtPlasmaC pueden necesitar aumentar el parámetro Q (p. ej. desde Q1 hasta Q2).

2.6.15.1. Habilitar/deshabilitar la operación del THC:

```
M62 P2 desactivará el THC (sincronizado con movimiento)
M63 P2 habilitará el THC (sincronizado con movimiento)
M64 P2 desactivará el THC (inmediatamente)
M65 P2 habilitará el THC (inmediatamente)
```

Reducir las velocidades de corte: (por ejemplo, para cortar agujeros)

```
M67 E3 Q0 establecería la velocidad al 100% de la velocidad solicitada
M67 E3 Q40 establecería la velocidad al 40% de la velocidad solicitada
M67 E3 Q60 establecería la velocidad al 60% de la velocidad solicitada
M67 E3 Q100 establecería la velocidad al 100% de la velocidad solicitada
```

Compensación del cortador:

```
G41.1 D#<_hal[plasmac_run.kerf-width-f]>; para la izquierda de la ruta programada
G42.1 D#<_hal[plasmac_run.kerf-width-f]> para la derecha de la ruta programada
G40 para desactivar la compensación
```

nota
Los integradores deben familiarizarse con la documentación de LinuxCNC para los diversos comandos propios de código G mencionados anteriormente.

2.6.16. Offsets externos y corte por plasma

Los offsets externos se introdujeron en LinuxCNC con la versión 2.8. Por externo, significa que podemos aplicar un desplazamiento externo al código G del que el planificador de trayectorias no sabe nada. Es más fácil de explicar con un ejemplo. Imagine un torno con un desplazamiento externo aplicado por una fórmula matemática para mecanizar el lóbulo en una leva. El torno gira a ciegas con el diámetro de corte establecido en un diámetro fijo y el desplazamiento externo mueve la herramienta hacia adentro y hacia afuera para mecanizar el lóbulo de la leva a través de un desplazamiento externo aplicado. Para configurar nuestro torno para mecanizar esta leva, necesitamos asignar una parte de

la velocidad y aceleración del eje a offsets externos o la herramienta no puede moverse. Aquí es donde entra en juego la variable ini OFFSET_AV_RATIO. Digamos que decidimos que necesitamos asignar el 20 % de la velocidad y la aceleración al offset externo al eje Z. Establecemos esto igual a 0.2. La consecuencia de esto es que su velocidad y aceleración máximas para el eje Z del torno es solo el 80 % de lo que podría ser.

Los offsets externos son un método muy poderoso para hacer ajustes de altura de la antorcha al eje Z a través de un THC. Pero el plasma tiene que ver con las altas velocidades y la aceleración rápida, por lo que no tiene sentido limitar estos parámetros. Afortunadamente en una máquina de plasma, el eje Z está 100 % controlado por el THC o no lo está. Durante el desarrollo de los offsets externos de LinuxCNC se reconoció que el movimiento del eje Z por código G y por THC eran mutuamente excluyentes. Esto nos permite trucar los offsets externos para que den el 100 % de la velocidad y la aceleración todo el tiempo. Podemos hacer esto duplicando los ajustes de velocidad y aceleración del eje Z de la máquina en el archivo INI y establecer OFFSET_AV_RATIO = 0.5. De esa manera, el 100 % de la velocidad y aceleración máximas estarán disponibles tanto para sondeo como para THC.

Ejemplo: En una máquina métrica con un motor NEMA23 con un accionamiento directo a un husillo de bolas de 5 mm, se determinó que una velocidad máxima de 60 mm/s y una aceleración de 700 mm/s^2 eran valores seguros sin pérdida de pasos. Para esta máquina, configure el eje Z en el archivo INI de la siguiente manera:

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

La articulación asociada con este eje tendría las variables de velocidad y aceleración establecidas de la siguiente manera:

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

Para obtener más información sobre las compensaciones externas (para V 2.8 y superior), lea la sección [AXIS_<letra>] del documento del archivo INI y Offsets de eje externos en la documentación de LinuxCNC.

2.6.17. Lectura de voltaje de arco con el THCAD de Mesa

La tarjeta Mesa THCAD es un notable convertidor de voltaje a frecuencia, preciso y con buen precio, que está diseñado para el entorno eléctrico ruidoso y hostil asociado con el corte por plasma. Internamente tiene un rango de 0-10 voltios. Este rango puede ampliarse simplemente agregando algunas resistencias como se describe en la documentación. Esta placa está disponible en tres versiones, el más nuevo THCAD-5, con un rango de 0-5 voltios, el THCAD-10 con un rango de 0-10 V y el THCAD-300 que está precalibrado para un rango extendido de 300 V. Cada placa se calibra individualmente y se le pega una etiqueta adhesiva indicando la frecuencia a 0 V y a escala completa. Para usarse con LinuxCNC, se recomienda seleccionar el divisor 1/32 en el enlace apropiado en la placa. En este caso, asegúrese también de dividir también las frecuencias establecidas por 32. Esto es más apropiado para el hilo servo de 1 kHz y también le da más tiempo al THCAD para promediar y suavizar la salida.

Hay mucha confusión acerca de cómo decodificar la salida THCAD, así que consideremos la Mesa 7i76e y el THCAD-10 por un momento con los siguientes datos de calibración hipotéticos:

- Escala completa 928 kHz (928 kHz/32 = 29 kHz)
- 0 V 121.6 kHz (121.6 kHz/32 = 3.8 kHz)

Debido a que la escala completa es de 10 voltios, entonces la frecuencia por voltio es:

(29000 Hz - 3800 Hz) / 10 V = 2520 Hz por Voltio

Asumiendo que tenemos una entrada de 5 voltios, la frecuencia calculada sería:

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

Ahora debería quedar bastante claro cómo convertir la frecuencia a su voltaje equivalente:

Voltaje = (frecuencia [Hz] - 3800 Hz) / (2520 Hz/V)

2.6.17.1. Conexiones THCAD

En el lado de alto voltaje:

- Conecte el voltaje de arco dividido o bruto a I_N+ e I_N-
- Conecte el blindaje del cable de interconexión a la conexión de blindaje.
- Conecte el otro terminal del blindaje a tierra.

Suponiendo que está conectado a una 7i76e de Mesa, conecte la salida a la entrada del codificador de husillo:

- THCAD +5 V a TB3 Pin 6 (+5 VP)
- THCAD -5 V a TB3 Pin 1 (GND)
- THCAD FOUT+ a TB3 Pin 7 (ENC A+)
- THCAD FOUT- a TB3 Pin 8 (ENC A-)

2.6.17.2. Prueba inicial de THCAD

Asegúrese de tener las siguientes líneas en su archivo INI (suponiendo una Mesa 7i76e):

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

Encienda su controlador y abra Halshow (AXIS: Mostrar configuración de homing), profundice para encontrar el pin `hm2_7i76e.0.encoder.00.velocity pin`. Con 0 voltios aplicados, debería estar rondando la frecuencia de 0 voltios (3,800 en nuestro ejemplo). Tome una batería de 9 voltios y conéctela a I_N+ e I_N-. Para un THCAD-10 ahora puede calcular la velocidad esperada. (26,480 en nuestro ejemplo hipotético). Si pasa esta prueba, está listo para configurar su controlador de plasma LinuxCNC.

2.6.17.3. ¿Cuál modelo THCAD utilizar?

El THCAD-5 es útil si tiene la intención de usarlo para la detección óhmica. No hay duda de que el THCAD-10 es el dispositivo más flexible y es fácil alterar la escala. Sin embargo, hay una advertencia que puede entrar en juego con algunos cortadores de plasma más baratos con un divisor de voltaje incorporado. Es decir, las resistencias internas pueden ser percibidas por el THCAD como parte de su propia resistencia externa y devolver resultados erróneos. Por ejemplo, el divisor 16:1 en las cortadoras de plasma Everlast debe tratarse como 24:1 (y 50:1 se convierte en 75:1). Esto no es un problema con marcas de mayor reputación (p. ej. Thermal Dynamics, Hypertherm, ESAB, etc.). Así que, si observa voltajes de corte inferiores a los esperados, podría ser preferible volver a configurar el THCAD para leer el voltaje de arco bruto.

Recordando que los voltajes de arco de plasma son potencialmente letales, aquí hay algunos criterios sugeridos.

Inicio del arco piloto Debido a que no es probable que haya ningún EMI significativo, debería poder instalar de forma segura el THCAD en su panel de control si ha seguido nuestras pautas de construcción.

- Si no tiene un divisor de voltaje, instale resistencias de escala dentro del cortador de plasma e instale el THCAD en el panel de control o siga las sugerencias para máquinas de arranque de alta frecuencia.
- Si tiene un divisor de voltaje, instale un THCAD-10 en su panel de control. No hemos tenido problemas con esta configuración con un cortador de plasma Thermal Dynamics de 120 A.

Inicio HF Instale el THCAD en el inversor ya que la señal de frecuencia es mucho más inmune al ruido EMI.

- Si no tiene un divisor de voltaje y tiene espacio dentro del cortador de plasma, instale un THCAD-300 dentro del cortador de plasma.
- Si no tiene un divisor de voltaje y no tiene espacio dentro del cortador de plasma, instale un THCAD-10 en una caja de metal fuera del cortador de plasma e instale el 50 % de la resistencia de escala en cada uno de los I_N+ e I_N- dentro la carcasa del cortador de plasma para que no salgan voltajes letales de la carcasa.
- Si tiene un divisor de voltaje, instale un THCAD-10 en una caja de metal fuera del cortador de plasma

Tensión de arco presentada en un conector En este caso, independientemente del método de inicio del arco, probablemente ya hayan resistencias incluidas en los circuitos para evitar choques letales, por lo que se recomienda un THCAD-10 para que esta resistencia (generalmente 200 kΩ) pueda tenerse en cuenta al elegir una resistencia de escala en tanto estas resistencias distorsionarán el voltaje reportado por el THCAD-300.

2.6.18. Post procesadores y anidamiento

El plasma no es diferente a otras operaciones de CNC, ya que:

1. Es diseñado en CAD (donde su salida es un formato DXF o, a veces, SVG).
2. Es procesado en CAM para generar el código G final que se carga en la máquina
3. Corta las piezas mediante comandos en código G de CNC.

Algunas personas logran buenos resultados con las herramientas Inkscape y G-Code, pero SheetCam es una solución de muy buen precio y hay una serie de posprocesadores disponibles para LinuxCNC. SheetCam tiene una serie de características avanzadas diseñadas para el corte por plasma y, por el precio, es una obviedad para cualquiera que realice regularmente corte por plasma.

2.6.19. Diseñando para entornos eléctricos ruidosos

El corte por plasma es inherentemente un entorno eléctrico extremadamente hostil y ruidoso. Si tiene problemas de EMI las cosas no funcionarán correctamente. En un ejemplo más que obvio, puede que la computadora se reinicie al encender la antorcha, pero puede tener cualquier cantidad de otros síntomas extraños. Casi todos sucederán solo cuando la antorcha esté cortando, y con frecuencia al primer disparo.

Por lo tanto, los creadores de sistemas deben seleccionar los componentes con cuidado y diseñar desde cero para hacer frente a este entorno hostil para evitar el impacto de la interferencia electromagnética (EMI). El fallar en esto puede resultar en incontables horas infructuosas de resolución de problemas.

Elegir placas ethernet como la Mesa 7i76e, o la más barata 7i96, ayuda al permitir que la PC se ubique lejos de la electrónica y la máquina de plasma. Este hardware también permite el uso de sistemas

lógicos de 24 voltios que son mucho más tolerantes al ruido. Los componentes deben montarse en una carcasa metálica conectada a la tierra de la red electrica. Se recomienda encarecidamente instalar un filtro EMI en la conexión de alimentación de red. La forma más sencilla es utilizar un conector IEC de alimentación de red con filtro EMI de uso común en PC y electrodomésticos, lo que permite lograrlo sin trabajo adicional. Planifique la disposición de los componentes en el cuadro para que la alimentación de la red, los cables de motor de alto voltaje y las señales lógicas se mantengan lo más separadas posible entre sí. Si se tienen que cruzar, manténgalas a 90 grados.

Peter Wallace de Mesa Electronics sugiere: “Si tiene una fuente de plasma compatible con CNC con un divisor de voltaje, montaría el THCAD dentro de su caja electrónica con el resto del hardware de movimiento. Si tiene una fuente de plasma manual y está leyendo voltaje de plasma sin procesar, montaría el THCAD lo más cerca posible de la fuente de plasma (incluso dentro de la caja de la fuente de plasma si cabe). En este caso, asegúrese de que todo el lado bajo de las conexiones THCAD están completamente aisladas de la fuente de plasma. Si utiliza una caja blindada para el THCAD, el blindaje debe conectarse a la tierra de su caja de electrónica, no a la tierra de la fuente de plasma.”

Se recomienda poner cables de tierra individuales para los motores y la antorcha desde sus cajas regresando a un punto central de conexión a tierra en la máquina. Conecte el cable de tierra de plasma a este punto y, opcionalmente, una varilla de tierra clavada en el suelo lo más cerca posible de la máquina (especialmente si se trata de una máquina de plasma de inicio por alta frecuencia).

El cableado externo a los motores debe estar blindado y dimensionado adecuadamente para manejar la corriente que pasa por el circuito. El blindaje debe dejarse desconectado en el extremo del motor y conectado a tierra en el extremo de la caja de control. Considere usar un pin adicional en cualquier conector en la caja de control para que la tierra pueda extenderse hasta la caja de control y conectarse a tierra al chasis directamente en el controlador del motor paso a paso / servo.

Somos conscientes de que al menos un fabricante de sistemas comerciales ha tenido problemas con el ruido eléctrico inducido en el circuito de detección óhmico. Si bien esto puede mitigarse utilizando toroides de ferrita y enrollando el cable, también se recomienda agregar un filtro de alimentación a través de la línea de alimentación donde la señal de detección óhmica ingresa al gabinete de la electrónica.

Tommy Berisha, el maestro de la construcción de máquinas de plasma con un presupuesto ajustado dice: “Si tiene un presupuesto limitado, considere usar viejos alimentadores de energía para computadoras portátiles. Son muy buenos, el filtrado es bueno, completamente aislado, limitados por la corriente (esto se vuelve muy importante cuando algo sale mal), y ajustar 2 o 3 de ellos en serie es fácil ya que están aislados. Tenga en cuenta que algunos tienen la conexión a tierra al terminal de salida negativa, por lo que tiene que ser desconectado, simplemente haciendo uso de un cable de alimentación sin contactos a tierra.”

2.6.20. Mesas por agua

El nivel mínimo de agua debajo del nivel de corte de la antorcha debe ser de alrededor de 40 mm, es bueno tener espacio debajo de los listones para que el agua pueda nivelarse y escapar durante el corte, y es muy bueno tener un poco de agua sobre la placa de metal que se está cortando en la medida que se deshace del polvo, dejarlo sumergido es la mejor manera, pero no es preferible para sistemas con uso a tiempo parcial, ya que corroerá la antorcha. Agregar bicarbonato de sodio al agua mantendrá la mesa en buenas condiciones durante muchos años, ya que no permite la corrosión mientras los listones están bajo el agua y también reduce el olor del vapor de agua. Algunas personas usan un depósito de agua con una entrada de aire comprimido para poder empujar el agua desde el depósito hacia la mesa a demanda y así permitir cambios en el nivel del agua.

2.6.21. Mesas de tiro descendente

Muchas mesas comerciales utilizan un diseño de tiro descendente, por lo que se usan ventiladores para aspirar el aire a través de los listones para capturar humos y chispas. A menudo, las mesas

se dividen en zonas, por lo que solo se abre una sección debajo de la antorcha hacia la salida de ventilación, a menudo utilizando rams y solenoides de aire para abrir las persianas. La activación de estas zonas es relativamente sencilla si utiliza la posición de la articulacion o del eje desde uno de los pines de movimiento y el componente lincurve para mapear las zonas de tiro descendente al pin de salida indicado.

2.6.22. Diseñando para velocidad y aceleración

En el corte por plasma, la velocidad y la aceleración son los reyes. A mayor aceleración, menos necesitará la máquina reducir la velocidad al negociar curvas. Esto implica que el pórtico debe ser lo más ligero posible sin sacrificar la rigidez torsional. Una caja de aluminio de seccion 100 mm x 100 mm x 2 mm tiene una rigidez torsional equivalente a un perfil extruido de ranuras T de 80 mm x 80 mm, pero es un 62 % más liviana. Por tanto, ¿la facilidad de las ranuras en T sopesa el trabajo de construcción adicional?

2.6.23. Distancia recorrida por revolución del motor

Los motores paso a paso sufren de resonancia y es probable que un piñón de transmisión directa signifique que el motor está funcionando en condiciones desfavorables. Para máquinas de plasma se considera ideal una distancia alrededor de 15-25 mm por revolución del motor, pero incluso alrededor de 30 mm por revolución sigue siendo aceptable. Un tornillo de bolas de 5 mm de paso con una unidad de reducción de 3:1 o 5:1 es ideal para el eje Z.

2.6.24. Configuración de plasma con QtPlasmaC de LinuxCNC

QtPlasmaC que consta de un componente HAL (plasmac.hal) más configuraciones completas para la GUI QtPlasmaC, ha recibido considerables contribuciones de muchos en el movimiento de código abierto LinuxCNC que ha avanzado en la comprensión de los controladores de plasma desde alrededor de 2015. Se han realizado muchas pruebas y trabajos de desarrollo para lograr que QtPlasmaC alcance su estado de funcionamiento actual. Se ha incluido todo, desde el diseño del circuito hasta el control y la configuración del código G. Además, QtPlasmaC admite THC externos como el Proma 150, pero realmente se destaca cuando se combina con un controlador Mesa, ya que esto permite al integrador incluir el convertidor de voltaje a frecuencia Mesa THCAD, diseñado específicamente para lidiar con el entorno hostil de plasma.

QtPlasmaC está diseñado para ser independiente e incluye la capacidad de incluir tus tablas de corte, pero también incluye características para ser utilizadas con un postprocesador como SheetCam.

El sistema QtPlasmaC ahora está incluido en la Versión 2.9 y superior de LinuxCNC. Ahora esta bastante maduro y se ha mejorado significativamente desde que se escribió la primera versión de esta guía. QtPlasmaC definirá el soporte de plasma de LinuxCNC por muchos años venideros, ya que incluye todas las características de un sistema patentado de control de plasma de gama alta a un precio de código abierto.

2.6.25. Control Hypertherm RS485

Algunas cortadoras de plasma Hypertherm tienen una interfaz RS485 para permitir que el controlador (por ejemplo, LinuxCNC) establezca amperios presión y el modo. Varias personas han utilizado un componente de espacio de usuario escrito en Python para lograr esto. Desde hace poco, QtPlasmaC admite ahora esta interfaz de forma nativa. Consulte la documentación de QtPlasmaC para saber cómo usarlo.

La combinación de una velocidad de transmisión lenta utilizada por Hypertherm y el componente de espacio de usuario hace que esto sea bastante lento para alterar los estados de la máquina, por lo que generalmente no es viable cambiar la configuración sobre la marcha mientras se corta.

Al seleccionar una interfaz RS485 para usar en el extremo de la PC, los usuarios informaron que las interfaces USB a RS485 no son confiables. Se han logrado resultados buenos y confiables utilizando una interfaz RS232 basada en hardware (p. ej. PCI/PCIe o puerto de placa base) y un convertidor RS485 apropiado. Algunos usuarios han informado éxito con una tarjeta Sunix N/P:SER5037A PCI RS2322 y un convertidor genérico XC4136 RS232 a RS485 (que a veces también puede incluir un cable USB).

2.6.26. Postprocesadores para corte por plasma

Los programas CAM (Fabricación asistida por computadora) son el puente entre CAD (Diseño asistido por computadora) y la operación final del CNC (Control numérico por computadora). A menudo incluyen un postprocesador configurable por el usuario para definir el código que se genera para una máquina o dialecto específico de G-Code.

Muchos usuarios de LinuxCNC están muy contentos con Inkscape para convertir archivos de vectores SVG en código G. Considera esta opción si haces corte por plasma como pasatiempo o para uso doméstico.

Sin embargo, si sus necesidades son más complejas, probablemente la mejor solución de precio más razonable sea SheetCam. SheetCam es compatible con Windows y Linux y hay postprocesadores disponibles, incluida la configuración de QtPlasmaC. SheetCam le permite anidar partes sobre una hoja completa de material y le permite configurar conjuntos de herramientas y fragmentos de código para satisfacer sus necesidades. Los postprocesadores SheetCam son archivos de texto escritos en el lenguaje de programación Lua y, en general, son fáciles de modificar para adaptarse a sus requisitos exactos. Para obtener más información, consulte el sitio web de SheetCam y su foro de soporte.

Se incluye otro postprocesador popular con el popular paquete Fusion360, pero los postprocesadores incluidos necesitarán cierta personalización.

LinuxCNC es una aplicación CNC y las discusiones sobre técnicas CAM distintas a esta discusión introductoria están fuera del alcance de LinuxCNC.

Capítulo 3

Asistentes de Configuración

3.1. Asistente de configuración de pasos

3.1.1. Introducción

LinuxCNC es capaz de controlar una amplia variedad de maquinaria, utilizando diferentes interfaces de hardware.

StepConf es un programa que genera archivos de configuración para LinuxCNC para un tipo especifico de máquina CNC: aquellas que son controladas a través de un *puerto paralelo estándar* con señales de tipo *paso y dirección*.

StepConf se instala automaticamente cuando se instala LinuxCNC; se encuentra en el menú CNC.

El asistente genera un archivo en el directorio linuxcnc/config en el cual guarda las opciones de cada configuración que usted genere. Cuando se desea cambia algo, se necesita seleccionar el archivo cuyo nombre coincida con la configuración que desea modificar. La extensión del archivo es .stepconf.

El asistente StepConf necesita una resolución de pantalla de al menos 800 x 600.

3.1.2. Página inicial

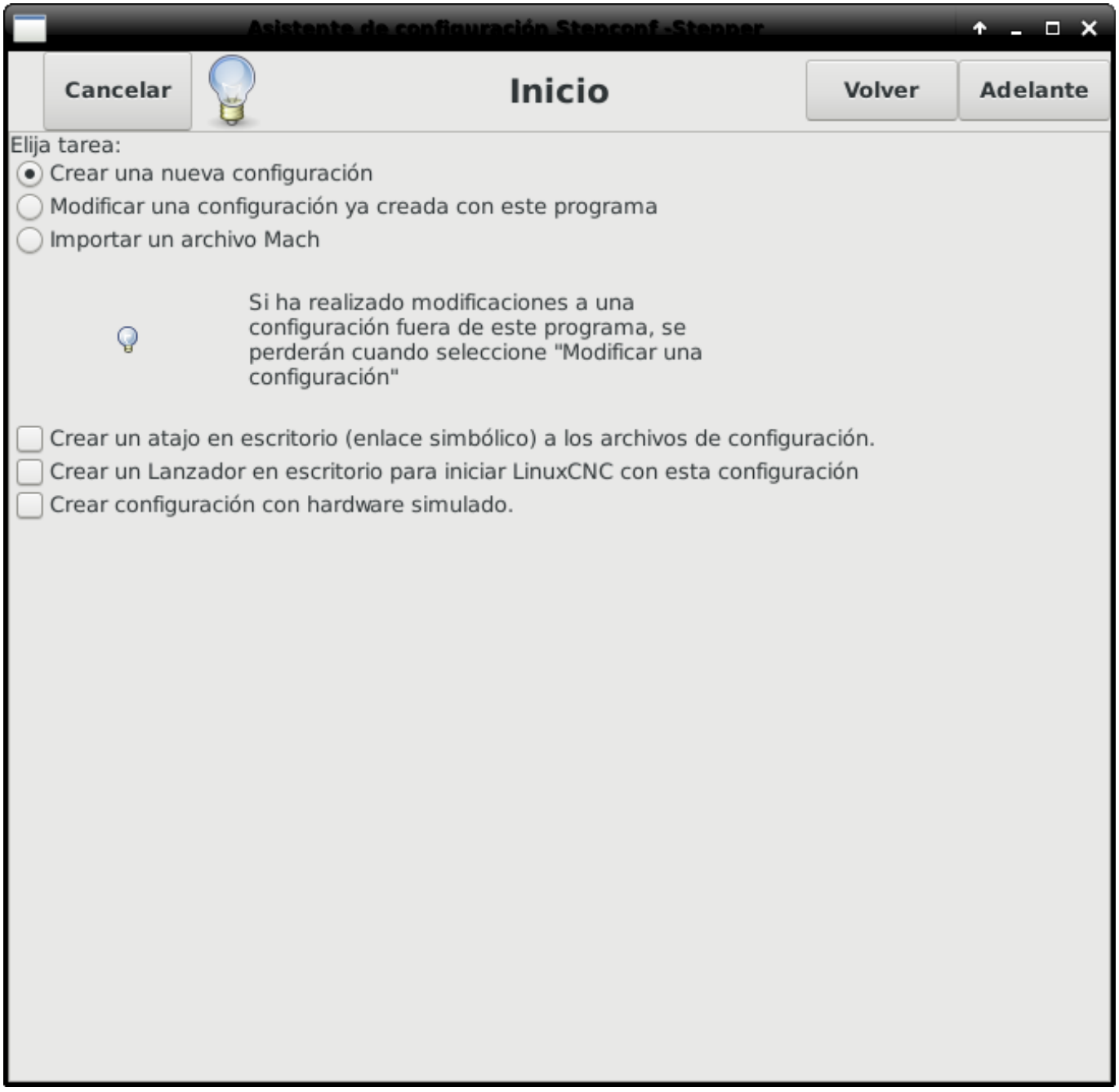


Figura 3.1: Página de entrada de StepConf

Los tres primeros botones radio son autoexplicativos:

- *Crear una nueva configuración* - Crea una configuración nueva.
- *Modificar una configuración ya creada…* - Modifica una configuración existente. Después de seleccionar esta opción, aparecerá una pantalla de selección de archivo y usted deberá seleccionar el archivo con extensión .stepconf que desea modificar. Si usted realizó alguna modificación previa a

los archivos principales .hal o .ini, estas modificaciones se perderán. Las modificaciones en los archivos custom.hal y custom_postgui.hal no serán cambiadas por el asistente Stepconf. El asistente Stepconf resaltará la ultima configuración que se construyó.

- *Importar* - Importa un archivo de configuración Mach e intenta convertirlo a un archivo de configuración de LinuxCNC. Después de la importación, pase por las páginas de StepConf para confirmar/modificar las entradas. El archivo mach xml original no será modificado.

Las siguientes opciones se registrarán en un archivo de preferencias para la próxima ejecución de StepConf.

- *Crear atajo en escritorio* - Se generará en su escritorio un enlace a los archivos.
- *Crear un lanzador en el escritorio* - Se generará un lanzador en su escritorio para iniciar la aplicación.
- *Crear hardware simulado* - Permite crear una configuración para pruebas, incluso si no tiene el hardware real.

3.1.3. Información básica



Figura 3.2: Página de información básica

- *Crear hardware simulado* - Permite crear una configuración para pruebas, incluso si no tiene el hardware real.
- *Nombre de la máquina* - Elija un nombre para su máquina. Utilice solo letras mayúsculas, minúsculas, dígitos, - y _.
- *Configuración de ejes* - Seleccione XYZ (Fresadora), XYZA (Fresadora de 4 ejes) o XZ (Torno).

- *Unidades de máquina* - Seleccione pulgadas o milímetros. Todas las preguntas posteriores estarán en las unidades seleccionadas. Cambiar esto también cambia los valores predeterminados en la sección de Ejes. Si cambias esto después seleccionar valores en cualquiera de las secciones de ejes, serán sobrescritos por los valores predeterminados de las unidades seleccionadas.

- *Tipo de driver* - Si usted tiene uno de los controladores de motor a pasos de la lista de selección selecciónelo directamente. De lo contrario, seleccione *Otro* y busque los 4 valores de tiempo en la hoja de especificaciones de su controlador y introdúzcalos en *nano segundos* en la sección de *Configuración de tiempos del controlador*. Si sus manuales le dan los datos en microsegundos multiplíquelos por 1000. Por ejemplo, si el manual indica 45 µs, escriba 4500 ns.

  En la página wiki de LinuxCNC.org puede ser consultada una lista de controladores populares, así como sus tiempos, en Tiempos de controladores de pasos.

  El acondicionamiento o aislamiento extra de señal, como cuando se usan optoacopladores y filtros RC en tarjetas breakboards, pueden imponer restricciones de tiempo por si mismos, además de las del controlador. Puede ser el caso que se requiera agregar tiempo extra a los valores de temporizacion para compensar los filtros o aislamientos.

  El Selector de configuración de LinuxCNC tiene configuraciones listas para máquinas de la marca Sherline, en caso de que posea una de estas. * *Tiempo de paso* - Cuánto tiempo es *Encendido* el pulso de paso en nanosegundos. Si no estas seguro de esta configuración, un valor de 20,000 funcionará con la mayoría de los controladores. * *Espacio de paso* - Tiempo mínimo entre dos pulsos de paso en nanosegundos. Si no estas seguro de esta configuración, un valor de 20,000 funcionará con la mayoría de los controladores. * *Sostenimiento de dirección* - Tiempo que debe ser mantenido activo el pin de dirección después de ordenar cambio de dirección, en nanosegundos. Si no estas seguro de esta configuración, un valor de 20,000 funcionará con la mayoría de los controladores. * *Preparación de dirección* - Cuánto tiempo debe preceder a un cambio de dirección después del último pulso de paso en nanosegundos. Si no estas seguro de esta configuración, un valor de 20,000 funcionará con la mayoría de los controladores. * *Uno / dos puertos paralelos* - Seleccione cuántos puertos paralelos se van a configurar. * *Fluctuación máxima de período base* - Introduzca aquí el resultado de la prueba de latencia. Para correr la prueba de latencia presione el botón *Prueba de fluctuación de período base*. Vea la sección de la Prueba de latencia para más detalles.

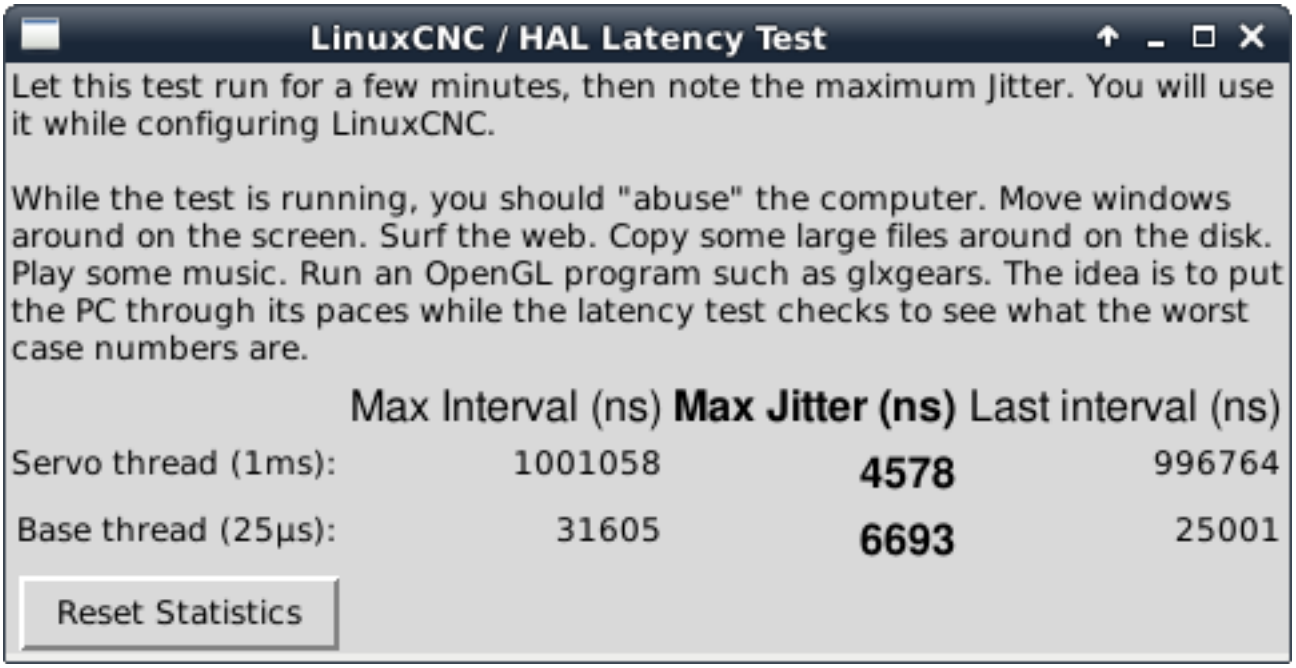


Figura 3.3: Prueba de latencia

- *Velocidad de paso máxima* - StepConf calculará automáticamente la tasa máxima de pulsos de pasos basándose en las características del controlador del motor capturadas y en el resultado de la prueba de latencia.
- *Período base mínimo* - StepConf calculará automáticamente el periodo base mínimo basándose en las características del controlador del motor capturadas y el resultado de la prueba de latencia.

El número importante de la prueba de latencia es la *Fluctuación máxima*. En el ejemplo de arriba la fluctuación máxima es de 9075 nanosegundos (ns) o 9.075 microsegundos (µs). Introduce la fluctuación máxima en el cuadro de Período base de fluctuación máxima.

3.1.4. Ajustes del puerto paralelo

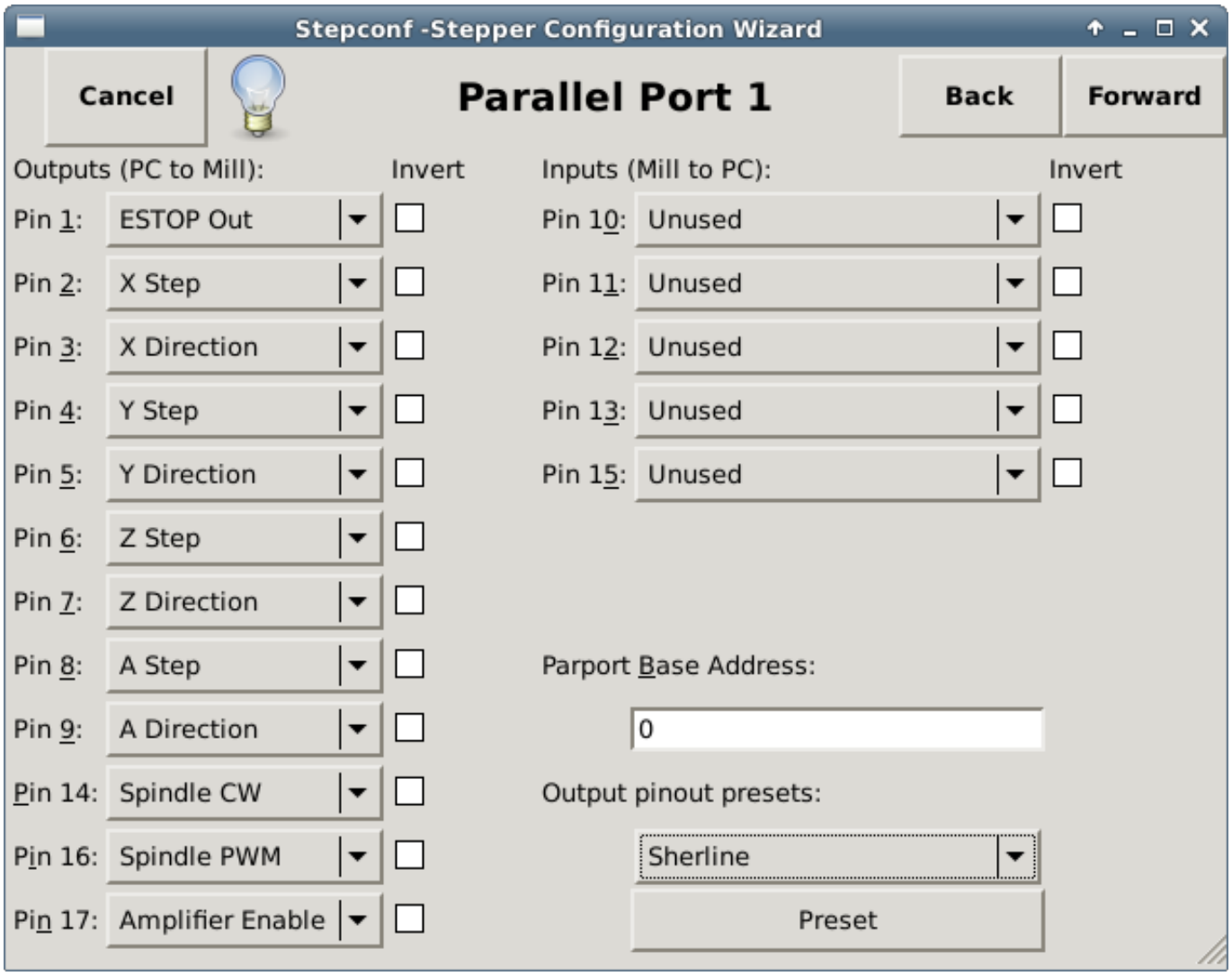


Figura 3.4: Página de ajuste del puerto paralelo

Puede especificar la dirección como un número hexadecimal (por lo regular 0x378) o como un número de puerto predeterminado de Linux (probablemente 0)

Para cada pin, elija la señal que coincida con la asignación de pines de su puerto paralelo. Active la casilla de verificación *Invertir* si la señal de control requiere ser invertida (0V para verdadero/activo, 5V para falso/inactivo).

- *Asignación de pines de salida predefinidos* - Se configurarán automáticamente los pines del 2 al 9 de acuerdo al estándar de las máquinas Sherline (Dirección en los pines 2, 4, 6, 8) o Xylotex (Dirección en los pines 3, 5, 7, 9).
- *Entradas y salidas* - Si el pin no será utilizado como entrada o salida, seleccionarlo como *Sin uso*.
- *Señal de paro externo (E stop)* - Esta señal pude ser seleccionada en una de las listas desplegables. Una cadena de señal de paro típica utiliza solo contactos en serie normalmente cerrados.
- *Interruptores de posición home y límites de seguridad* - Estos pines pueden ser seleccionados para la mayoría de las configuraciones utilizando las listas desplegables.
- *Bomba de Carga* - Si su controlador de motor requiere de una señal de bomba de carga, simplemente seleccione esta opción de la lista desplegable y conecte la señal al pin seleccionado. La salida de la bomba de carga será conectada al hilo base por el programa StepConf. La salida de bomba de carga será aproximadamente 1/2 de la tasa de generación de pulsos de paso máxima mostrada en la página de configuración básica.
- *Voltaje de arco de plasma* - Si requiere un THCAD de Mesa para ingresar un voltaje de arco de plasma entonces seleccione Voltaje de arco de plasma en la lista de pines de salida. Esto habilitará una página para THCAD durante el procedimiento de configuración para la captura de los parámetros de la tarjeta.

3.1.5. Configuración del puerto paralelo 2

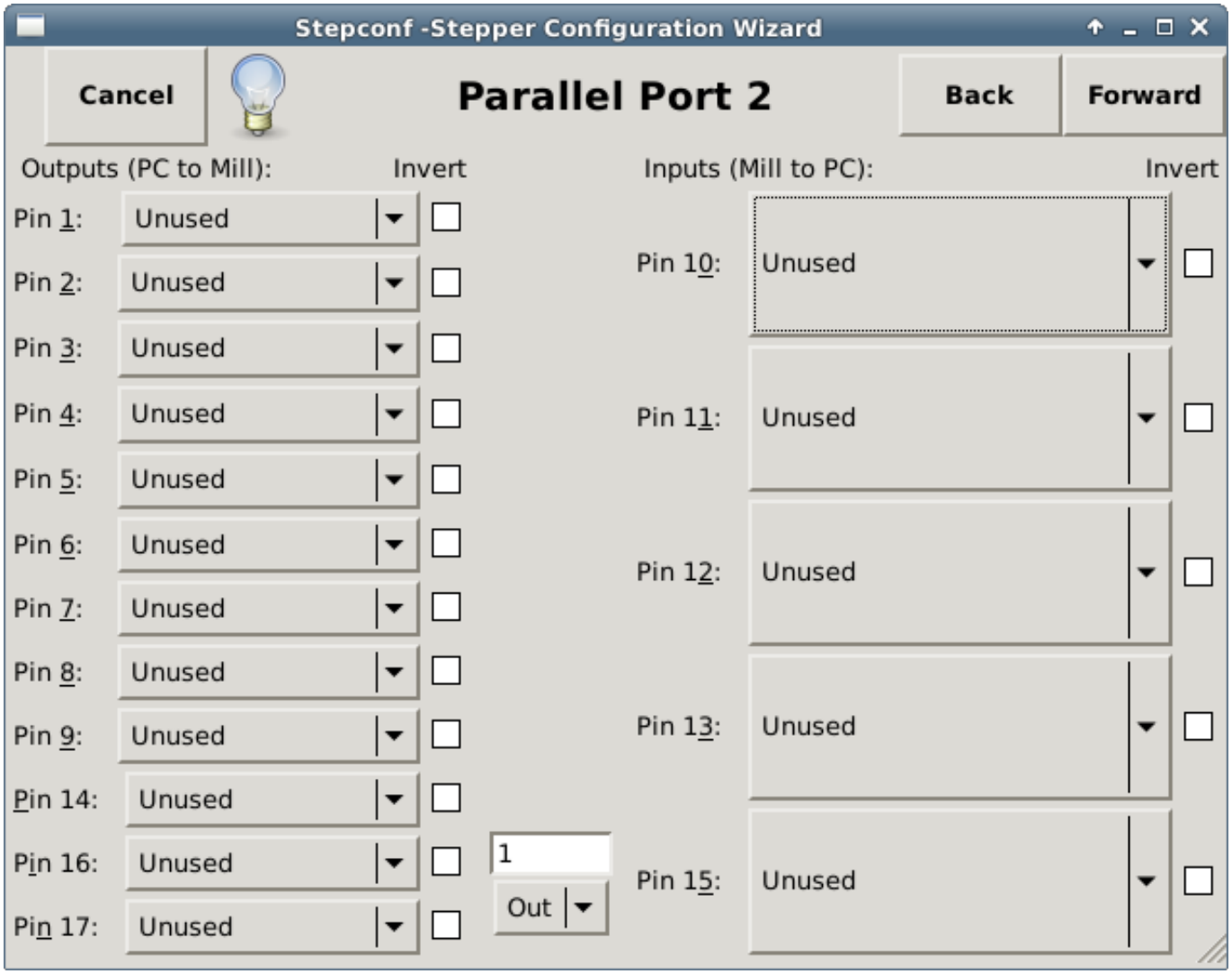


Figura 3.5: Página de configuración del puerto paralelo 2

El segundo puerto paralelo (si está seleccionado) puede ser configurado y asignar sus pines en esta página. No se pueden seleccionar señales de paso y dirección. Puede seleccionarlo de entrada o de salida para maximizar el número de pines de entrada/salida que estén disponibles. Puede especificar la dirección como hexadecimal (a menudo 0x378) o como el número de puerto predeterminado de Linux (probablemente 1).

3.1.6. Configuración de ejes

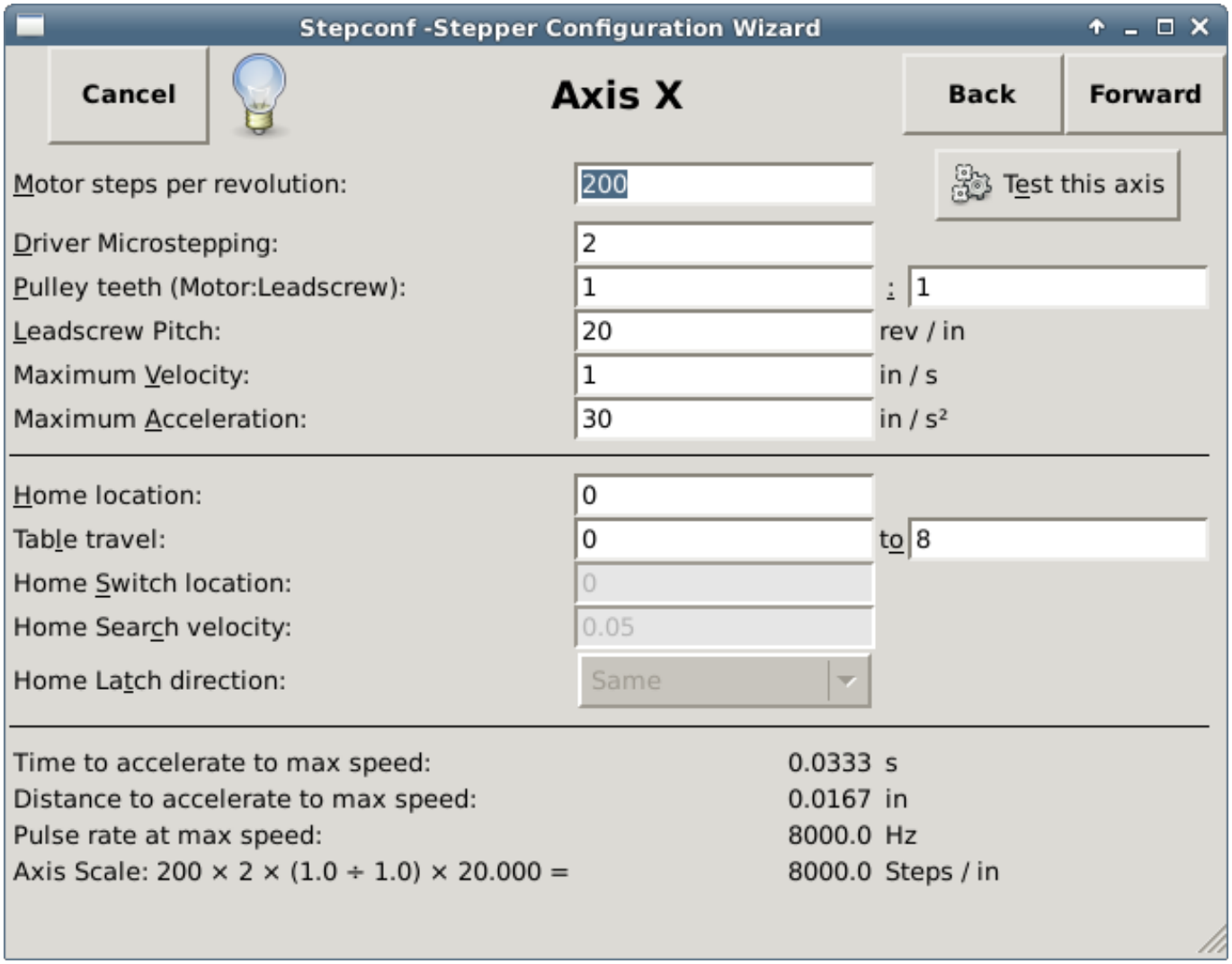


Figura 3.6: Página de configuración de ejes

- *Pasos del motor por revolución* - El número de pasos completos por revolución del motor. Si solo se tiene el dato de los grados por paso del motor (p. ej. 1.8 grados), se debe dividir 360 por el número de grados por paso para encontrar el número de pasos por revolución del motor.
- *Micropasos del controlador* - El número de micropasos producidos por el controlador. Introduzca *2* para semipasos.
- *Relación de polea* - Si su máquina tiene poleas o engranes entre el motor y el tornillo, escriba su relación mecánica aquí. Si no tiene, escriba *1:1*.
- *Paso de tornillo* - Introduce aquí el paso del tornillo. Si seleccionaste unidades *Pulgadas*, teclea el número de hilos por pulgada (por ejemplo, 8 para un tornillo de 8 TPI). Si se tiene un tornillo con varias entradas, se necesita saber cuantas vueltas se requieren para mover la tuerca una pulgada. Si seleccionaste *mm* como unidades, teclea el número de milímetros que la tuerca se moverá por revolución (ejemplo, 2 para 2 mm/rev). Si la máquina se mueve en la dirección opuesta a la esperada, teclea un valor negativo en lugar de un número positivo, o invierte el pin de dirección del eje.
- *Velocidad máxima* - Entre la velocidad máxima del eje, en unidades por segundo.

- *Aceleración máxima* - El valor correcto de esta casilla solo puede ser determinado por experimentación. Vea Encontrar velocidad máxima para ajustar la velocidad y Encontrar aceleración máxima para ajustar la aceleración.
- *Posición de casa* - La posición a la que la máquina se moverá después de completar el procedimiento de inicio del eje. Para máquinas sin interruptores de posición casa, esta es la posición a la cual el operador deberá mover la máquina antes de presionar el botón de casa. Si se combinan los interruptores de casa y limite, se deberá mover la máquina fuera del interruptor para inicializar el eje o se recibirá un error de límite de articulación.
- *Carrera* - El rango de carrera para el eje con base en el origen de la máquina. La posición de casa debe estar dentro del área de carrera y ser distinto a cualquiera de los valores de rango de carrera.
- *Posición del interruptor de casa* - La ubicación en la cual el interruptor de casa se activa o desactiva, relativa al origen de la máquina. Este apartado y los dos siguientes solo aparecerán cuando se selecciona la existencia de interruptores casa en la asignación de pines del puerto paralelo. Si se combinan los interruptores de casa y límite, la ubicación del interruptor casa no puede ser la misma que la posición casa o se producirá un error de límite de articulación.
- *Velocidad de búsqueda de casa* - Velocidad a usarse en la búsqueda de los interruptores casa. Si el interruptor se encuentra cercano al limite de carrera del eje, esta velocidad debe ser seleccionada de tal forma que el eje tenga tiempo suficiente para desacelerar hasta detenerse antes de llegar al límite físico de la carrera. Si el interruptor se encuentra cerrado en un rango corto de carrera (en lugar de estar cerrado desde el punto de disparo hasta un final de carrera) esta velocidad deberá ser seleccionada de tal forma que el eje pueda desacelerar hasta detenerse antes de que el interruptor se abra otra vez, y el procedimiento de homing deberá comenzarse siempre desde el mismo lado del interruptor. Si la máquina se mueve en la dirección contraria al inicio del homing, cambie el signo del parámetro *Velocidad de búsqueda de casa*.
- *Dirección del pestillo de casa* - Seleccione *Mismo* para que el eje se aleje del interruptor y luego se acerque de vuelta a una velocidad muy baja. La segunda vez que se cierre el interruptor definirá la posición de casa. Seleccione *Opuesto* para hacer que eje se aleje liberando lentamente el interruptor; cuando el interruptor se abra, se marcará la posición casa.
- *Tiempo para acelerar a velocidad máxima* - Tiempo para alcanzar la velocidad máxima, calculado a partir de *Aceleración máxima* y *Velocidad máxima*.
- *Distancia para acelerar a velocidad máxima* - Distancia para alcanzar máxima velocidad desde posición de parado.
- *Frecuencia de pulsos a velocidad máxima* - Este dato se calcula con base en los valores anteriores. El valor más grande de *Tasa de pulso a velocidad máxima* determina *BASE_PERIOD*. Valores por encima de 20000Hz pueden producir tiempos de respuesta muy bajos o incluso bloqueos (la tasa máxima varia entre computadoras)
- *Escala del eje* - El número que será usado en el archivo INI en la sección [SCALE]. Representa cuántos pasos se deben dar por unidad de usuario.
- *Probar este eje* - Esta opción abre una ventana para permitir probar cada eje. Puede ser utilizada después de llenar toda la información referente a este eje.

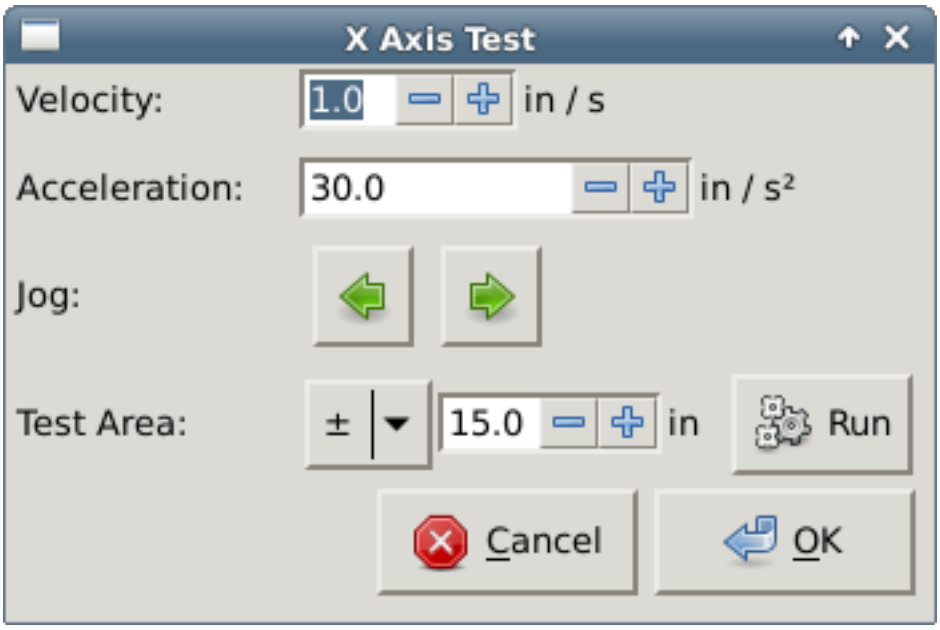


Figura 3.7: Prueba de eje

Es un comprobador básico que solo emite señales de paso y dirección para probar diferentes valores de aceleración y velocidad.

importante
Para utilizar la prueba de este eje, debe habilitar manualmente el eje si se requiere. Si su controlador tiene una bomba de carga, tendrá que puentear para pasarla por alto. La prueba del eje no reacciona a las entradas de interruptor de límite, úsese con precaución.

3.1.6.1. Determinar la velocidad máxima

Comience con una baja aceleración (por ejemplo, **2 pulgadas/s²** o **50 mm/s²**) y la velocidad que espera alcanzar. Usando los botones provistos, mueva el eje hasta cerca del centro de su carrera. Tenga cuidado, porque con un valor de aceleración bajo puede recorrerse una distancia sorprendentemente larga hasta que el eje desacelere completamente y pare.

Después de calcular la cantidad de carrera disponible, ingrese una distancia segura en el Área de prueba, teniendo en cuenta que después de un bloqueo, el motor puede comenzar a moverse en una dirección inesperada. Luego haga clic en Ejecutar. La máquina comenzará a avanzar y retroceder a lo largo de este eje. En esta prueba, es importante que la combinación de aceleración y área de prueba permita que la máquina alcance la velocidad seleccionada y la mantenga al menos una corta distancia, cuanto mayor sea la distancia, mejor será esta prueba. La fórmula **d=0.5 * v * v/a** da la distancia mínima requerida para alcanzar la velocidad especificada con la aceleración dada. Si es conveniente y seguro hacerlo, empuje la mesa contra la dirección del movimiento para simular las fuerzas de corte. Si la máquina se para, reduzca la velocidad y comience nuevamente la prueba.

Si la máquina no se paró, haga clic en el botón *Ejecutar* para parar. El eje vuelve ahora a la posición donde comenzó. Si la posición es incorrecta, el eje se estancó o perdió pasos durante la prueba. Reduzca la velocidad y comience la prueba otra vez.

Si la máquina no se mueve, se detiene o pierde pasos, incluso bajando la velocidad, verifique lo siguiente:

- Tiempos de onda de paso correctos
- Asignación de pines correcta, incluyendo *Invertir* en los pines de paso
- Cableado correcto y bien protegido
- Problemas físicos con el motor, acoplamiento del motor, tornillo de avance, etc.

Una vez que haya encontrado una velocidad a la que el eje no se detiene o pierde pasos durante este procedimiento de prueba, reducirlo en un 10 % y usarlo como la *Velocidad máxima* del eje.

3.1.6.2. Determinando la aceleración máxima

Con la velocidad máxima que encontró en el paso anterior, ingrese el valor de aceleración a probar. Usando el mismo procedimiento anterior, ajuste el valor de Aceleración hacia arriba o hacia abajo según sea necesario. En esta prueba, es importante que la combinación de aceleración y área de prueba permitan que la máquina alcance la velocidad seleccionada. Una vez que haya encontrado un valor en el que el eje no se detiene ni pierde pasos durante este procedimiento de prueba, reducirlo en un 10 % y usarlo como Aceleración máxima del eje.

3.1.7. Configuración del husillo

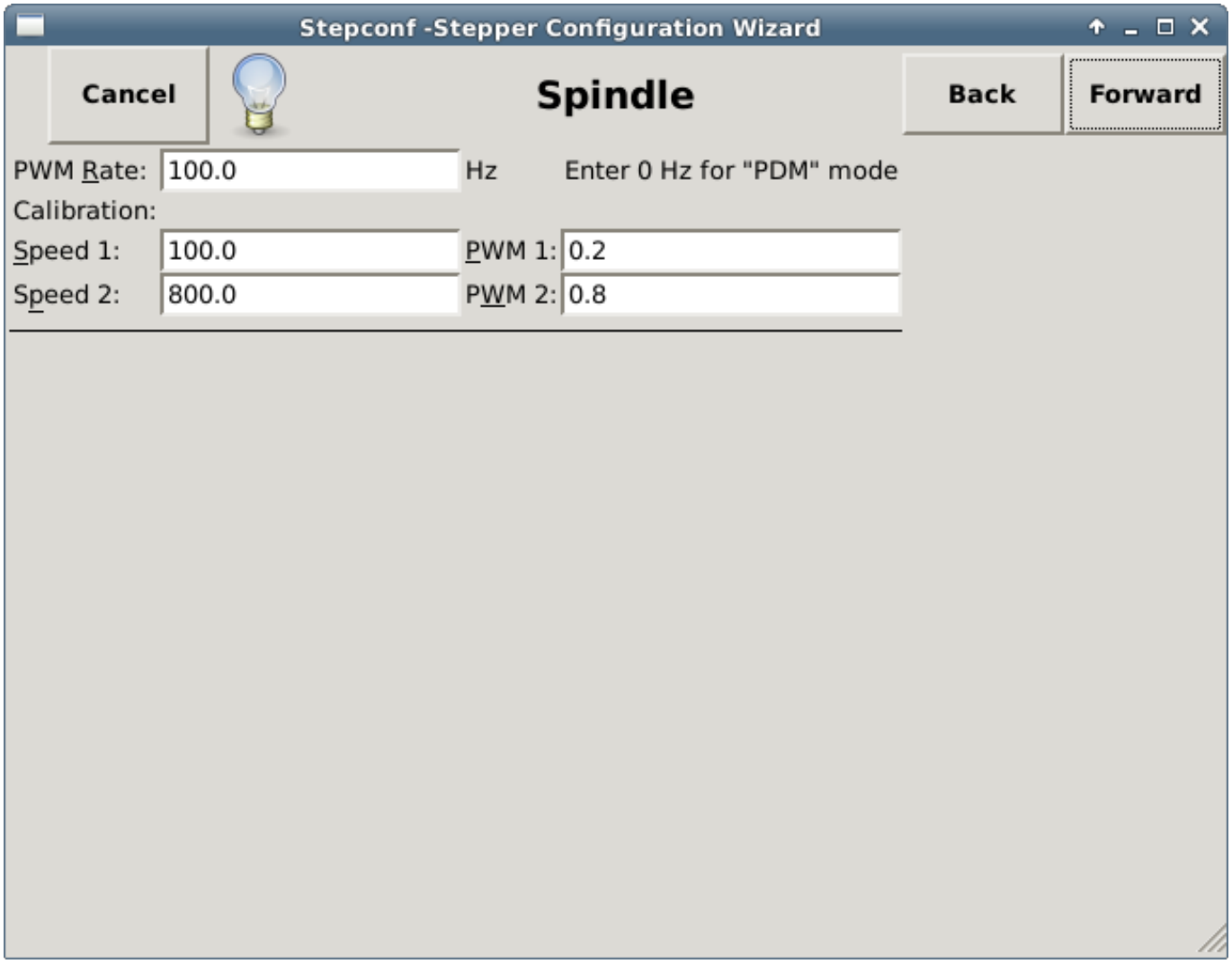


Figura 3.8: Página de configuración del husillo

Esta página solo aparece cuando se selecciona *Husillo PWM* en la página *Puerto paralelo* para una de las salidas.

3.1.7.1. Control de velocidad del husillo

Si hay *Husillo PWM* en la asignación de pines, debe aportarse la siguiente información:

- *Frecuencia PWM* - La *frecuencia portadora* de la señal PWM al husillo. Entrar *0* para el modo PDM, que es útil para generar un voltaje de control analógico. Consulte la documentación de su controlador de husillo para conocer el valor apropiado.
- *Velocidad 1 y 2, PWM 1 y 2*: el archivo de configuración generado utiliza una relación lineal simple para determinar el valor PWM para un valor RPM dado. Si los valores no se conocen, se pueden determinar. Para más información, ver Determinación de la calibración del husillo.

3.1.7.2. Movimiento sincronizado del husillo

Cuando las señales apropiadas de un codificador de husillo están conectadas a LinuxCNC a través de HAL, LinuxCNC admite el roscado en torno. Estas señales son:

- *Índice del husillo* - Es un pulso que ocurre una vez por revolución del husillo.
- *Fase A del husillo* - Este es un pulso que ocurre en múltiples ubicaciones, igualmente espaciadas, a medida que gira el husillo.
- *Fase B del husillo (opcional)* - Este es un segundo pulso, pero con un desplazamiento de la fase A del husillo. Las ventajas de usar tanto A como B son detección de dirección, mayor inmunidad al ruido y mayor resolución.

Si hay *Fase A de husillo* e *Índice de husillo* en la asignación de pines, se debe ingresar la siguiente información:

- *Usar husillo a velocidad* - Con la retroalimentación del codificador se puede hacer que LinuxCNC espere a que el husillo alcance la velocidad ordenada antes de que se mueva la alimentación. Seleccione esta opción y establezca la escala *lo más cercana*.
- *Ganancia del filtro de visualización de velocidad* - Configuración para ajustar la estabilidad de la visualización de la velocidad del husillo.
- *Ciclos por revolución* - El número de ciclos de la señal A del husillo durante una revolución. Esta opción solo está habilitada cuando una entrada se ha configurado como *Fase A del husillo*
- *Velocidad máxima en roscado* - La velocidad máxima del husillo utilizada en el roscado. Para un husillo de altas RPM o un codificador de husillo con alta resolución, es obligatorio un valor bajo de *BASE_PERIOD* .

3.1.7.3. Determinación de la calibración del husillo

Ingrese los siguientes valores en la página configuración del husillo:

Velocidad 1:	0	**PWM 1:**	0
Velocidad 2:	1000	**PWM 2:**	1

Termine los pasos restantes del proceso de configuración, luego, inicie LinuxCNC con su configuración. Encienda la máquina y seleccione la pestaña MDI. Inicie el giro del husillo ingresando: *M3 S100*. Cambie la velocidad del husillo ingresando un número S diferente: *S800*. Los números válidos (en este momento) van de 1 a 1000.

Para dos números S diferentes, mida la velocidad real del eje en RPM. Registre los números S y las velocidades reales del eje. Ejecute StepConf nuevamente. Para *Velocidad*, ingrese la velocidad medida, y para *PWM* ingrese el número S dividido entre 1000.

Ya que la mayoría de los controladores de husillo son ligeramente no lineales en sus curvas de respuesta, lo mejor es:

- Asegurarse que las dos velocidades de calibración no estén demasiado juntas en RPM.
- Asegurarse que las dos velocidades de calibración estén en el rango de velocidades que típicamente usará durante el fresado.

Por ejemplo, si su husillo va de 0 RPM a 8000 RPM, pero generalmente usa velocidades de 400 RPM (10 %) a 4000 RPM (100 %), encuentre los valores de PWM que dan 1600 RPM (40 %) y 2800 RPM (70 %).

3.1.8. Opciones

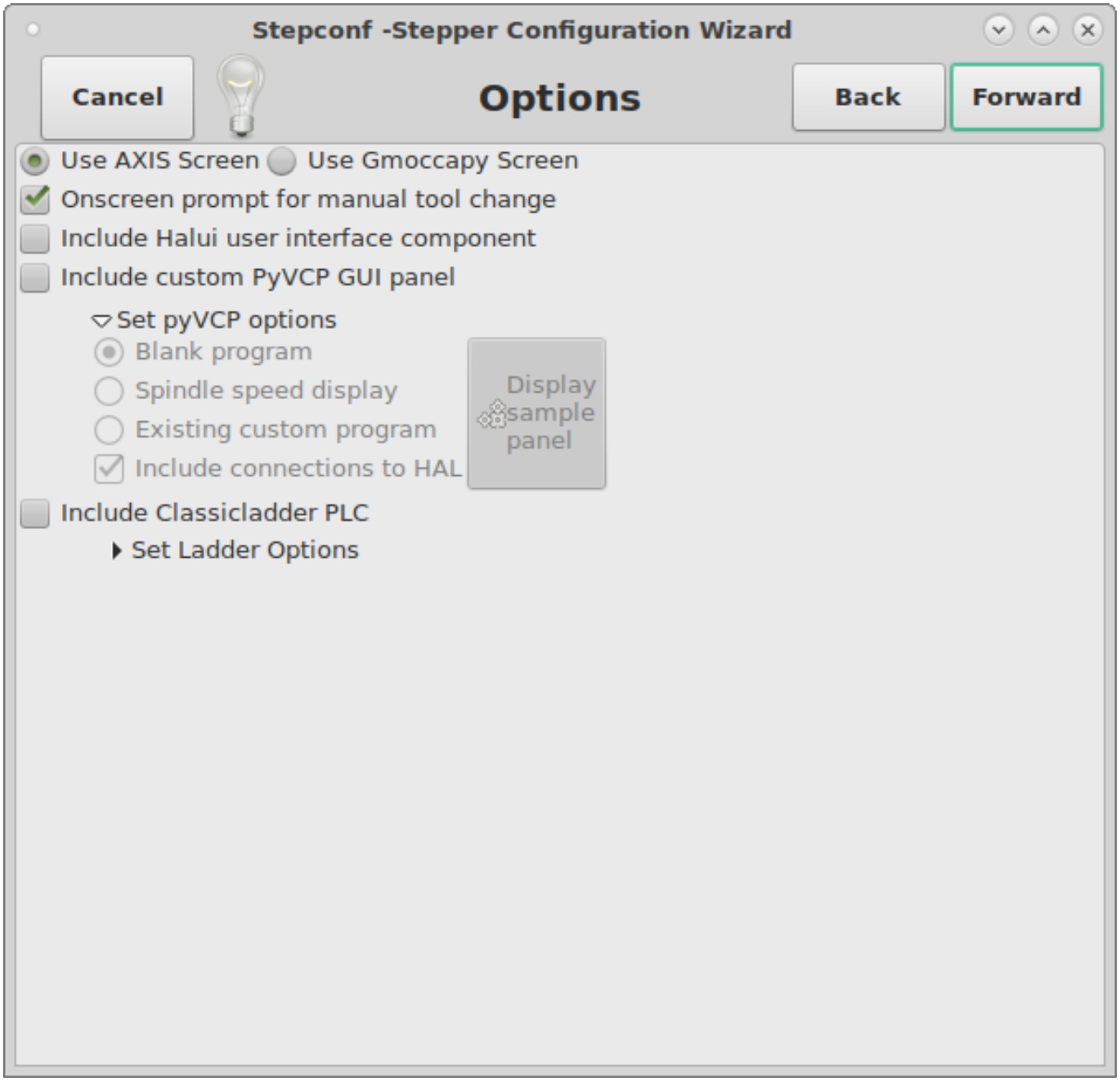


Figura 3.9: Configuración de opciones avanzadas

- *Incluir Halui* - Esto agregará el componente de interfaz de usuario Halui. Ver el capítulo HALUI para más información.
- *Incluir PyVCP* - Esta opción agrega el archivo base del panel PyVCP o un archivo ejemplo para trabajar en él. Ver el capítulo PyVCP para más información.
- *Incluir ClassicLadder PLC* - Esta opción agregará el PLC ClassicLadder (Controlador lógico programable). Ver el Capítulo ClassicLadder para más información.

- *Indicador en pantalla para cambio de herramienta* - Si esta casilla está marcada, LinuxCNC para y le pide que cambie la herramienta cuando se encuentre *M6*. Esta característica generalmente solo es útil si tiene herramientas predimensionadas.

3.1.9. Completar la configuración de la máquina

Haga clic en *Listo* para escribir los archivos de configuración. Más tarde, puede volver a ejecutar este programa y ajustar la configuración que ingresó antes.

3.1.10. Carrera y casa de los ejes

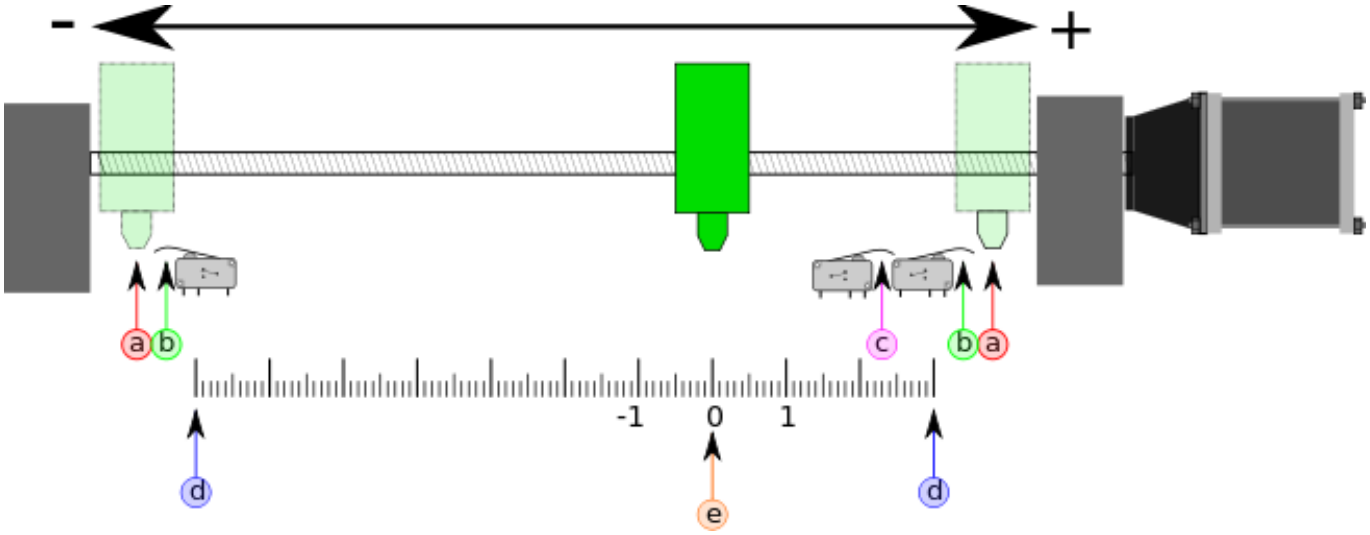


Figura 3.10: Carrera y casa de un eje

Para cada eje hay un rango limitado de recorrido. El final físico del recorrido se llama *parada dura*.

aviso
Si se va a sobrepasar una parada dura mecánica ¡se dañará el tornillo o el marco de la máquina!

Antes de la *parada dura* hay un *interruptor de límite*. Si durante la operación normal se topa con el interruptor de límite LinuxCNC apaga el amplificador de motor. La distancia entre la *parada dura* y el *interruptor de límite* debe ser lo suficientemente larga como para permitir que un motor sin alimentación llegue a detenerse.

Antes del *interruptor de límite* hay un *límite suave*. Este es un límite impuesto en el software después de homing. Si un comando MDI o un programa de código G sobrepasara el límite suave, no se ejecutará. Si un trote sobrepasara el límite suave, se detiene en el límite suave.

El *interruptor casa* se puede colocar en cualquier lugar dentro de la carrera (entre paradas duras). Siempre que el hardware externo no desactive los amplificadores de motor cuando se alcanza el interruptor de límite, uno de los interruptores de límite puede ser utilizado como un interruptor casa.

La *posición cero* es la ubicación en el eje que es 0 en el sistema de coordenadas de la máquina. Por lo general, la *posición cero* estará dentro de los *límites suaves*. En los tornos, el modo de velocidad de

superficie constante requiere que *X = 0* en la máquina corresponda al centro de rotación del husillo cuando no está activo el offset de la herramienta.

La *posición casa* es la ubicación dentro de la carrera a la que el eje será movido al final de la secuencia de casa. Este valor debe estar dentro de los *límites suaves*. En particular, la *posición casa* nunca debe ser exactamente igual a un *límite suave*.

3.1.10.1. Operar sin interruptores de limite

Una máquina puede ser operada sin interruptores de límite. En este caso, solo los límites suaves impiden que la máquina alcance la parada dura. Los límites suaves solo funcionan después de que la máquina este en casa.

3.1.10.2. Operando sin interruptores de casa

Una máquina puede operarse sin interruptores de casa. Si la máquina tiene interruptores de límite, pero no hay interruptores casa, lo mejor es utilizar un interruptor de límite como el interruptor casa (p. ej., elija *Límite mínimo + Home X* en la asignación de pines). Si la máquina no tiene interruptores en absoluto, o los interruptores de límite no pueden ser utilizados como interruptores de casa por otra razón, entonces la máquina debe ponerse en casa *a ojo* o usando marcadores de posición. Homing a ojo no es tan repetible como por interruptores, pero aún permite que los límites suaves sean de utilidad.

3.1.10.3. Opciones de cableado de los interruptores de casa y límite

El cableado ideal para interruptores externos sería de una entrada por interruptor. Sin embargo, el puerto paralelo de PC solo ofrece un total de 5 entradas, mientras que hay hasta 9 interruptores en una máquina de 3 ejes. En cambio, múltiples interruptores pueden conectarse entre sí en varias formas para que se requiera un menor número de entradas.

Las siguientes figuras muestran la idea general de cablear múltiples interruptores a un solo pin de entrada. En cada caso, cuando se activa un interruptor, el valor visto en ENTRADA va de lógica ALTA a BAJA. Sin embargo, LinuxCNC espera un valor VERDADERO cuando se cierra un interruptor, por lo que la casilla correspondiente *Invertir* debe verificarse en la página de configuración de asignación de pines. El resistor pull-up que se muestra en los diagramas jala la entrada alta hasta que se realice la conexión a tierra y la entrada pasa a baja. Sin resistencia, la entrada puede flotar entre encendido y apagado cuando el circuito está abierto. Normalmente, para un puerto paralelo, puede usar resistencias de 47 kΩ.

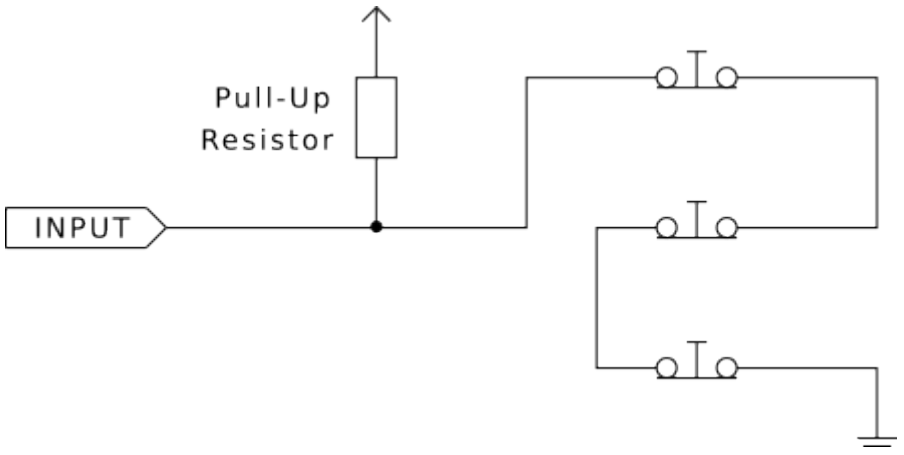


Figura 3.11: Interruptores normalmente cerrados (N/C) cableados en serie (diagrama simplificado)

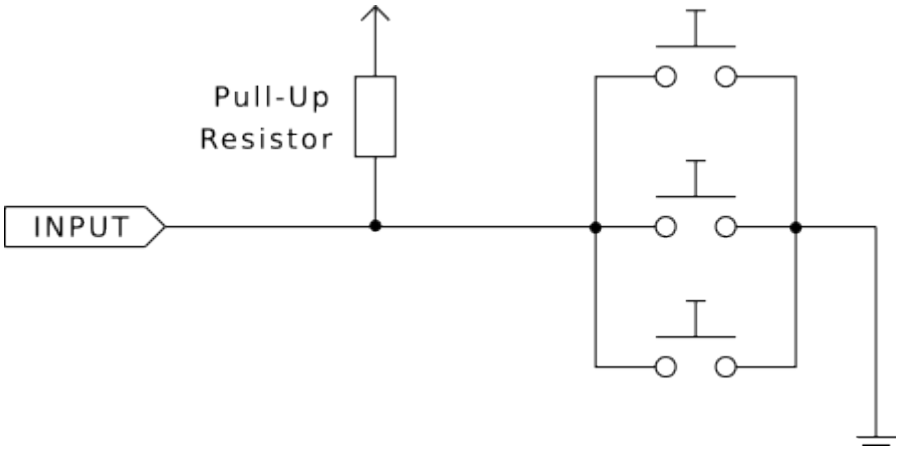


Figura 3.12: Interruptores normalmente abiertos (N/A) cableados en paralelo (diagrama simplificado)

Las siguientes combinaciones de interruptores están permitidas en StepConf:

- Combinar los interruptores de casa para todos los ejes
- Combinar los interruptores de límite para todos los ejes
- Combinar ambos interruptores de límite para un eje
- Combinar ambos interruptores de límite y el interruptor de casa para un eje
- Combinar un interruptor de límite y el interruptor de casa para un eje

Las últimas dos combinaciones también son apropiadas cuando se usa el tipo contacto + casa.

3.2. Asistente de configuración Mesa

PnCconf está hecho para ayudar a construir configuraciones que utilizan productos específicos Mesa *Anything I/O*.

Puede configurar sistemas servo de bucle cerrado o sistemas de hardware paso a paso. Utiliza un enfoque de *asistente* similar a StepConf (utilizado para sistemas accionados por pasos por software, por puerto paralelo).

PnCconf aún se encuentra en una etapa de desarrollo (Beta), por lo que hay algunos errores y carece de algunas características. Por favor informe errores y haga sus sugerencias en la página del foro de LinuxCNC o en la lista de correo.

Hay dos líneas pensamiento cuando se utiliza PnCconf:

Una es usar PnCconf para configurar siempre su sistema; si decide cambiar algunas opciones, vuelva a cargar PnCconf y permítale que configure las nuevas opciones. Esta funciona bien si su máquina es bastante estándar y puede usar archivos personalizados para agregar características no estándar. PnCconf intenta trabajar con usted en este aspecto.

La otra es usar PnCconf para construir una configuración que esté cerca de lo deseado y luego editar a mano para adaptarlo a sus necesidades. Esta sería la elección si necesita modificaciones extensas más allá del alcance de PnCconf o solo quiere jugar con o aprender LinuxCNC.

Navegue por las páginas del asistente con los botones de Adelante, Volver y Cancelar. Hay también un botón de Ayuda que brinda información sobre las páginas, diagramas y una página de salida.

sugerencia
La página de ayuda de PnCconf debe tener la información más actualizada y contiene detalles adicionales.

3.2.1. Instrucciones punto por punto

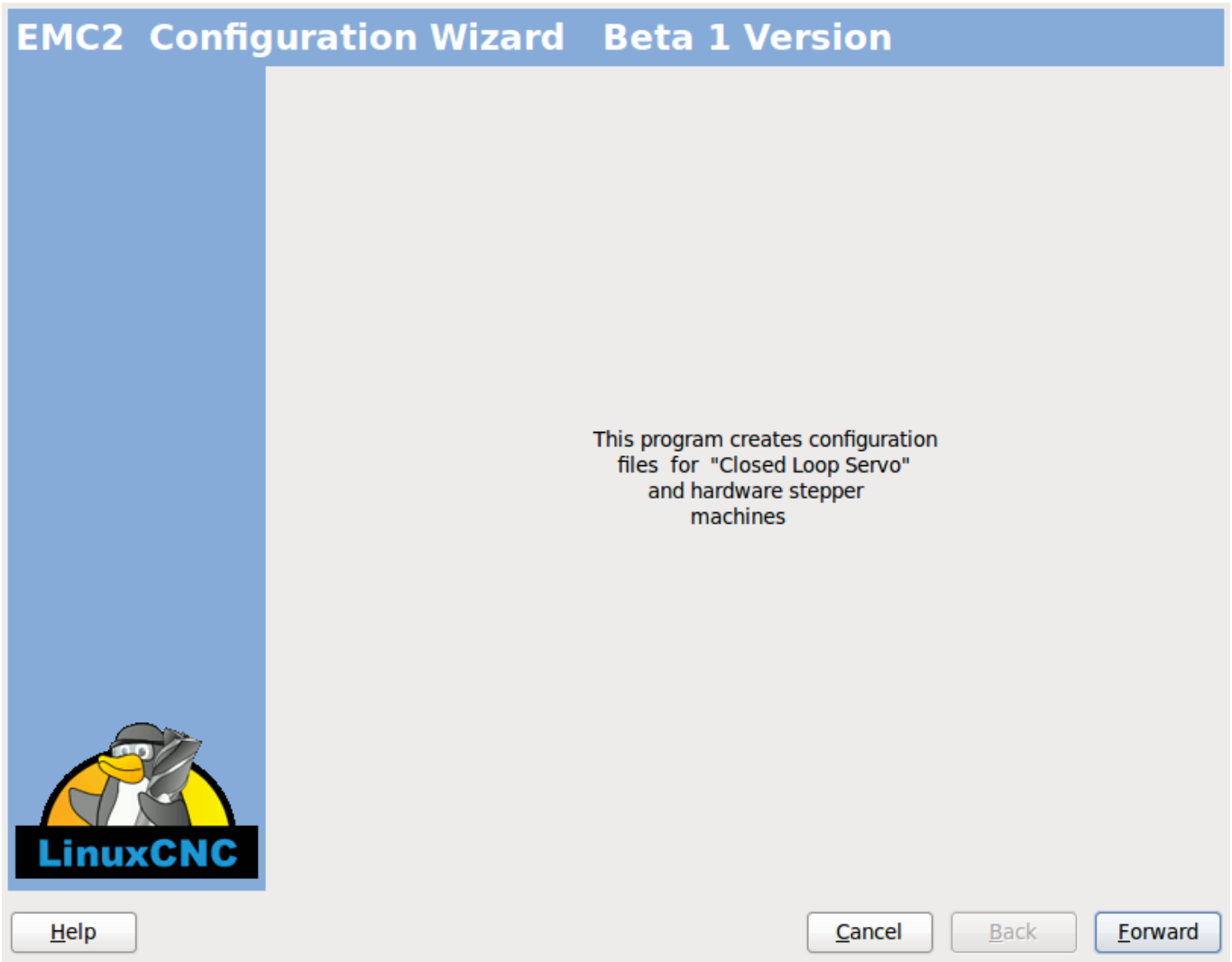


Figura 3.13: Ventana splash PnCConf

3.2.2. Crear o editar

Se permite seleccionar una configuración previamente guardada o crear una nueva. Si elige *Modificar una configuración* y luego presiona *Adelante,* se mostrará una ventana de selección de archivos. PnCconf preselecciona su último archivo guardado. Elija la configuración que usted desea editar. Si realizó algún cambio en los archivos principales HAL o INI **PnCconf sobre-escribirá** esos archivos y tales cambios se perderán. Algunos archivos no serán sobre-escritos y PnCconf colocará una nota en esos archivos. También le permite seleccionar las opciones de atajo de escritorio/lanzador. Un atajo de escritorio colocará un icono de carpeta en el escritorio que apunta a sus nuevos archivos de configuración. De lo contrario, tendrá que buscar en su carpeta home bajo linuxcnc/configs.

Un lanzador de escritorio agregará un icono al escritorio para iniciar directamente con su configuración. También puede iniciarlo desde el menú principal utilizando el Selector de Configuración *LinuxCNC* encontrado en el menú CNC y seleccionando el nombre de su configuración.

3.2.3. Información básica de la máquina

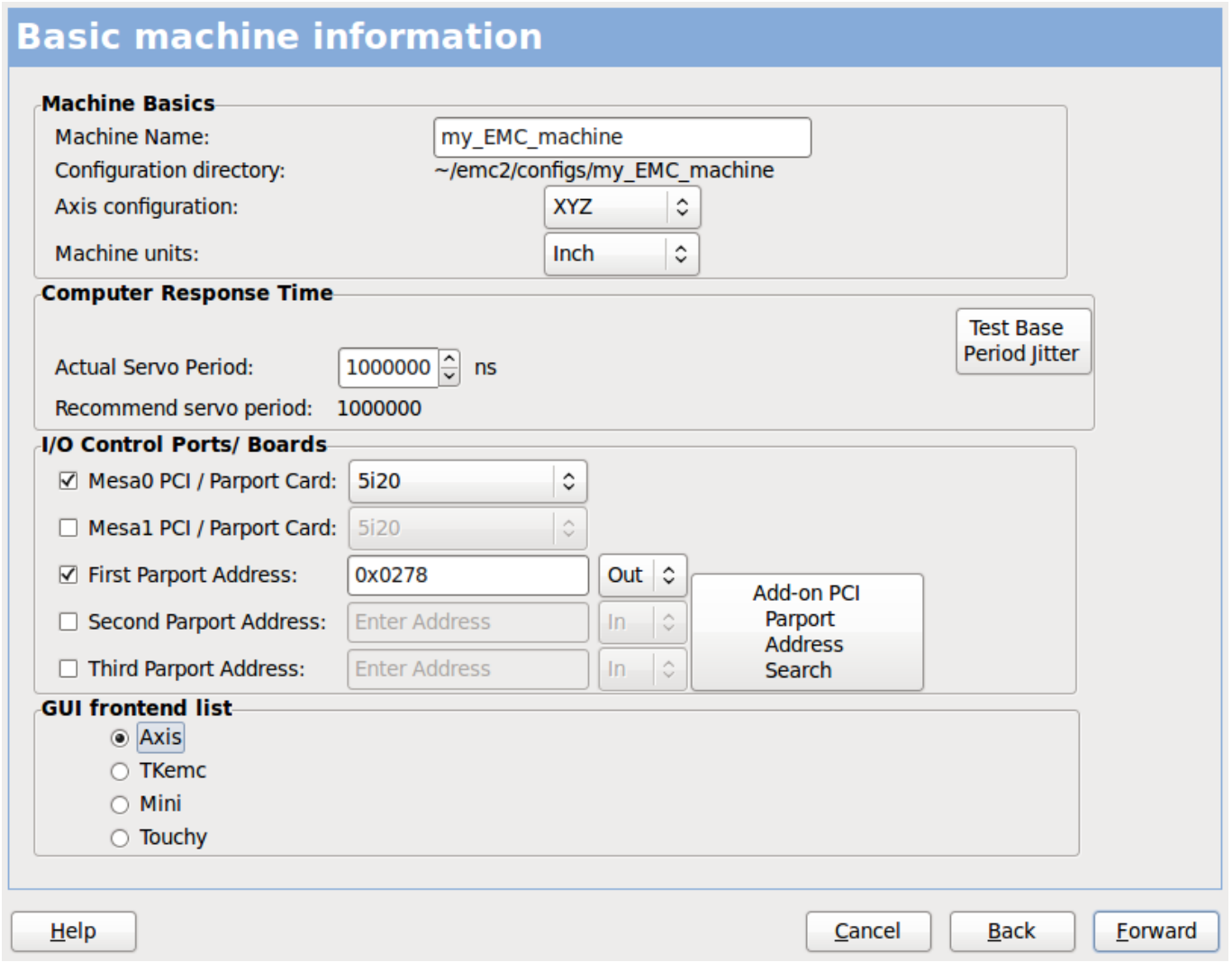


Figura 3.14: PnCConf básico

Máquina
Si usa un nombre con espacios PnCconf reemplazará los espacios con guiones bajos (como regla general, a Linux no le gustan los espacios en los nombres). Elegir una configuración de ejes selecciona qué tipo de máquina está construyendo y qué ejes están disponibles. El selector de "unidades de máquina" permite la entrada de datos en unidades métricas o imperiales en pasos posteriores del proceso de configuración.

---

sugerencia
Los valores predeterminados no se convierten cuando se usa sistema métrico, así que ¡asegúrese de que sean valores razonables!

Tiempo de respuesta de la computadora
: El período servo configura el latido del corazón del sistema. La latencia describe la diferencia de tiempo entre lo programado por el sistema para realizar una acción y el tiempo en que realmente se hace. Como un ferrocarril, LinuxCNC requiere todo en una línea de tiempo muy estrecha y consistente o pasaran cosas malas. LinuxCNC requiere y usa un sistema operativo *en tiempo real*, que solo significa que tiene un tiempo de respuesta de baja latencia (retraso). Cuando LinuxCNC requiere y hace sus cálculos, no puede ser interrumpido por solicitudes de menor prioridad (como la entrada del usuario a botones de pantalla, dibujo, etc).

Probar la latencia es crucial y algo clave a revisar antes de seguir adelante. Por favor siga las instrucciones en la página Prueba de latencia antes de seguir adelante.

Una vez conformes con la latencia se debe elegir un período de servo. En la mayoría de los casos, un período servo de 1000000 ns esta bien (eso da una tasa de cálculo del servo de 1 kHz - 1000 cálculos por segundo). Si está construyendo un servosistema de lazo cerrado que controla el par (corriente) en lugar de velocidad (voltaje), una tasa más rápida sería mejor; algo así como 200000 (velocidad de cálculo de 5 kHz). El problema con baja la velocidad del servo es que deja menos tiempo disponible para que la computadora haga otras cosas además de cálculos de LinuxCNC. Normalmente, la pantalla (GUI) se vuelve menos responsiva. Usted debe decidir sobre el equilibrio. Tenga en cuenta que si afina su servosistema de lazo cerrado y luego cambia el período del servo, probablemente necesite volver a afinarlo.

Puertos/tarjetas de control de E/S
: PnCconf es capaz de configurar máquinas que tienen hasta dos placas Mesa y tres puertos paralelos. Los puertos paralelos solo se pueden usar para E/S de baja velocidad (frecuencia del servo).

Mesa
: Debe elegir al menos una placa Mesa ya que PnCconf no configurará los puertos paralelos para contar codificadores o pasos de salida o señales PWM. Las tarjetas Mesa disponibles en el cuadro de selección se basan en lo que PnCconf encuentra en el firmware del sistema. Hay opciones para agregar firmware personalizado y/o *lista negra* (ignorar) algún firmware o tarjetas usando un archivo de preferencias. Si no se encuentra firmware, PnCconf mostrará una advertencia y usará un firmware interno de muestra - no se podrán realizar pruebas. Un punto a tener en cuenta es que si se eligen dos tarjetas PCI Mesa, actualmente no hay manera de predecir qué tarjeta es 0 y cual es 1 - se debe probar - mover las tarjetas podría cambiar su orden. Si se configuran dos tarjetas, ambas deben instalarse para que las pruebas funcionen.

Puerto paralelo
: Se pueden usar hasta 3 puertos paralelos (denominados parports) como E/S simple. Usted debe establecer la dirección del parport. Puede ingresar el puerto paralelo de Linux según su sistema de numeración de puertos (0, 1 o 2) o ingresar la dirección real. La dirección de un parport en placa base es a menudo 0x0278 o 0x0378 (escrito en hexadecimal) pero puede ser encontrado en la página de BIOS. La página del BIOS se encuentra cuando usted arranca su computadora, debe presionar una tecla para entrar en ella (como F2). En la página BIOS puede encontrar la dirección del puerto paralelo y configurarlo en el modo SPP, EPP, etc. en algunas computadoras esta información se muestra durante unos segundos durante el arranque. Para tarjetas de puerto paralelo PCI, la dirección se puede encontrar presionando el boton *buscar dirección de parport*. Aparece la página de salida de ayuda con una lista de todos los dispositivos PCI que se pueden encontrar. Debe haber una referencia a un dispositivos puerto paralelo con una lista de direcciones. Una de esas direcciones debería funcionar. No todos los puertos paralelos PCI funcionan correctamente. Cualquiera de los tipos se puede seleccionar como *in* (máximo cantidad de pines de entrada) o *out* (cantidad máxima de pines de salida).

Lista de interfaces de usuario
: Esto especifica las pantallas gráficas que usará LinuxCNC. Cada una tiene unas opciones diferentes.

AXIS

- totalmente compatible con tornos.
- es el front-end más desarrollado y utilizado
- está diseñado para ser utilizado con mouse y teclado
- está basado en tkinter, por lo que integra PyVCP (paneles de control virtual basados en Python) de forma natural.
- tiene una ventana gráfica 3D.
- permite VCP integrado en el lateral o en la pestaña central

TkLinuxCNC

- pantalla azul de alto contraste
- ventana de gráficos separada
- Sin integración VCP

Touchy

- Touchy fue diseñado para ser utilizado con una pantalla táctil, con los mínimos interruptores físicos y un volante MPG.
- Requiere botones para ciclo de inicio, aborto y señales de un solo paso
- También requiere que se seleccione trote MPG de eje compartido.
- está basado en GTK por lo que integra GladeVCP (paneles de control virtual) de forma natural.
- permite paneles VCP integrados en la pestaña central
- no tiene ventana gráfica
- se puede cambiar el aspecto con temas personalizados

QtPlasmaC

- todas las funciones de configuración de plasmac con base en la infraestructura QtVCP.
- operación por mouse/teclado o pantalla táctil
- Sin integración VCP

3.2.4. Configuración de externos

Esta página le permite seleccionar controles externos como jogging o mando manual de velocidades.

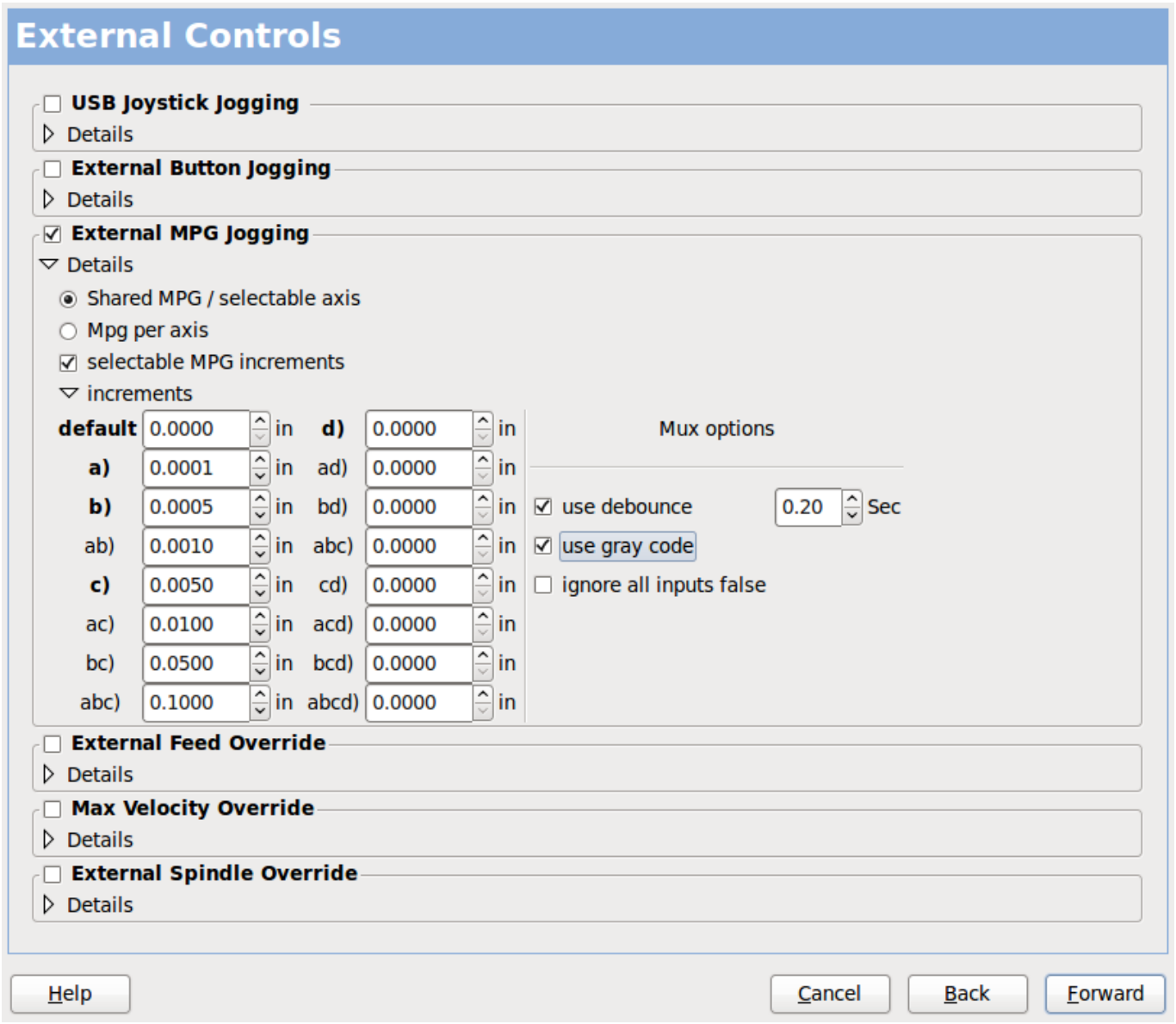


Figura 3.15: Controles externos

Si selecciona palanca de mando para trote, necesitará que siempre este conectada para que LinuxCNC la cargue. Para usar palancas analógicas en un trote usable probablemente necesite agregar algún código HAL personalizado. El jogging MPG requiere un generador de impulsos conectado a un codificador contador de MESA. Los controles de mando manual pueden usar un generador de impulsos (MPG) o interruptores (como un dial giratorio). Los botones externos se pueden usar con una palanca de mando OEM basada en interruptores.

Trote con palanca de mando

Requiere instalar una *regla de dispositivo* personalizada en el sistema. Este es un archivo que LinuxCNC usa para conectarse a la lista de dispositivos de Linux. PnCconf le ayudará a hacer ese archivo.

- *Buscar regla de dispositivo* buscará reglas en el sistema, puede usar esto para encontrar el nombre de los dispositivos que ya ha construido con PnCconf.

- *Agregar una regla de dispositivo* le permitirá configurar un nuevo dispositivo siguiendo las indicaciones. Necesitará que su dispositivo este disponible.
- *prueba de dispositivo* le permite cargar un dispositivo, ver los nombres de sus pines y verificar sus funciones con halmeter.

El trote con palanca de mando usa componentes HALUI y hal_input.

Botones externos
: permite trotar el eje con botones simples a una velocidad de trote específica. Probablemente lo mejor para trote rápido.

Trote MPG
: Le permite usar un generador manual de impulsos para mover ejes de la máquina.

Los MPG a menudo se encuentran en máquinas de grado comercial. Emiten pulsos en cuadratura que se pueden contar con un codificador contador MESA. PnCconf permite un MPG por eje o un MPG compartido con todos los ejes. Permite la selección de velocidades de trote usando interruptores o una sola velocidad.

La opción de incrementos seleccionables usa el componente mux16. Este componente tiene opciones como antirrebote y código gris para ayudar a filtrar la entrada del interruptor en bruto.

Ajuste manual
: PnCconf permite mando manual de velocidades de avance y/o velocidad del husillo con un generador de pulsos (MPG) o interruptores (por ejemplo, rotativos).

3.2.5. Configuración de la GUI

Aquí puede establecer valores predeterminados para las pantallas de visualización, agregar paneles de control virtual (VCP), y establecer algunas opciones de LinuxCNC ..

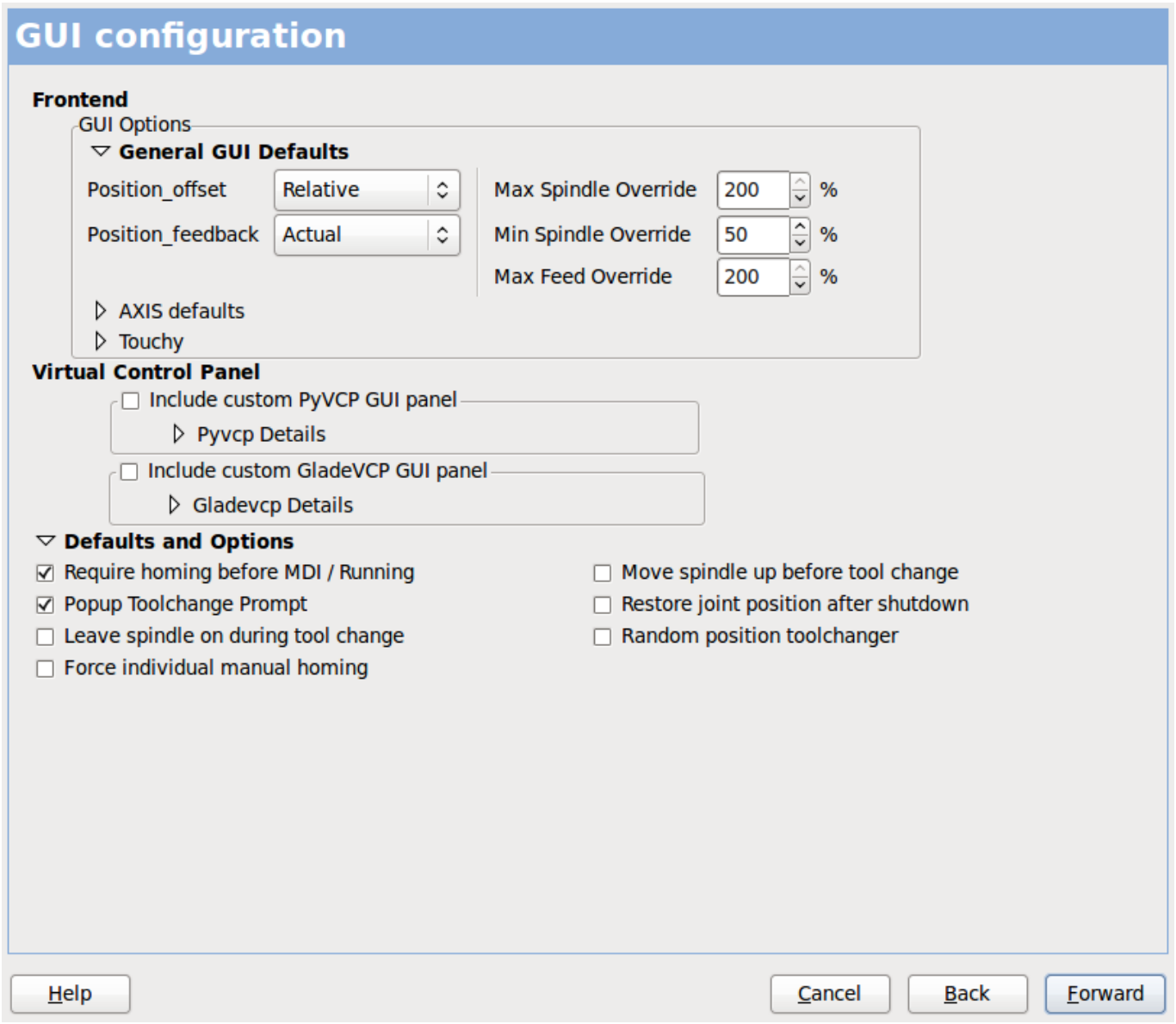


Figura 3.16: Configuración de la GUI

Opciones de GUI

Las opciones predeterminadas permiten elegir los valores predeterminados generales para cualquier pantalla de visualización.

Los valores predeterminados de AXIS son opciones específicas de AXIS. Si elige las opciones de tamaño, posición o forzar maximizar, PnCconf le preguntará si es correcto sobrescribir el archivo de preferencias (.axisrc). A menos que haya agregado comandos manualmente a este archivo, sera correcto permitirlo. La posición y forzar maximizar se pueden usar para mover AXIS a un segundo monitor si el sistema es capaz.

Los valores predeterminados de Touchy son opciones específicas de Touchy. La mayoría de las opciones de Touchy pueden ser cambiadas mientras Touchy se está ejecutando usando la página de preferencias. Touchy usa GTK para dibujar su pantalla, el cual soporta temas. Los temas controlan el aspecto y sensación básicos de un programa. Puede descargar temas de la red o editarlos usted mismo. Hay una lista de los temas actuales en la computadora entre los que puede elegir. Para ayudar

a resaltar alguna parte del texto, PnCconf le permite anular los valores predeterminados de los temas. Las opciones de posición y forzar maximizar se pueden usar para mover Touchy a un segundo monitor si el sistema es capaz.

Las opciones de QtPlasmaC son específicamente para QtPlasmaC, cualquier opción común que no sea requerida estará deshabilitada. Si se selecciona QtPlasmaC entonces la pantalla siguiente del asistente será para configurar los botones de usuario específicos para QtPlasmaC y las opciones de VCP no estarán disponibles.

Opciones de VCP
: Los paneles de control virtuales permiten agregar controles y visualizaciones a la pantalla. AXIS y Touchy pueden integrar estos controles dentro de la pantalla en posiciones designadas. Hay dos tipos de VCP: PyVCP que usa *Tkinter* para dibujar la pantalla y GladeVCP que usa *GTK* para dibuja la pantalla.

PyVCP
: El archivo XML de las pantallas PyVCP solo se puede construir a mano. PyVCPs encajan naturalmente con AXIS ya que ambos usan TKinter.

Los pines HAL se crean para que el usuario los conecte dentro de su archivo HAL personalizado. Hay un panel de visualización de husillo de ejemplo para que el usuario lo use tal como está o lo use como base. Puede seleccionar un archivo en blanco en el que luego puede agregar los controles *widgets* o seleccionar un ejemplo de visualización del husillo que mostrará su velocidad e indicará si está a la velocidad requerida.

PnCconf conectará los pines HAL de visualización del husillo adecuados para usted. Si utiliza AXIS el panel se integrará en el lado derecho. Si no utiliza AXIS el panel estará separado *en solitario* de la pantalla.

Puede usar las opciones de geometría para ajustar el tamaño y mover el panel, por ejemplo para moverlo a una segunda pantalla si el sistema es capaz. Si presiona el botón *Ver muestra de panel*, se respetarán las opciones de tamaño y posición.

GladeVCP
: GladeVCP encaja naturalmente dentro de la pantalla Touchy ya que ambos usan GTK para dibujarse, pero se puede cambiar el tema de GladeVCP para combinarse bastante bien con AXIS (pruebe Redmond).

Usa un editor gráfico para construir sus archivos XML. Los pines HAL se crean para que el usuario los conecte dentro de su archivo HAL personalizado.

GladeVCP también permite una interacción de programación mucho más sofisticada (y complicada) que PnCconf actualmente no aprovecha (ver GladeVCP en el manual).

PnCconf tiene paneles de muestra para que el usuario los utilice tal como están o los tome como base. Con GladeVCP, PnCconf le permitirá seleccionar diferentes opciones para su panel de muestra.

En *Opciones de muestra* seleccione cuáles le gustaría. Los botones de puesta cero usan comandos HALUI que puede editar más adelante en la sección HALUI.

Auto Z touch-off también requiere el programa touch-off de classic ladder y una entrada de sonda seleccionada. Requiere una placa conductora de toque y una herramienta conductora puesta a tierra. Para obtener una idea de cómo funciona, consulte:

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

En *Opciones de visualización*, el tamaño, la posición y forzar máximizar se pueden usar en un panel *autónomo* para cosas tales como colocar la pantalla en un segundo monitor si el sistema es capaz.

Puede seleccionar un tema GTK que establezca la apariencia básica del panel. Por lo general, deseara que esto coincida con la pantalla de la interfaz. Estas opciones se usarán si presiona el botón *Ver*

muestra de panel. Con GladeVCP dependiendo de la pantalla frontal, puede seleccionar dónde se mostrará el panel.

Puede forzarlo a que sea independiente, o con AXIS puede estar en el centro o en el lado derecho. Con Touchy puede estar en el centro.

Opciones y valores predeterminados

- Requiere homing antes de MDI/Running
  - Si desea poder mover la máquina antes del homing, desmarque esta casilla de verificación.
- Indicación emergente de herramienta
  - Para cambios de herramienta, elija entre un aviso en pantalla o exportación de nombres de señal estándar para un archivo HAL de cambio de herramientas personalizado proporcionado por el usuario
- Dejar el husillo encendido durante el cambio de herramienta:
  - Utilizado para tornos
- Forzar homing manual individual
- Mover el husillo hacia arriba antes de cambiar la herramienta
- Restaurar la posición de la articulación después de apagar
  - Utilizado para máquinas con cinemáticas no triviales
- Cambiadores de herramienta de posición aleatoria
  - Se usa para cambiadores de herramientas que no devuelven la herramienta a la misma ranura. Necesitará agregar código HAL personalizado para admitir cambiadores de herramientas.

3.2.6. Configuración de tarjetas Mesa

Las páginas de configuración de Mesa le permiten utilizar diferentes firmwares. Si en la página básica seleccionó una tarjeta Mesa, aquí elija el firmware disponible y seleccione qué y cuántos componentes están disponibles.

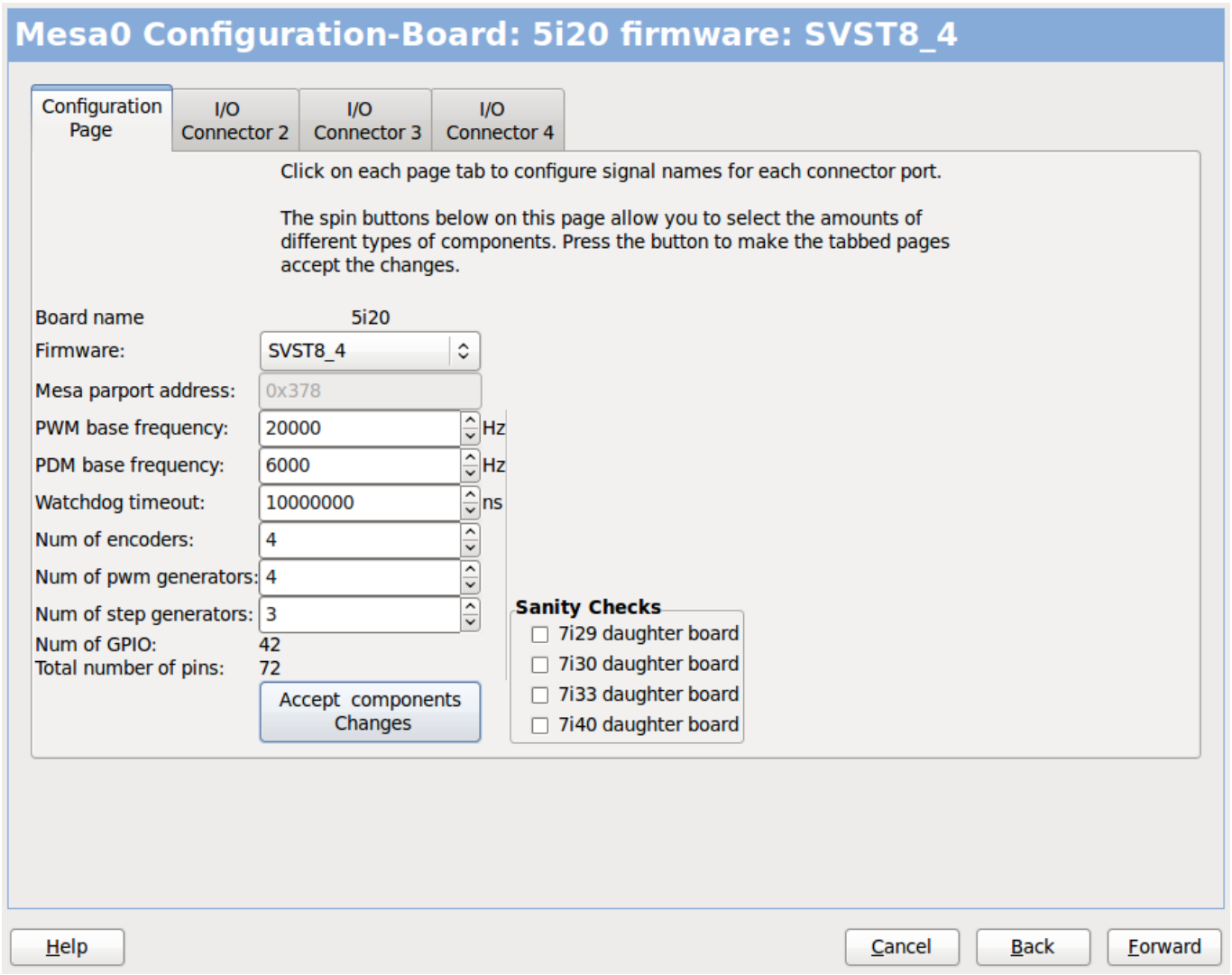


Figura 3.17: Configuración de placa Mesa

La dirección de parport solo se usa con la tarjeta parport de Mesa, la 7i43. Los puertos paralelos en placa base generalmente usan 0x278 o 0x378, aunque debería poder encontrar la dirección desde la página del BIOS. La 7i43 requiere que el puerto paralelo use el modo EPP, también establecido en la página del BIOS. Si usa un puerto paralelo PCI, se puede ubicar la dirección con el botón de búsqueda en la página básica.

nota
Muchas tarjetas PCI no soportan apropiadamente el protocolo EPP.

La frecuencia base de PDM PWM y 3PWM establece el equilibrio entre rizado y linealidad. Si usa tarjetas hija Mesa, los documentos para la placa deben dar recomendaciones.

importante
Es importante seguir esto para evitar daños y obtener el mejor rendimiento.

```
La 7i33 requiere PDM y una frecuencia base PDM de 6 MHz
La 7i29 requiere PWM y una frecuencia base PWM de 20 kHz
La 7i30 requiere PWM y una frecuencia base PWM de 20 kHz
La 7i40 requiere PWM y una frecuencia base PWM de 50 kHz
La 7i48 requiere UDM y una frecuencia base PWM de 24 kHz
```

Tiempo de espera de Watchdog
: se usa para establecer cuánto tiempo esperará la placa MESA antes de matar las salidas si la comunicación se interrumpe desde la computadora. Por favor, recuerde que Mesa usa salidas *activas bajas* lo que significa que cuando el pin de salida está activado, será bajo (aproximadamente 0 voltios) y si la salida es alta (aproximadamente 5 voltios), está apagado. Asegúrese de que su equipo es seguro cuando esté en el estado apagado (watchdog activado).

Número de codificadores/generadores PWM/generadores de pasos
: Puede elegir el número de componentes disponibles anulando la selección de los no utilizados. No todos los tipos de componentes están disponibles con todos los firmware.

Elegir por debajo de la cantidad máxima de componentes permite ganar más pines GPIO. Si usa tarjetas hijas, tenga en cuenta que no debe deseleccionar los pines que usa la tarjeta. Por ejemplo, algunos firmware admiten dos tarjetas 7i33; si solo tiene una puede anular la selección de suficientes componentes para utilizar el conector que admite la segunda 7i33. Los componentes son deseleccionados numéricamente con el número más alto primero hacia abajo sin saltar números. Si haciendo esto los componentes no están donde los quiere, entonces debe usar un firmware diferente. El firmware dicta dónde, qué y las cantidades máximas de los componentes. Es posible un firmware personalizado, pregunte amablemente al contactar a los desarrolladores de LinuxCNC y Mesa. Usar firmware personalizado en PnCconf requiere procedimientos especiales y no siempre es posible, aunque se intenta hacer que PnCconf sea lo más flexible posible.

Después de elegir todas estas opciones, presione el botón *Aceptar cambios a los componentes* y PnCconf actualizará las páginas de configuración de E/S. Solo se mostrarán las pestañas de E/S para los conectores disponibles, dependiendo de la placa Mesa.

3.2.7. Configuración de E/S Mesa

Las pestañas se utilizan para configurar los pines de entrada y salida de las placas Mesa. PnCconf le permite crear nombres de señal personalizados para usar en archivos HAL personalizados.

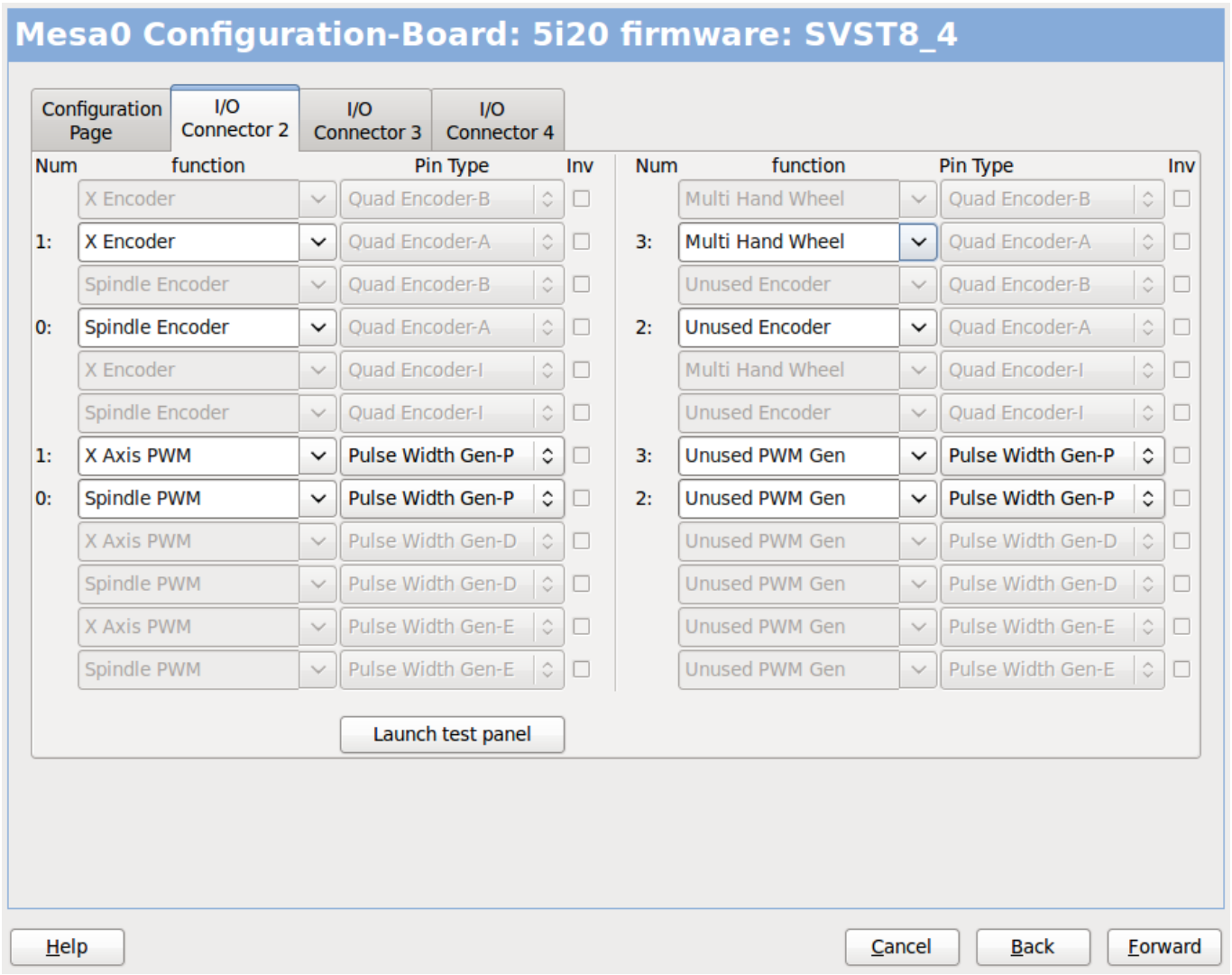


Figura 3.18: Configuración del conector E/S Mesa 2

En esta pestaña con este firmware, los componentes están configurados para una tarjeta hija 7i33, generalmente utilizada con servos de lazo cerrado. Tenga en cuenta que los números de componente de los codificadores contadores y los controladores PWM no están en orden numérico. Siguen los requisitos de la tarjeta hija.

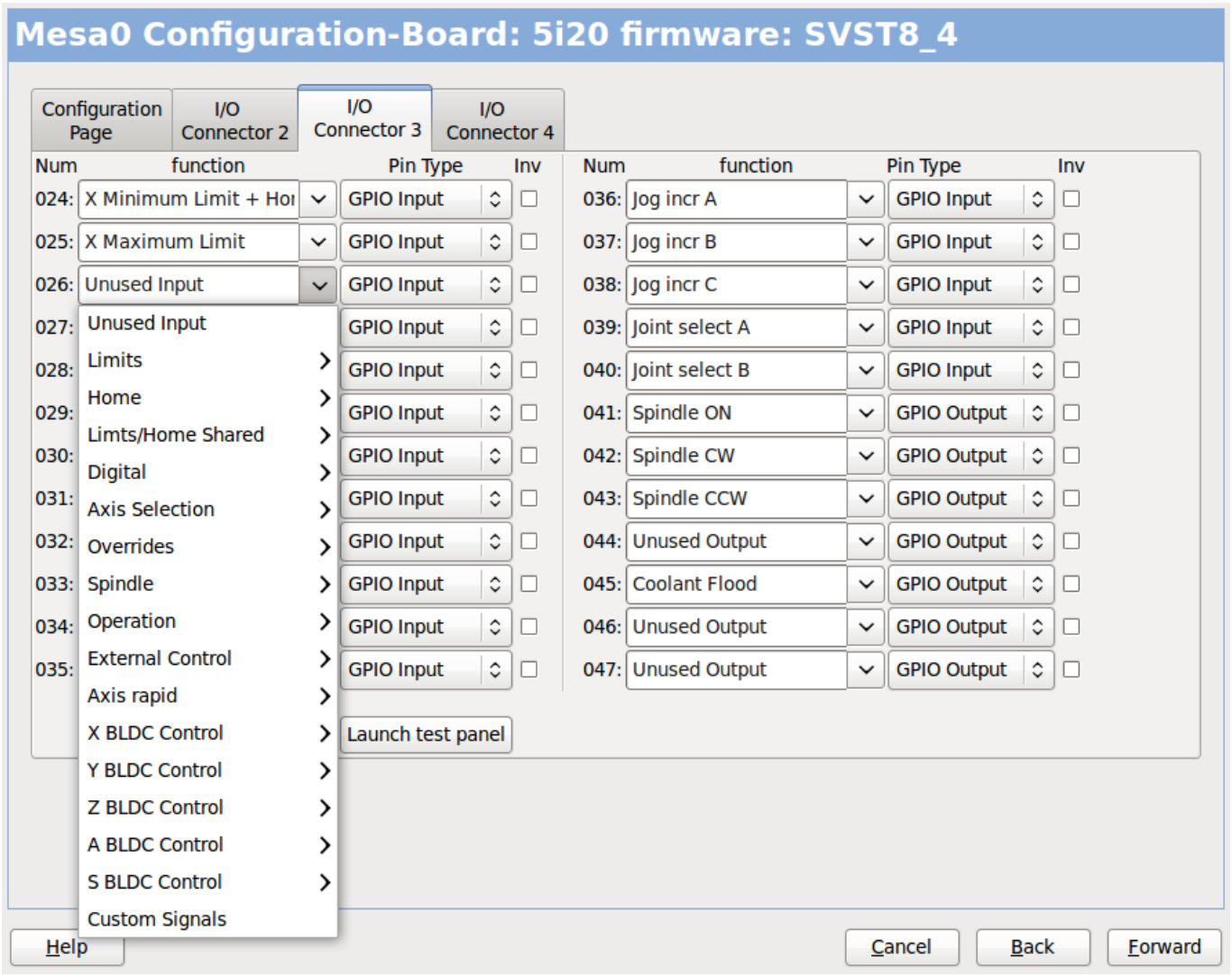


Figura 3.19: Configuración del conector E/S Mesa 3

En esta pestaña, todos los pines son GPIO. Tenga en cuenta los números de 3 dígitos - coincidirán con el número de pin HAL. Los pines GPIO se pueden seleccionar como entrada o salida y se pueden invertir.

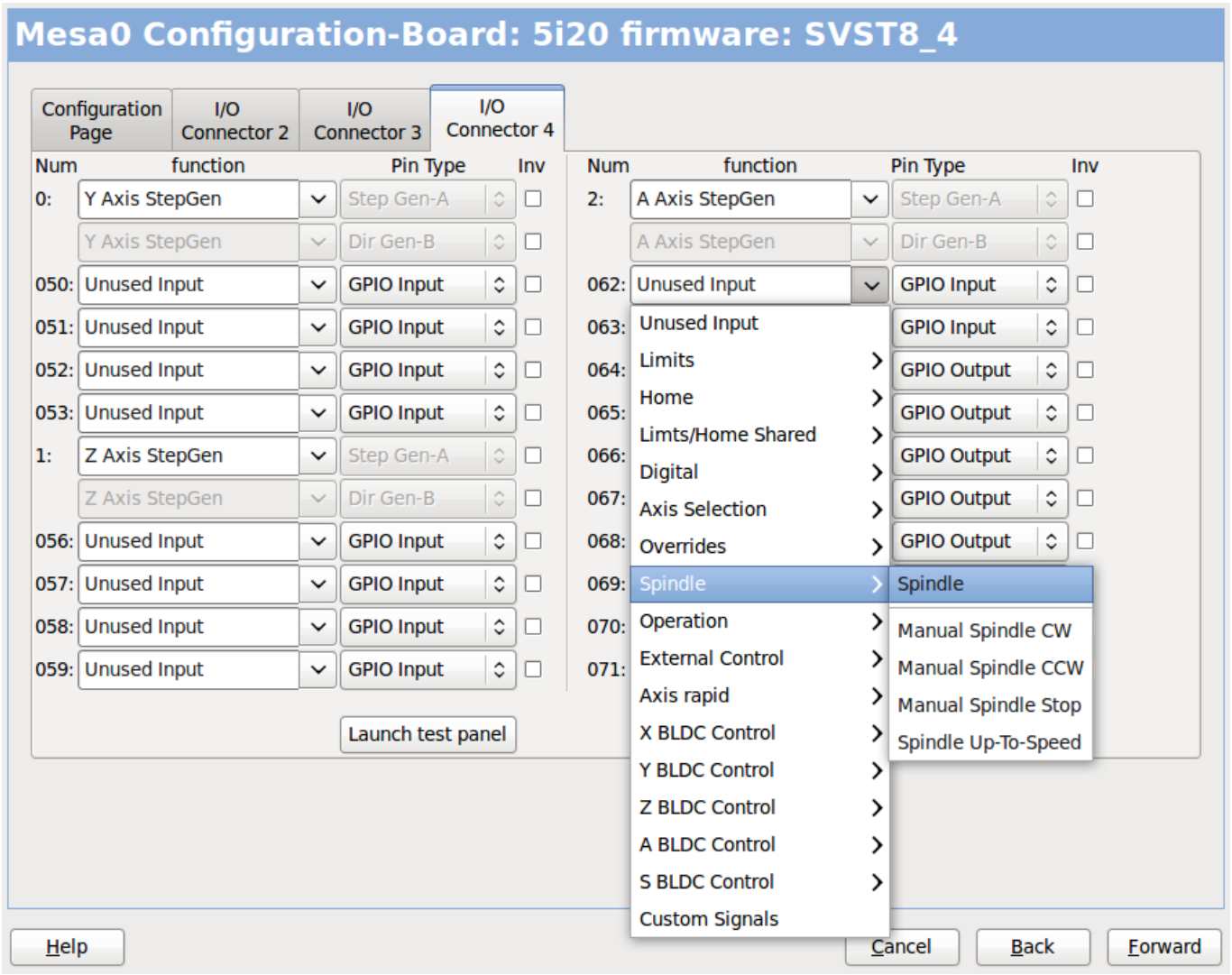
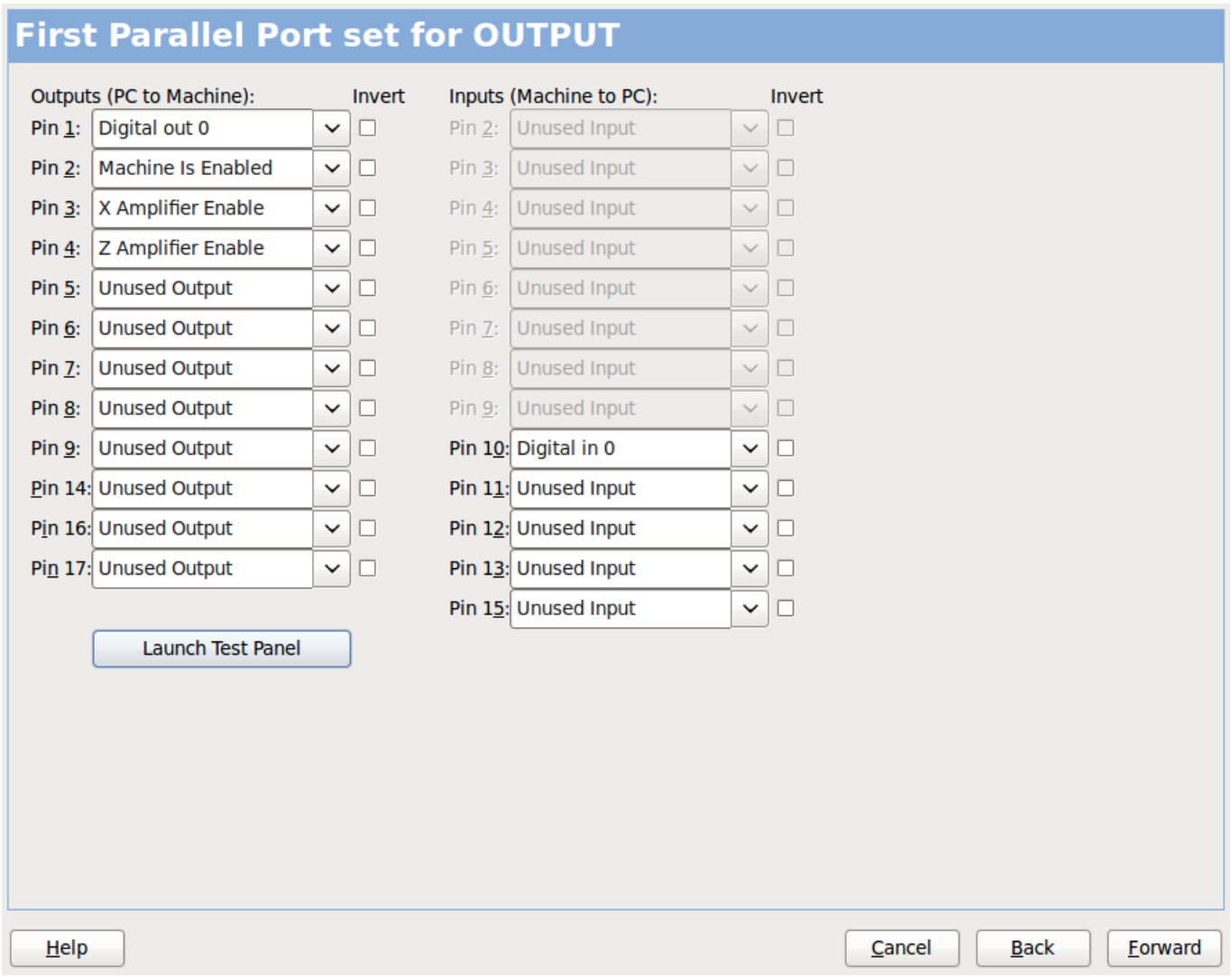


Figura 3.20: Configuración del conector E/S Mesa 4

En esta pestaña hay una mezcla de generadores de pasos y GPIO. Los generadores de pasos y los pines de dirección se pueden invertir. Tenga en cuenta que invertir un pin Step Gen-A (el pin de salida de paso) cambia el tiempo del paso. Debería coincidir con lo que espera su controlador.

3.2.8. Configuración de puerto paralelo



El puerto paralelo se puede usar para E/S simple, similar a los pines GPIO de Mesa.

3.2.9. Configuración de ejes

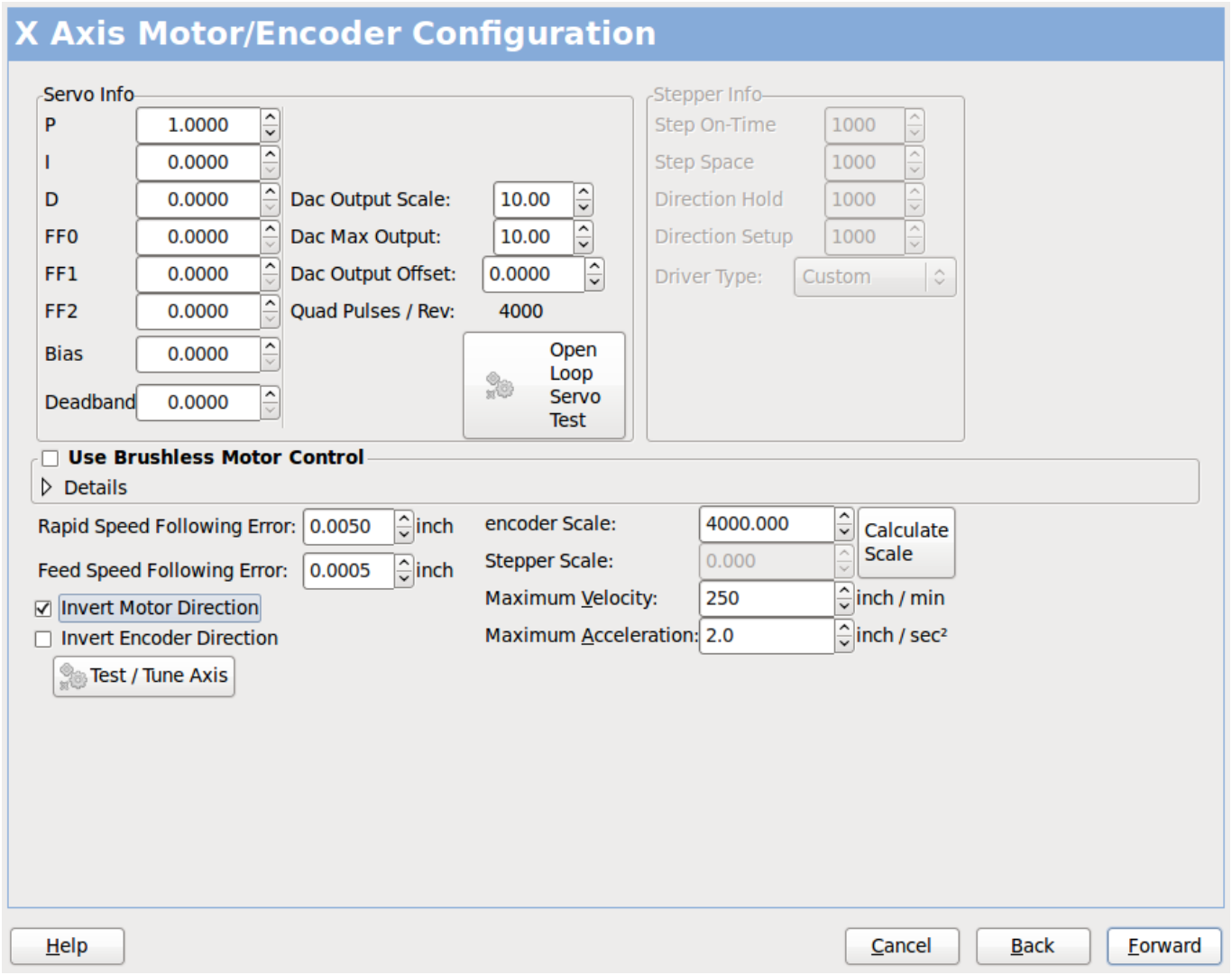


Figura 3.21: Configuración del controlador del eje

Esta página permite configurar y probar la combinación de motor y/o codificador. Si usa un servomotor, hay disponible una prueba de lazo abierto. Si usa un paso a paso, hay disponible una prueba de afinación.

Prueba de lazo abierto

La prueba de lazo abierto es importante ya que confirma la dirección del motor y codificador. El motor debe mover el eje en la dirección positiva cuando se presiona el botón positivo y también el codificador debe contar en positivo. El movimiento del eje debe seguir el estándar del Manual de Maquinaria [1] o la pantalla gráfica AXIS no tendrá mucho sentido. Esperemos que la página de ayuda y los diagramas le ayudan a resolver esto. Tenga en cuenta que las direcciones de los ejes se basan en el movimiento de la HERRAMIENTA, y no en el movimiento de la mesa. No hay rampa de aceleración con la prueba de lazo abierto, así que comience con números DAC bajos. Al mover el eje una distancia conocida, se puede confirmar la escala del codificador. El codificador

[1] "nomenclatura de los ejes" en el capítulo "Control Numérico" en "Machinery's Handbook" publicado por Industrial Press.

debe contar incluso sin el amplificador habilitado dependiendo de cómo se suministra energía al mismo.



aviso
Si el motor y el codificador no están de acuerdo con la dirección de conteo, entonces el servo se descontrolara cuando use el control PID.

Dado que en este momento no se puede probar la configuración PID en PnCconf, la configuración es realmente para cuando reedite una configuración - ingrese su configuración PID probada.

Escalado DAC
Se utilizan escala de salida, salida máxima y offset de salida para adaptar la salida DAC.

Calcular DAC
Estos dos valores son los factores de escala y compensación para la salida del eje al amplificador de motor. El segundo valor (compensación) se resta del cálculo de salida (en voltios), y dividido por el primer valor (factor de escala), antes de ser escrito a los convertidores D/A. Las unidades en el valor de la escala están en voltios verdaderos por voltios de salida DAC. Las unidades en el valor de compensación están en voltios. Estos pueden ser utilizado para linearizar un DAC.

Específicamente, al escribir salidas, LinuxCNC primero convierte la salida deseada en unidades cuasi-SI a valores de actuador brutos, por ejemplo, voltios para un amplificador DAC. Esta escala se ve así: El valor de la escala se puede obtener analíticamente haciendo un análisis de unidad, es decir, las unidades son [unidades SI de salida]/[unidades de actuador]. Por ejemplo, en una máquina con un amplificador de modo de velocidad tal que 1 voltio da como resultado una velocidad de 250 mm/seg. Tenga en cuenta que las unidades del desplazamiento están en unidades de máquina, por ejemplo, mm/seg, y se restan de las lecturas del sensor. El valor de este desplazamiento se obtiene al encontrar el valor de su salida que da 0.0 para la salida del actuador. Si el DAC está linealizado, este desplazamiento es normalmente 0.0.

La escala y el offset pueden usarse para linealizar el DAC, lo que da como resultado valores que reflejan los efectos combinados de la ganancia del amplificador, la no linealidad del DAC, unidades DAC, etc. Para ello, siga este procedimiento:

- Construya una tabla de calibración para la salida, indicando al DAC el voltaje deseado y midiendo el resultado:

Cuadro 3.2: Mediciones del voltaje de salida

Bruto	Medido
-10	**-9.93**
-9	**-8.83**
0	**-0.96**
1	**-0.03**
9	**9.87**
10	**10.07**

- Haz un ajuste lineal de mínimos cuadrados para obtener los coeficientes a, b de modo que meas=a*raw+b
- Tenga en cuenta que queremos una salida en bruto de modo que nuestro resultado medido sea idéntico a la salida ordenada. Esto significa

- cmd=a*raw+b
- bruto=(cmd-b)/a

- Como resultado, los coeficientes a y b del ajuste lineal pueden ser utilizados como la escala y el offset para el controlador directamente.

SALIDA MAXIMA
El valor máximo para la salida de la compensación PID que se escribe en el amplificador del motor, en voltios. El valor de salida calculado se fija a este límite. El límite se aplica antes de escalar a unidades de salida en bruto. El valor se aplica simétricamente tanto para el lado positivo como para el negativo.

Prueba de afinación
La prueba de afinación desafortunadamente solo funciona con sistemas basados en pasos. Otra vez confirme que las direcciones en el eje son correctas. Luego pruebe el sistema moviendo el eje de ida y vuelta; si la aceleración o la velocidad máxima son demasiado altas se perderán pasos. Al trotar, tenga en cuenta que puede tomar un tiempo para que un eje con baja aceleración se detenga. Los interruptores de límite no funcionan durante esta prueba. Puede establecer un tiempo de pausa para cada final del movimiento de prueba. Esto le permitiría configurar y leer un indicador de cuadrante para ver si está perdiendo pasos.

Tiempos de motores a pasos
Se necesitan ajustar los tiempos de los motores a paso para adaptarse a los requisitos del controlador de pasos. PnCconf proporciona algunos tiempos predeterminados del controlador o permite configuraciones de tiempo personalizadas. Consulte http://wiki.linuxcnc.org/cgi-bin/-wiki.pl?Stepper_Drive_Timing para algunos números de tiempo más conocidos (siéntase libre de agregar los que haya descubierto). Si tiene duda, utilice números grandes como 5000, esto solo limitará la velocidad máxima.

Control de motor sin escobillas
Estas opciones se utilizan para permitir un control de bajo nivel de motores sin escobillas utilizando firmwares especiales y placas hijas. También permite la conversión de sensores HALL de un fabricante a otro. Solo está parcialmente soportado y requerirá que uno para termine las conexiones HAL. Póngase en contacto con la lista de correo o el foro para más ayuda.

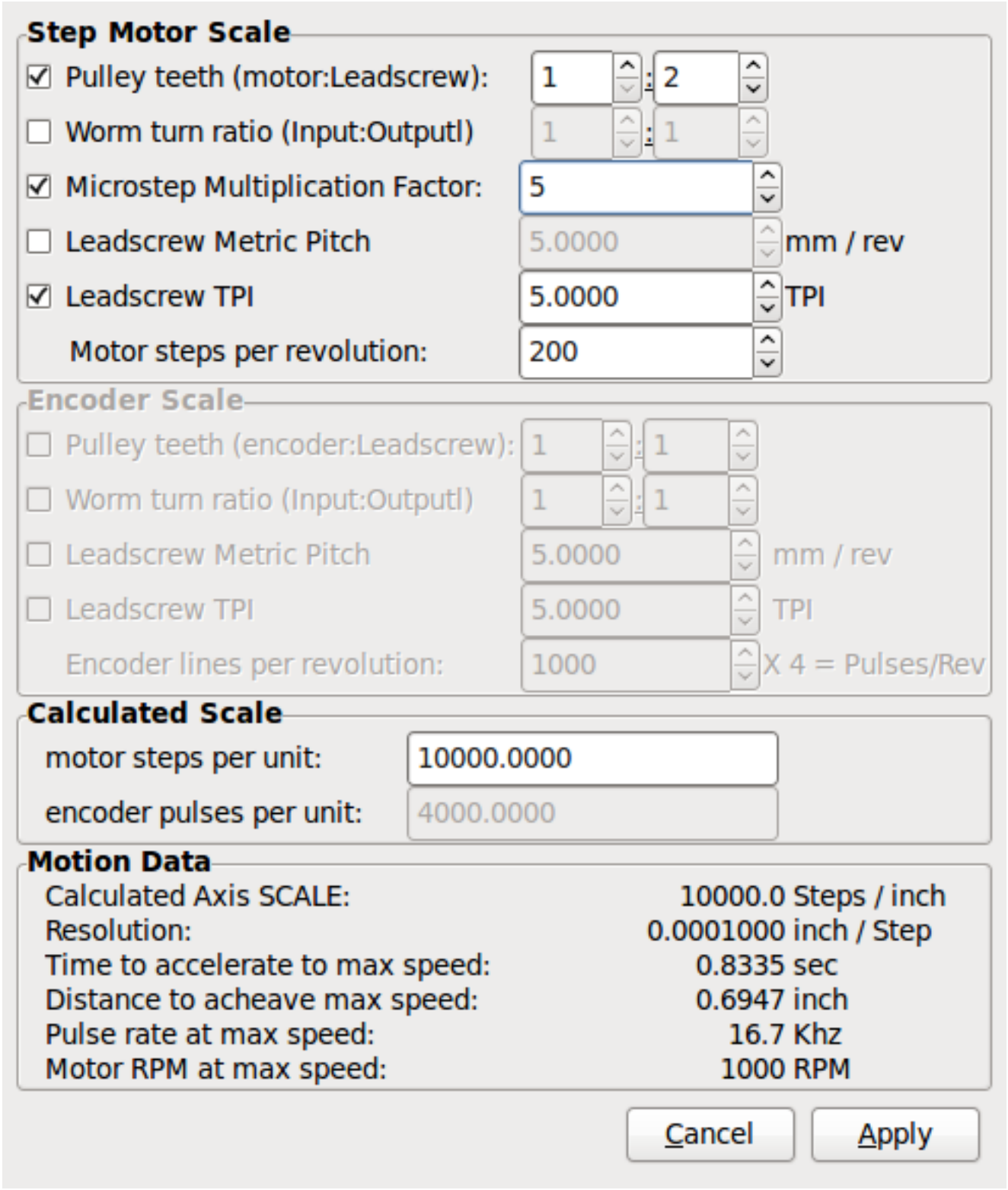


Figura 3.22: Cálculo de la escala del eje

La configuración de la escala se puede ingresar directamente o se puede apoyar con el botón *Calcular escala*. Usa las casillas de verificación para seleccionar los cálculos apropiados. Nota que los *dientes de polea* requieren el número de dientes y no la relación de engranaje. La relación de giro del sinfín es justo lo contrario que requiere la relación de transmisión. Si estas feliz con el escala presiona Aplicar,

de lo contrario presiona Cancelar e ingresa la escala directamente.

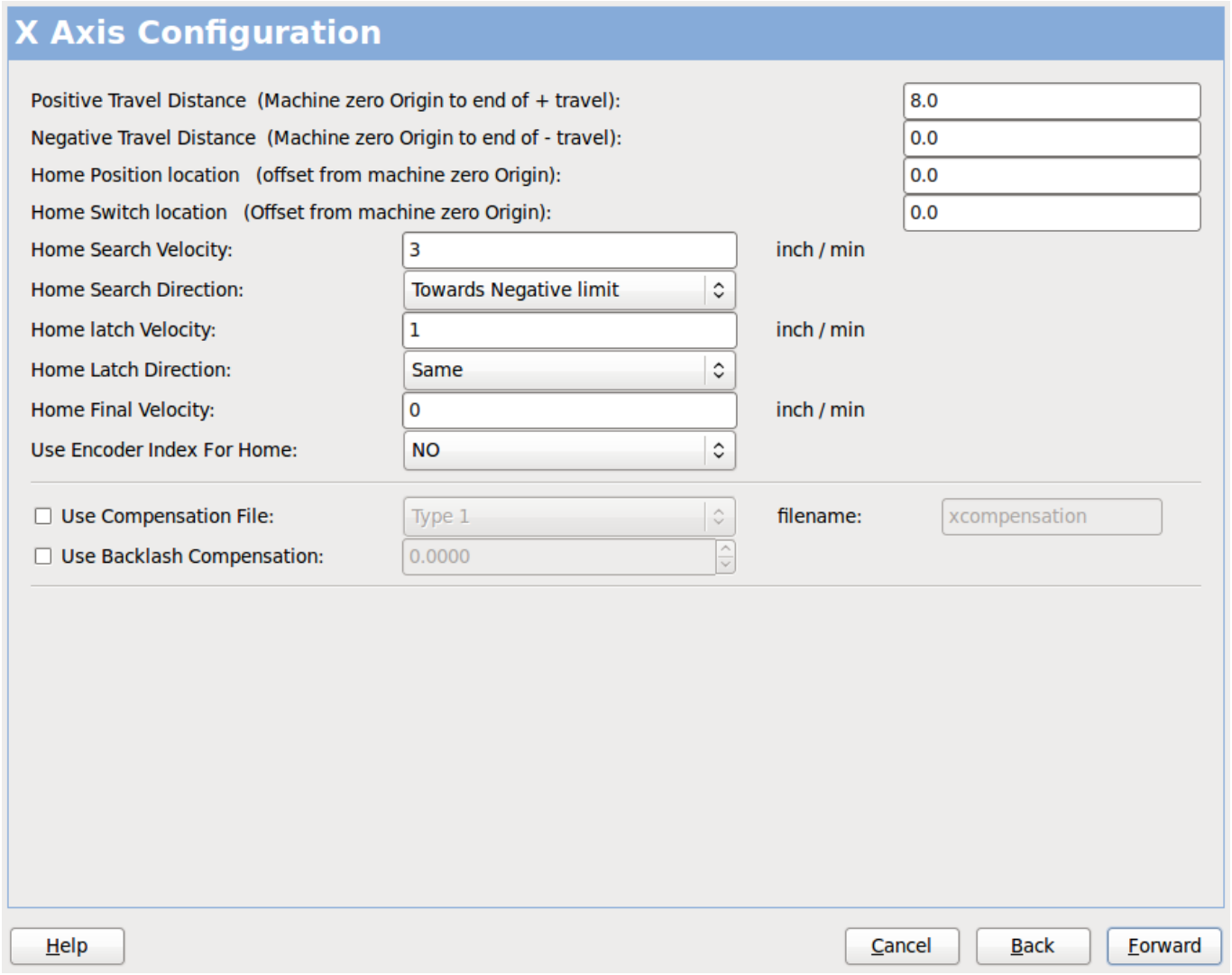


Figura 3.23: Configuración de ejes

Consulte también la pestaña del diagrama para ver dos ejemplos de interruptores de casa y límite. Son dos ejemplos de muchas formas diferentes de establecer homing y límites.

importante
Es muy importante comenzar con el eje moviéndose en la dirección correcta porque si no ¡llegar bien a casa es muy difícil!

Recuerda que direcciones positivas y negativas se refieren a la HERRAMIENTA, no a la mesa, según el Manual de maquinistas.

En una típica fresadora de rodilla o de cama

- cuando la MESA se mueve hacia fuera, es la dirección positiva de Y
- cuando la MESA se mueve a la izquierda, esa es la dirección X positiva

- cuando la MESA se mueve hacia abajo, esa es la dirección Z positiva
- cuando la CABEZA se mueve hacia arriba, esa es la dirección Z positiva

En un torno típico

- cuando la HERRAMIENTA se mueve hacia la derecha, lejos del mandril
- esa es la dirección positiva de Z
- cuando la HERRAMIENTA se mueve hacia el operador
- esa es la dirección positiva de X. Algunos tornos tienen la X opuesta (p. ej. la herramienta en el lado de atrás), eso funcionará bien pero la visualización gráfica de AXIS no logrará reflejarlo.

Cuando se utilizan interruptores de homing y/o límite LinuxCNC espera que las señales HAL sean verdaderas cuando el interruptor está siendo presionado / disparado. Si la señal es incorrecta para un interruptor de límite entonces LinuxCNC pensará que la máquina está al final del límite todo el tiempo. Si la lógica de búsqueda del interruptor de casa es incorrecta parecerá que LinuxCNC va casa en la dirección equivocada, cuando realmente intenta regresar al interruptor de la casa.

Decidir sobre la ubicación de interruptor de límite
Los interruptores de límite son el respaldo a los límites de software en el caso de que algo eléctrico salga mal, p. ej. en caso de un servomotor desbocado. Los interruptores de límite deben colocarse de manera que la máquina no golpee el extremo físico del movimiento del eje. Recuerda el eje pasará por el punto de contacto si se mueve rápido. Los interruptores de límite deben ser *activo bajo* en la máquina, p. ej. la energía corre a través los interruptores todo el tiempo - una pérdida de energía (interruptor abierto) dispara. Si bien uno podría conectarlos de la otra forma, esto es a prueba de fallas. Es posible que deba invertirse para que la señal HAL en LinuxCNC sea *activo alto* - un VERDADERO significa que el interruptor se disparó. Cuando iniciando LinuxCNC si recibe una advertencia de límite y el eje NO está accionando el interruptor, invertir la señal es probablemente la solución. (use HALMETER para revisar la señal HAL correspondiente p. ej. joint.0.pos-lim-sw-in interruptor de límite positivo del eje X)

Decida la ubicación del interruptor de casa
Si está utilizando interruptores de límite, también puede utilizar uno como interruptor de casa. Un interruptor de casa separado es útil si se tiene un eje largo que al estar en uso suele estar muy lejos de los interruptores de límite, o cuando al mover el eje hacia los extremos presenta problemas de interferencia por el material. Considere que un eje largo en un torno hace que sea difícil llegar a los límites sin que la herramienta golpee el eje, por lo que un interruptor de casa separado más cerca de en medio puede ser mejor. Si tiene un codificador con índice, el interruptor de casa actúa como un curso y el índice será la ubicación real de casa.

Decidir sobre la posición de ORIGEN DE MÁQUINA
ORIGEN DE LA MÁQUINA es lo que utiliza LinuxCNC para hacer referencia de todos los sistemas de coordenadas del usuario. Se me ocurre una pequeña razón por la que tendría que estar en cualquier lugar. Sólo hay unos pocos códigos G que pueden acceder al sistema de COORDENADAS DE MÁQUINA (G53, G30 y G28) Si utiliza la opción de cambio de herramienta en G30, puede ser conveniente tener el origen en la posición de cambio de herramienta. Por convención, puede ser más fácil tener el ORIGEN en el interruptor de casa.

Decidir sobre la POSICIÓN DE CASA (final)
esto solo coloca el carro en una posición consistente y conveniente después de que LinuxCNC descubre dónde está ORIGEN.

Medir / calcular las distancias de carrera del eje positivo / negativo
Mueve el eje al origen. Marca una referencia en la guía movible y otra en el soporte fijo (estando alineados) y mueve la máquina hasta el final de los límites. La medida entre las marcas es una de las distancias de carrera. Mueve la mesa al otro extremo de la carrera. Mide las marcas de nuevo. Esa es la otra distancia de carrera. Si el origen está en uno de los límites entonces la distancia de carrera será cero.

ORIGEN (de máquina)
El Origen es el punto cero de la MÁQUINA. (no el punto cero donde colocó el cortador / material). LinuxCNC usa este punto para -desde dónde- referenciar todo lo demás. Debe estar dentro de los límites del software. LinuxCNC usa la ubicación del interruptor de casa para calcular la posición de origen (cuando se utilizan interruptores de casa o debe ponerse manualmente si no se utilizan interruptores de casa.

Distancia de carrera
Esta es la distancia máxima que el eje puede viajar en cada dirección. Esto puede o no ser medido directamente desde el origen hasta el interruptor de límite. Las distancias de carreras positiva y negativa deben sumar la distancia total de carrera.

DISTANCIA POSITIVA DE CARRERA
Esta es la distancia que recorre el Eje desde el origen hasta la distancia de carrera positiva o la carrera total menos la distancia de carrera negativa. La pondrías a cero si el origen se posiciona en el límite positivo. Siempre será cero o un número positivo.

DISTANCIA DE CARREA NEGATIVA
Esta es la distancia que recorre el Eje desde el origen hasta la distancia de carrera negativa. o la carrera total menos la distancia de carrera positiva. Lo pondrías a cero si el origen se posiciona en el límite negativo. Esto siempre será cero o un número negativo. Si olvidas hacerlo negativo PnCconf lo hará internamente.

POSICIÓN DE CASA (Final)
Esta es la posición en la que terminará la secuencia de casa. Se hace referencia desde el origen por lo que puede ser negativa o positiva dependiendo de qué lado se encuentra al Origen. Cuando esta en la posición casa (final) y debes moverte en la dirección positiva para llegar al Origen, entonces el número será negativo.

UBICACIÓN DEL INTERRUPTOR DE CASA
Esta es la distancia desde el interruptor de la casa a el origen. Podría ser negativo o positivo dependiendo de qué lado del Origen es situado. Estando en la ubicación del interruptor casa, si debes moverte en la dirección positiva para llegar al Origen, entonces el número será negativo. Si configura esto a cero entonces el El origen estará en la ubicación del interruptor límite (más la distancia para encontrar el índice si se usa).

Velocidad de búsqueda de casa
Velocidad de búsqueda curso a casa en unidades por minuto.

Dirección a buscar casa
Establece la dirección de búsqueda del interruptor de casa ya sea negativo (es decir, hacia el interruptor de límite negativo) o positivo (es decir, hacia el interruptor de límite positivo).

Velocidad de cierre de casa
Velocidad de búsqueda fina de casa en unidades por minuto.

Velocidad de casa final
Velocidad utilizada desde la posición de cierre hasta la posición casa (final) en unidades por minuto. Se establece en 0 para la velocidad rápida máxima.

Dirección del cerrojo de casa
Permite ajustar la dirección del cerrojo a la misma o al contrario de la dirección de búsqueda.

Usar el índice del codificador para casa
LinuxCNC buscará un pulso de índice de codificador mientras esté en la etapa de cierre de homing.

Utilizar archivo de compensación
Permite especificar un nombre de archivo Comp y el tipo. Permite una compensación sofisticada. Ver Sección AXIS del capítulo INI.

Utilice la compensación de contragolpe
Permite el ajuste de la compensación de holgura mecánica. No se puede utilizar con el archivo de compensación. Ver Sección AXIS del capítulo INI.

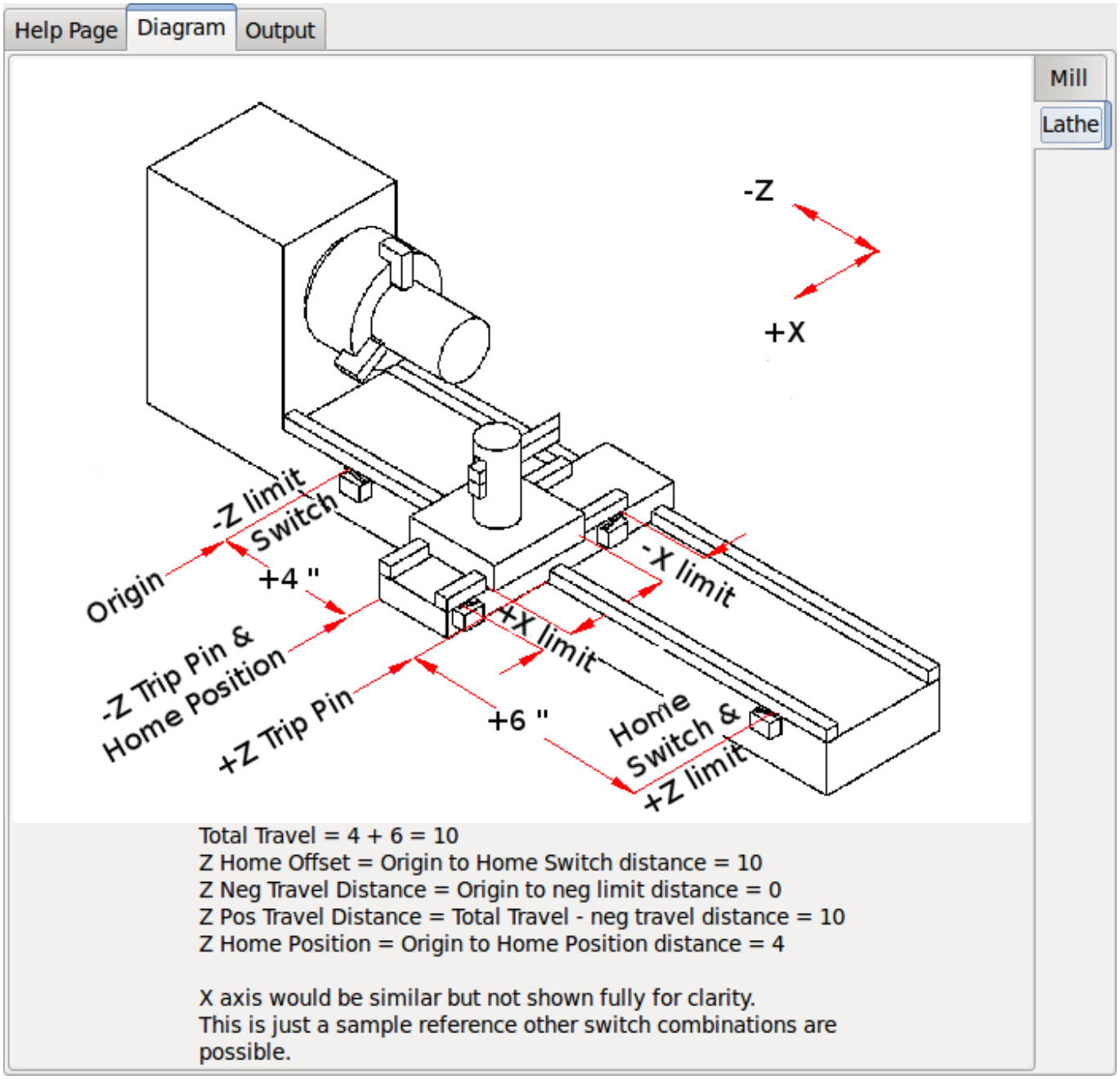


Figura 3.24: Diagrama de ayuda de AXIS

Los diagramas deberían ayudar a demostrar un ejemplo de interruptores de límite y direcciones de movimiento del eje estándar. En este ejemplo, el eje Z tiene dos interruptores de límite, el interruptor positivo se comparte como un interruptor casa. El ORIGEN DE LA MÁQUINA (punto cero) se encuentra en el límite negativo. El borde izquierdo del carro es el pin de disparo negativo y el derecha el pin de disparo positivo. Deseamos que la POSICIÓN FINAL DEL CASA esté a 4 pulgadas del ORIGEN en el lado positivo. Si el carro se moviera al límite positivo, mediríamos 10 pulgadas entre el límite negativo y el pin de disparo negativo.

3.2.10. Configuración del husillo

Si selecciona señales de husillo, esta página estará disponible para configurar el control del husillo.

sugerencia
¡Muchas de las opciones en esta página no se mostrarán a menos que la opción correcta haya sido seleccionada en páginas anteriores!

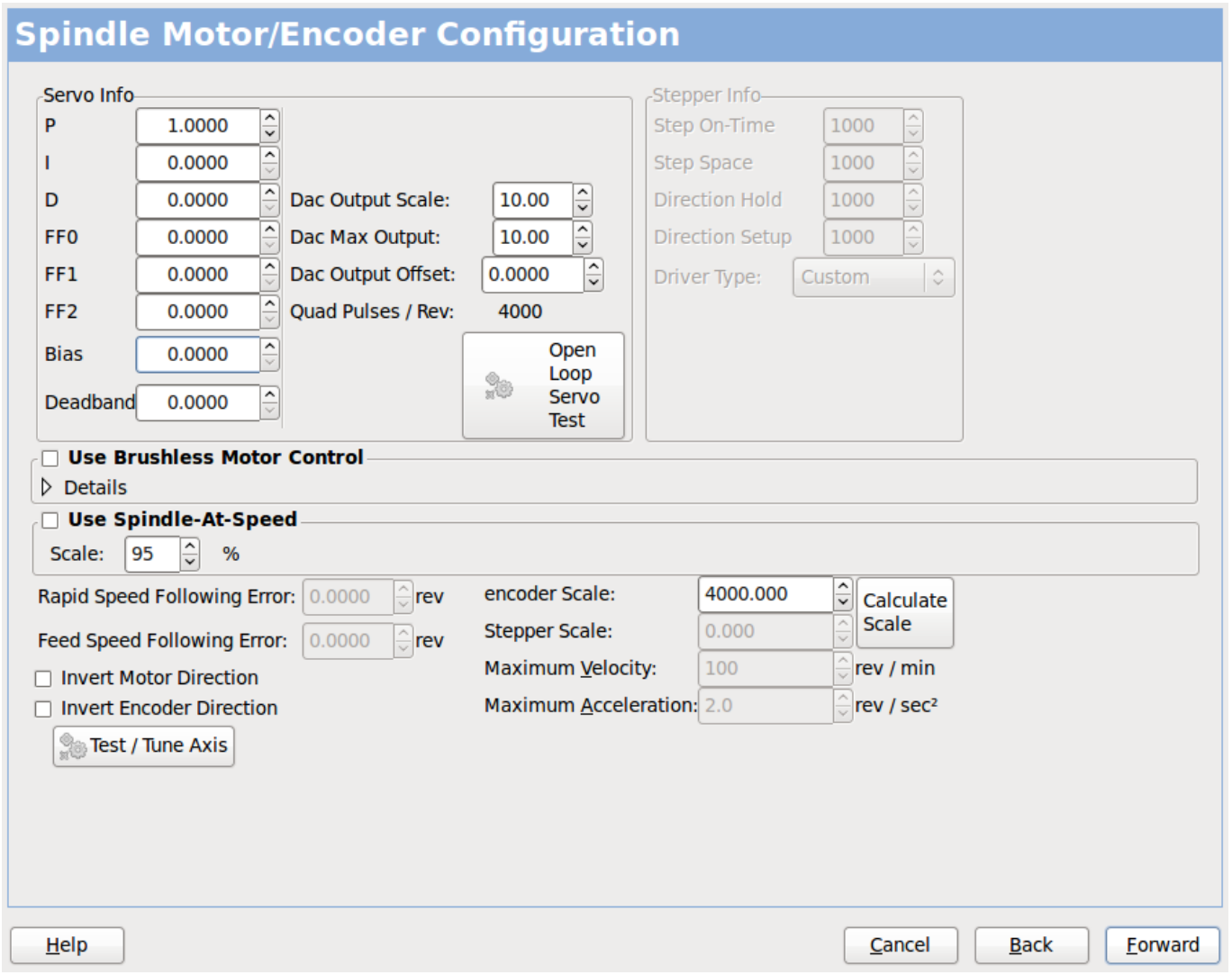


Figura 3.25: Configuración del motor/codificador de husillo

Esta página es similar a la página de configuración del motor del eje.

Hay algunas diferencias:

- A menos que uno haya elegido un eje accionado por pasos, no hay aceleración o limitación de velocidad.
- No hay soporte para cambios de engranajes o rangos.

- Si seleccionó una opción de visualización de husillo VCP, entonces podrán mostrarse las configuraciones de escala de husillo a velocidad y filtro.
- Spindle-at-speed permite a LinuxCNC esperar hasta que el husillo esté a la velocidad requerida antes de mover el eje. Esto es particularmente útil en tornos con alimentación de superficie constante y cambios de diámetro de velocidad grandes. Requiere cualquiera de los codificadores retroalimentación o una señal digital de velocidad de giro típicamente conectada a un control VFD.
- Si utiliza retroalimentación de codificador, puede seleccionar un ajuste de escala de husillo a-velocidad que especifique qué tan cerca debe estar la velocidad real a la velocidad solicitada para ser considerada a-velocidad.
- Si se utiliza retroalimentación de codificador, la visualización de velocidad VCP puede ser errática - la configuración del filtro se puede utilizar para suavizar la pantalla. La escala del codificador debe ser configurada para el codificador contador / engranaje utilizado.
- Si utiliza una sola entrada para un codificador de husillo, debe agregar la línea: setp hm2_7i43.0.encoder.0 mode 1 (cambiando el nombre de la placa y el número de codificador según sea necesario) en un archivo HAL personalizado. Para más información sobre el modo de contador consulte la Sección de codificadores en Hostmot2.

3.2.11. Opciones avanzadas

Esto permite configurar los comandos HALUI y cargar ClassicLadder y programas de escalera de muestra. Si seleccionó las opciones de GladeVCP como para puesta a cero del eje, se mostrarán comandos. Consulte el Capítulo HALUI para obtener más información sobre el uso de halcmds personalizados. Hay varias opciones de programa de escalera. El programa Estop permite un interruptor ESTOP externo o que la interfaz gráfica de usuario emita un Estop. También tiene una señal de bomba de lubricación temporizada. El touch-off de Z automático es con una placa de touch-off, el botón de touch-off de GladeVCP y comandos especiales HALUI para establecer el origen de usuario actual en cero y limpieza rápida. El programa modbus serie es básicamente un programa plantilla en blanco que configura ClassicLadder para modbus serie. Ver el Capítulo ClassicLadder en el manual.

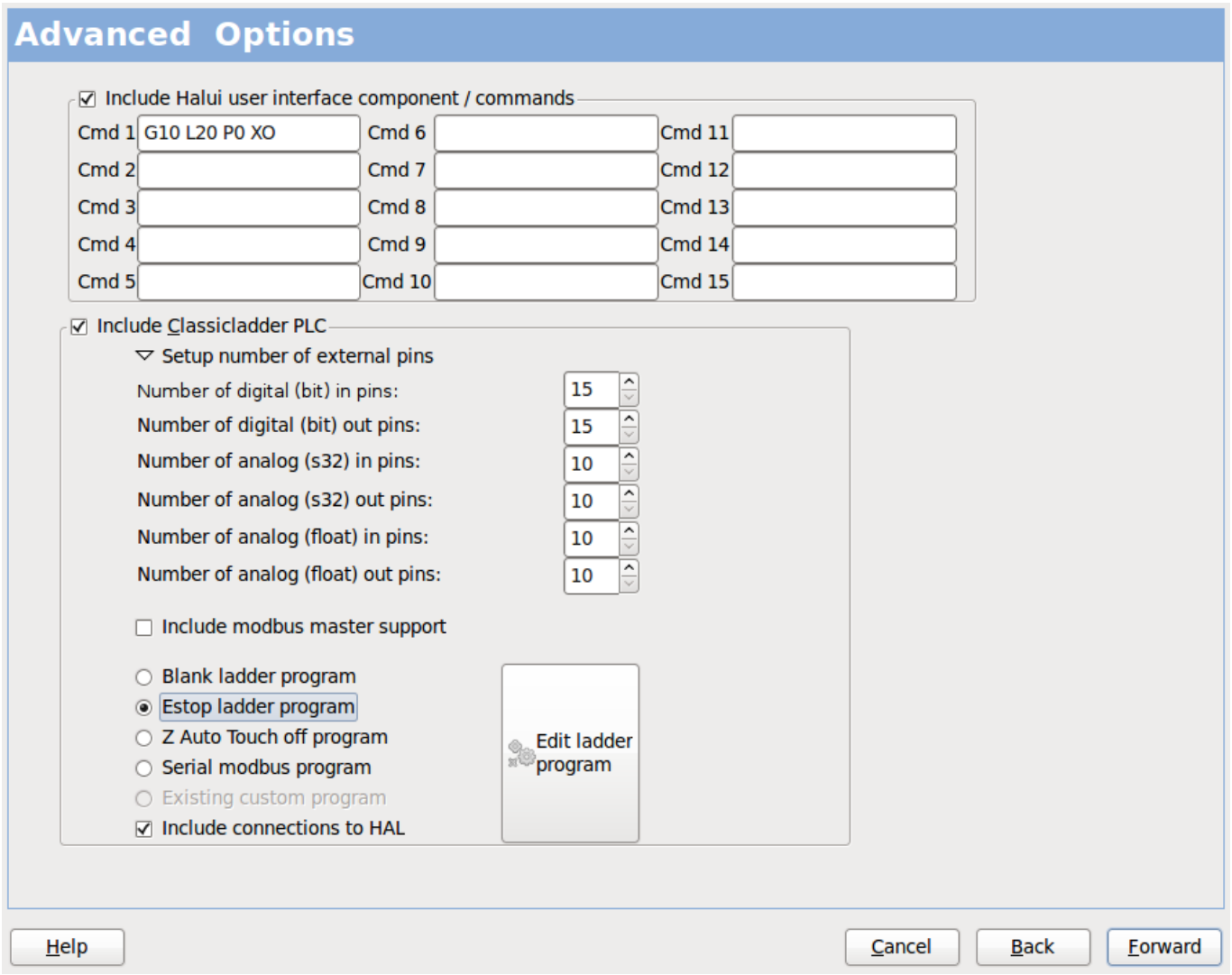


Figura 3.26: PnCconf, opciones avanzadas

3.2.12. Componentes HAL

En esta página puede agregar componentes HAL adicionales que pueda necesitar para archivos HAL personalizados. De esta manera, uno no debería tener que editar manualmente el archivo HAL principal, permitiendo aún los componentes que necesite el usuario.

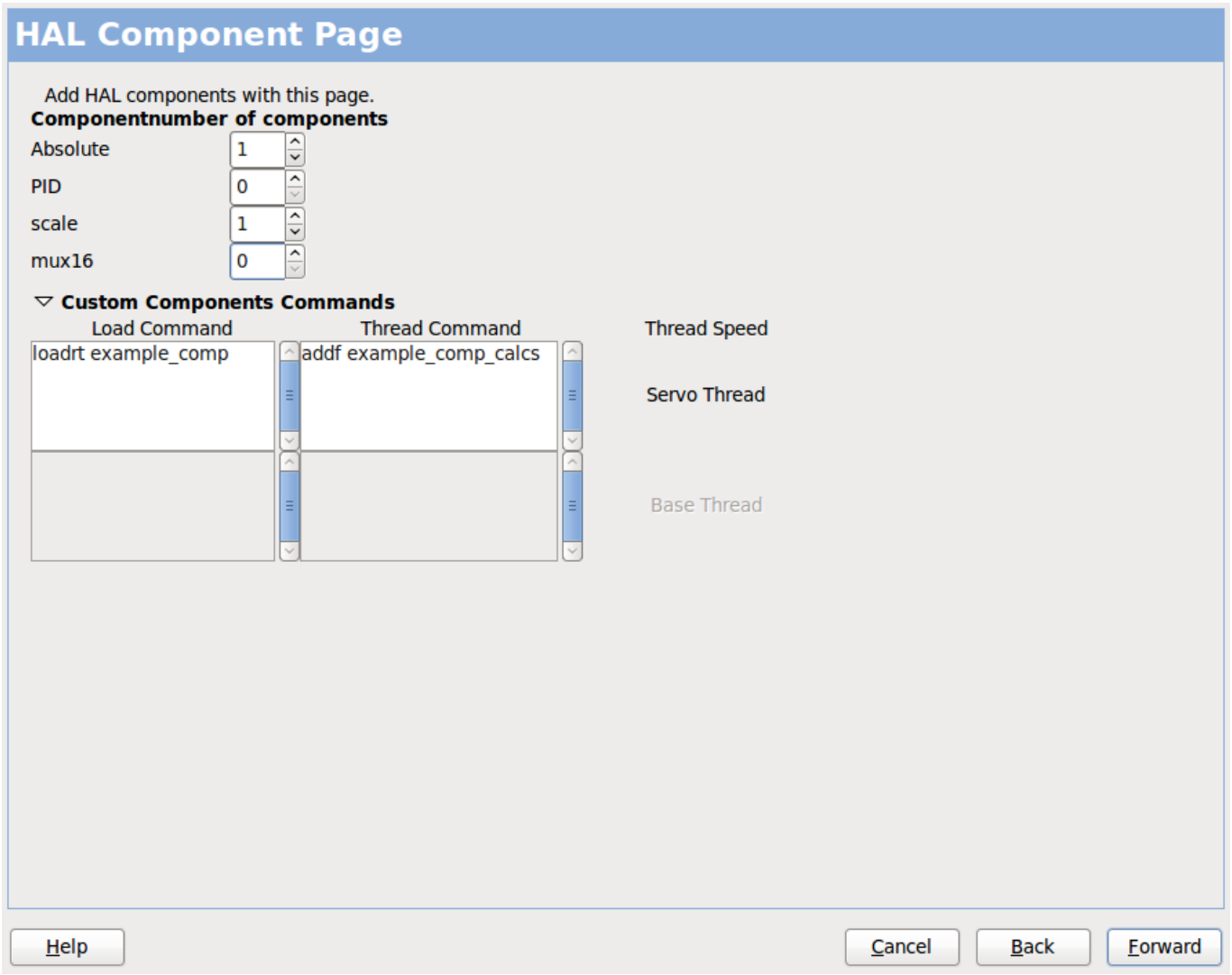


Figura 3.27: Componentes HAL

La primera selección es de componentes que PnCconf utiliza internamente. Puede configurar PnCconf para cargar instancias adicionales de los componentes para su archivo HAL personalizado.

Seleccione el número de instancias que necesitará su archivo personalizado, PnCconf agregará las que necesite después de ellas.

Esto significa que si tú necesitas 2 y PnCconf necesita 1, PnCconf cargará 3 instancias y utilizará la última.

Comandos de componentes personalizados

Esta selección le permitirá cargar componentes HAL que PnCconf no usa. Agregue el comando loadrt o loadusr, bajo el encabezado *Comando de carga* agregue el comando addf bajo el encabezado *Hilo de comando*. Los componentes se agregarán al hilo entre la lectura de entradas y la escritura de salidas, en el orden en que se escriben en el *Hilo de comando*.

3.2.13. Uso avanzado de PnCconf

PnCconf hace todo lo posible para permitir una personalización flexible por parte del usuario. PnCconf es compatible con nombres de señal personalizados, carga personalizada de componentes, archivos

HAL personalizados y firmware personalizado.

También hay nombres de señales que PnCconf siempre proporciona independientemente de las opciones seleccionado, para archivos HAL personalizados del usuario Pensando un poco la mayoría de las personalizaciones deberían funcionar independientemente de si posteriormente seleccionas opciones diferentes en PnCConf.

Eventualmente, si las personalizaciones están fuera del alcance del marco de trabajo de PnCconf, puedes usar PnCconf para construir una configuración base o usar uno de los ejemplos de configuración de LinuxCNC y solo editar a mano lo que quieras.

Nombres de señales personalizadas
: Si desea conectar un componente a algo en un archivo HAL personalizado, escriba un nombre de la señal único en el cuadro de entrada. Ciertos componentes agregarán terminaciones a su nombre de señal personalizado:

Codificadores agregarán < nombre personalizado > +:

- posición
- conteo
- velocidad
- habilitación de índice
- reinicio

Motores a pasos agregan:

- habilitación
- conteos
- posición del comando
- posición-fb
- velocidad-fb

PWM añade:

- habilitación
- valor

Los pines GPIO solo tendrán conectado el nombre de la señal ingresada

De esta manera, uno puede conectarse a estas señales en los archivos HAL personalizados y aún así tienes la opción de moverlos más tarde.

Nombres de señales personalizadas
: La página de Componentes de HAL se puede usar para cargar los componentes que necesita un usuario para personalización.

Cargando firmware personalizado
: PnCconf busca el firmware en el sistema y luego busca el archivo XML que puede convertir a lo que entiende. Esos archivos XML solo se suministran para firmware liberado oficialmente por el equipo de LinuxCNC. Para utilizar un firmware personalizado uno debe convertirlo en una matriz que PnCconf entienda y agregar su ruta de archivo al archivo de preferencias de PnCconf. Predeterminadamente esta ruta busca en el escritorio una carpeta llamada custom_firmware y un archivo llamado firmware.py.

El archivo de preferencias oculto está en el archivo de inicio del usuario, es llamado .pncconf-preferences y requiere uno para seleccionar *mostrar archivos ocultos* en tu administrador de archivos para ver y editarlo o en la línea de comandos usa *ls* con la opción *-a*. El contenido de este archivo se puede ver cuando carga por primera vez PnCconf - presione el botón de ayuda y mire la página de salida.

Pregunte en la lista de correo o el foro de LinuxCNC para obtener información sobre la conversión de firmware personalizado. No todo el firmware se puede utilizar con PnCconf.

Archivos HAL personalizados

Hay cuatro archivos personalizados que puede usar para agregarles comandos HAL:

- custom.hal es para comandos HAL que no tienen que ejecutarse después de cargar la interfaz GUI. Se ejecuta después del archivo HAL configuración-nombrada.
- custom_postgui.hal es para comandos que deben ejecutarse después de que se cargue AXIS o una visualización PyVCP independiente.
- custom_gvcp.hal es para comandos que deben ejecutarse después de cargar GladeVCP.
- shutdown.hal es para que los comandos se ejecuten cuando LinuxCNC se apaga de manera controlada.

Capítulo 4

Configuración

4.1. Conceptos para integradores

4.1.1. Ubicaciones de archivos

LinuxCNC busca la configuración y los archivos de código G en un lugar específico. La ubicación depende de cómo ejecute LinuxCNC.

4.1.1.1. Instalado

Si está ejecutando LinuxCNC desde el LiveCD o si lo instaló a través de un .deb y usa el selector de configuración *LinuxCNC* del menú LinuxCNC, mire en los siguientes directorios:

- El directorio LinuxCNC se encuentra en */home/dir-de-usuario/linuxcnc*.
- Los directorios de configuración se encuentran en */home/dir-de-usuario/linuxcnc/configs*.
  - Los archivos de configuración se encuentran en */home/dir-de-usuario/linuxcnc/configs/nombre-de-config*.
- Los archivos de código G se encuentran en /home/dir-de-usuario/linuxcnc/nc_files ’.

Por ejemplo, para una configuración llamada mill y nombre de usuario Fred, los directorios y la estructura de archivos se verían así.

- */home/fred/linuxcnc*
- */home/fred/linuxcnc/nc_files*
- */home/fred/linuxcnc/configs/mill*
  - */home/fred/linuxcnc/configs/mill/mill.ini*
  - */home/fred/linuxcnc/configs/mill/mill.hal*
  - */home/fred/linuxcnc/configs/mill/mill.var*
  - */home/fred/linuxcnc/configs/mill/tool.tbl*

4.1.1.2. Línea de comandos

Si ejecuta LinuxCNC desde la línea de comandos y especifica el nombre y la ubicación del archivo INI, las ubicaciones de los archivos pueden estar en un lugar diferente. Para ver las opciones para ejecutar LinuxCNC desde la línea de comandos ejecute *linuxcnc -h*.

---

nota
Las ubicaciones opcionales para algunos archivos se pueden configurar en el archivo INI. Ver las secciones <<sub:ini:sec:display,[DISPLAY]>> y <<sub:ini:sec:rs274ngc,[RS274NGC]>>.

4.1.2. Archivos

Cada directorio de configuración requiere al menos los siguientes archivos:

- Un archivo INI .ini
- Un archivo HAL .hal o un archivo HALTCL .tcl especificado en la sección HAL del archivo INI.

---

nota
Otros archivos pueden ser necesarios para algunas GUI.

Opcionalmente también puede tener:

- Un archivo de variables .var
  - Si omite un archivo .var en un directorio pero incluye <<sub:ini:sec:rs274ngc,[RS274NGC]>> PARAMETER_FILE=algunArchivo.var, el archivo se creará para usted cuando se inicie LinuxCNC.
  - Si omite un archivo .var y omite el elemento [RS274NGC]PARAMETER_FILE, se creará el archivo var rs274ngc.var cuando se inicie LinuxCNC. Puede haber algunos mensajes confusos si se omite [RS274NGC]PARAMETER_FILE.
- Un archivo de tabla de herramientas .tbl si <<sub:ini:sec:emcmot,[EMCMOT]>> TOOL_TABLE ha sido especificado en el archivo INI. Algunas configuraciones no necesitan una tabla de herramientas.

4.1.3. Sistemas steppers

4.1.3.1. Período base

BASE_PERIOD es el *latido* de su sistema LinuxCNC. [1] En cada período, el generador de pasos software decide si es hora de otro pulso de paso. Un período más corto le permitirá generar más pulsos por segundo, dentro de los limites. Pero si se hace muy corto, la computadora gastará mucho tiempo generando pulsos de paso y todo lo demás se ralentizara, o tal vez incluso se bloqueara. Los requisitos de latencia y de los controladores de motor afectan el período más corto que pueda usar.

Las latencias, en el peor de los casos, solo pueden ocurrir unas pocas veces por minuto, y las probabilidades de que ocurra una mala latencia justo cuando el motor está cambiando de dirección son bajas. Por lo tanto, puede obtener errores muy raros que arruinen una pieza de vez en cuando y son imposibles de solucionar.

[1] Esta sección se refiere al uso de **stepgen**, generador de pasos incorporado en LinuxCNC. Algunos dispositivos de hardware tienen su propio generador de pasos y no usan el incorporado de LinuxCNC. En ese caso, consulte su manual de hardware.

La forma más sencilla de evitar este problema es elegir un BASE_PERIOD que sea la suma del requisito de tiempo más largo de su controlador, y el peor caso de latencia de su computadora. Esto no siempre es la mejor opción. Por ejemplo, si está usando un controlador con un requisito de tiempo de espera de señal de dirección de 20 µs, y su prueba de latencia dijo que tiene una latencia máxima de 11 µs, entonces si configura BASE_PERIOD en 20+11 = 31 µs obtendrá unos nada bonitos 32,258 pasos por segundo en un modo y 16,129 pasos por segundo en otro modo.

El problema está con el requisito de tiempo de espera de 20 µs. Eso, más los 11 µs de latencia es lo que nos obliga a usar un período lento de 31 µs. Pero el generador de pasos de software de LinuxCNC tiene algunos parámetros que le permiten aumentar de un período a varios. Por ejemplo, si *steplen* [2] es cambiado de 1 a 2, entonces habrá dos períodos entre el principio y final del pulso de paso. Del mismo modo, si *dirhold* [3] es cambiado de 1 a 3, habrá al menos tres períodos entre el pulso de paso y un cambio del pin de dirección.

Si podemos usar *dirhold* para cumplir con el requisito de 20 µs de tiempo de espera, entonces el siguiente tiempo más largo son los 4.5 µs de tiempo alto. Agrega a la latencia de 11 µs los 4.5 µs de tiempo alto, y se obtiene un período mínimo de 15.5 µs. Cuando intentas 15.5 µs, encontrarás que la computadora es lenta, por lo que lo estableces en 16 µs. Si dejamos *dirhold* en 1 (valor predeterminado), entonces el tiempo mínimo entre paso y dirección es el período de 16 µs menos la latencia de 11 µs = 5 µs, que no es suficiente. Necesitamos otros 15 µs. Como el período es 16 µs, necesitamos un período más. Entonces cambiamos *dirhold* de 1 a 2. Ahora el tiempo mínimo desde el final del pulso de paso hasta el pin de cambio dirección es 5+16 = 21 µs, y no tendremos que preocuparnos por que el controlador de pasos en la dirección equivocada debido a la latencia.

Para obtener más información sobre stepgen, consulte la sección stepgen.

4.1.3.2. Tiempos de pasos

Step Timing y Step Space son diferentes en algunos drivers. En este caso el punto de paso se vuelve importante. Si el driver manda pasos en borde descendente, entonces el pin de salida debe invertirse.

4.1.4. Servosistemas

4.1.4.1. Operación básica

Los servosistemas tienen mayor velocidad y precisión que los sistemas paso a paso equivalentes, pero son más costosos y complejos. A diferencia de los sistemas paso a paso, los servosistemas requieren algún tipo de dispositivo de retroalimentación de posición, y debe ajustarse o *afinarse*, ya que no funcionan tan al salir de la caja como lo haría un sistema paso a paso. Estas diferencias existen porque los servos son sistemas de *lazo cerrado*, a diferencia de los motores paso a paso que generalmente funcionan en *lazo abierto*. ¿Qué significa *lazo cerrado*?. Veamos un diagrama simplificado de cómo está conectado un sistema servomotor.

[2] steplen se refiere a un parámetro que ajusta el rendimiento del generador de pasos incorporado de LinuxCNC, *stepgen*, que es un componente HAL. Este parámetro ajusta la longitud del pulso de paso en sí mismo. Sigue leyendo, todo se explicará eventualmente.

[3] dirhold se refiere a un parámetro que ajusta la duración del tiempo de espera de dirección.

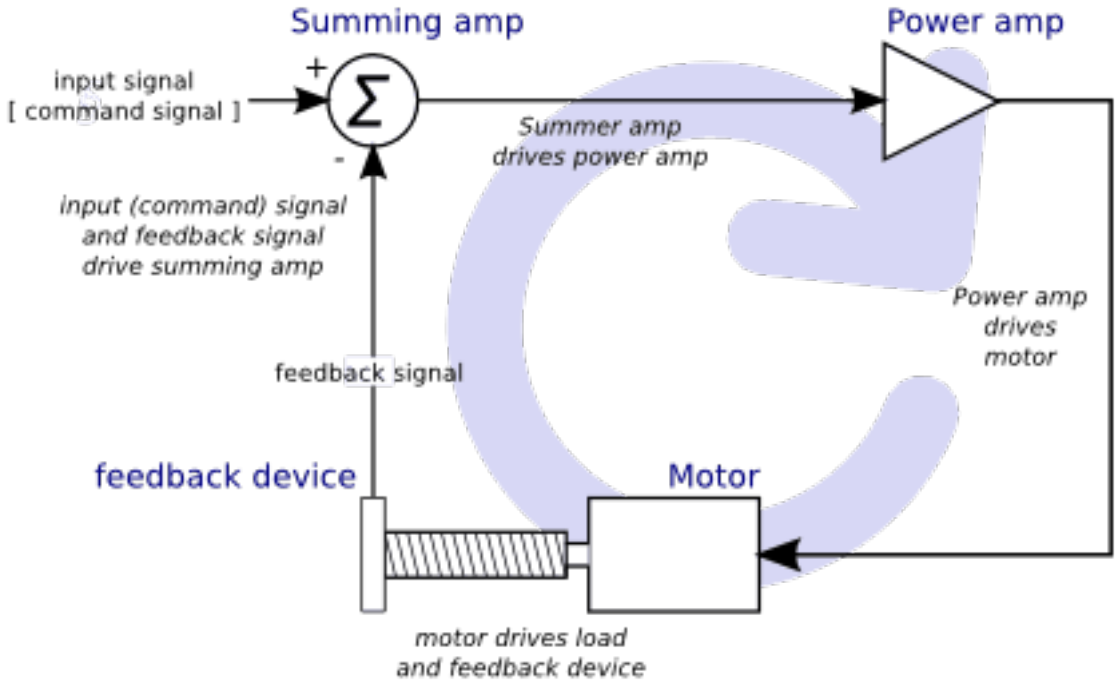


Figura 4.1: Bucle de servo

Este diagrama muestra que la señal de entrada (y la señal de retroalimentación) alimentan un amplificador sumador, el amplificador sumador acciona el amplificador de potencia, el amplificador de potencia impulsa el motor, el motor impulsa la carga (y el dispositivo de retroalimentación) y el dispositivo de retroalimentación (y la señal de entrada) vuelven a manejar el motor. Esto se parece mucho a un círculo (un lazo cerrado) donde A controla B, B controla C, C controla D y D controla A.

Si no ha trabajado antes con servosistemas, sin duda esto parecerá una idea muy extraña al principio, especialmente en comparación con la electrónica normal de circuitos donde las entradas proceden suavemente a las salidas y nunca van hacia atrás. [4] Si *todo* controla *todo lo demás*, ¿cómo puede funcionar eso, quién está a cargo?. La respuesta es que LinuxCNC *puede* controlar este sistema, pero tiene que hacerlo eligiendo uno de varios métodos de control. El método de control que utiliza LinuxCNC, uno de los más simples y mejores, se llama PID.

PID significa Proporcional, Integral y Derivativo. El valor proporcional determina la reacción al error actual, el valor integral determina la reacción en función de la suma de los errores recientes, y el valor derivado determina la reacción en función de la velocidad a la que el error ha estado cambiando. Son tres técnicas matemáticas comunes que se aplican a la tarea de lograr que un trabajo en proceso siga un punto establecido. En el caso de LinuxCNC, el proceso que queremos controlar es la posición real del eje y el punto de trabajo o consigna es la posición ordenada del eje.

[4] Si sirve de ayuda, el equivalente más cercano a esto en el mundo digital son las *máquinas de estado*, *máquinas secuenciales* y demás, donde lo que hacen las salidas *ahora*, depende de lo que las entradas (y las salidas) estaban haciendo *antes*. Si no ayuda, no importa.

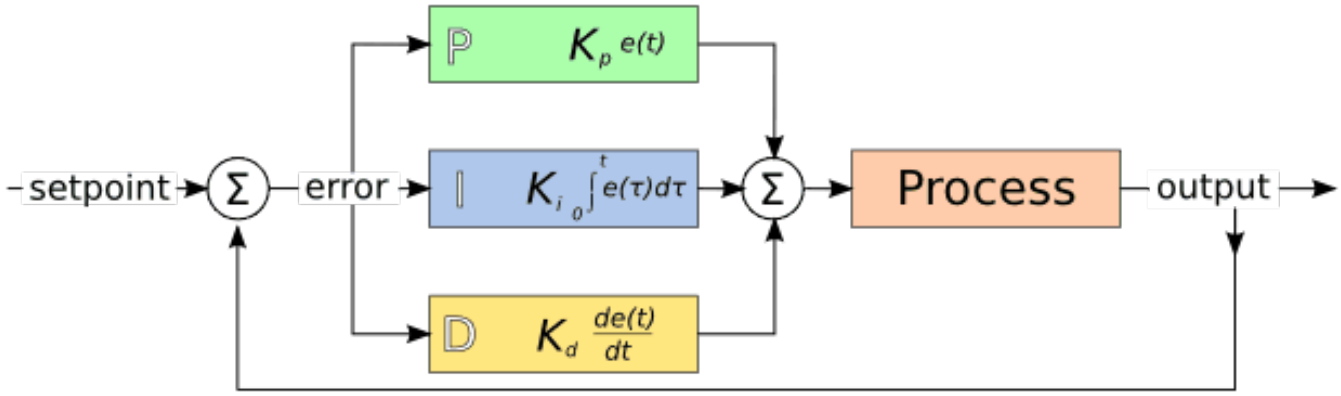


Figura 4.2: Ciclo PID

Al *ajustar* las tres constantes en el algoritmo del controlador PID, el controlador puede proporcionar acciones de control diseñadas para requisitos de procesos específicos. La respuesta del controlador puede describirse en términos de la capacidad de respuesta frente a un error, el grado con el que el controlador sobrepasa el punto de ajuste (overshoot) y el grado de oscilación del sistema.

4.1.4.2. Término proporcional

El término proporcional (a veces llamado ganancia) hace un cambio en la salida que es proporcional al valor de error actual. Una ganancia proporcional alta resulta en un gran cambio en la salida para un determinado cambio en el error. Si la ganancia proporcional es demasiado alta, el sistema puede volverse inestable. En contraste, una pequeña ganancia resulta en una respuesta de salida pequeña a un error de entrada grande. Si la ganancia proporcional es demasiado baja, la acción de control puede ser demasiado pequeña para responder a perturbaciones en sistema.

En ausencia de perturbaciones, el control proporcional puro no puede estabilizarse en su valor objetivo, pero retendrá un error de estado estable que es una función de la ganancia proporcional y la ganancia del proceso. A pesar del offset en estado estacionario, tanto la teoría de afinacion como la práctica industrial indican que es el término proporcional el que debe contribuir en mayor parte al cambio de la salida.

4.1.4.3. Término integral

La contribución del término integral (a veces llamado reset) es proporcional tanto a la magnitud del error como a la duración del mismo. Sumando el error instantáneo en el tiempo (integrando el error) da el desplazamiento acumulado que debería haberse corregido previamente. El error acumulado se multiplica por la ganancia integral y se suma a la salida del controlador.

El término integral (cuando se agrega junto al término proporcional) acelera el movimiento del proceso hacia el punto de ajuste y elimina el error residual de estado estacionario que ocurre con una controlador solo proporcional. Sin embargo, dado que el término integral responde a errores acumulados del pasado, puede causar que el valor presente sobrepase el valor del punto de ajuste (cruzar sobre el punto de ajuste y luego crear una desviación en la otra dirección).

4.1.4.4. Término derivativo

La velocidad de cambio del error de proceso se calcula determinando la pendiente del error en el tiempo (es decir, su primera derivada con respecto al tiempo) y multiplicando esta tasa de cambio por la ganancia derivativa.

El término derivativo reduce la velocidad de cambio de la salida del controlador y este efecto es más notable cerca del punto de ajuste del controlador. Por lo tanto, el control derivativo se utiliza para reducir la magnitud del sobreimpulso (overshoot) producido por el componente integral y mejorar la estabilidad de la combinación proceso / controlador.

4.1.4.5. Ajuste de bucle

Si los parámetros del controlador PID (las ganancias de los términos proporcional, integral y derivativo) se eligen incorrectamente, la entrada de proceso controlada puede ser inestable, es decir, su salida diverge, con o sin oscilación, y está limitada solo por saturación o rotura mecánica. Sintonizar un bucle de control es el ajuste de sus parámetros de control (ganancia proporcional/banda, ganancia integral/reset, ganancia derivativa/tasa) a los valores óptimos para la respuesta de control deseada.

4.1.4.6. Afinación manual

Un método de ajuste simple es establecer primero los valores I y D en cero. Aumente la P hasta que la salida del bucle oscile. Luego debe establecerse la P en aproximadamente la mitad de ese valor, buscando una respuesta del tipo *decaimiento de un cuarto de amplitud*. Despues se aumenta I hasta que cualquier offset se corrija en tiempo suficiente para el proceso. Sin embargo, demasiada I causará inestabilidad. Finalmente, aumente D, si es necesario, hasta que el ciclo sea aceptablemente rápido para alcanzar su referencia después de una perturbación de carga. Sin embargo, demasiada D causará una respuesta excesiva y un sobredisparo. Una afinación del bucle PID rápida generalmente sobredispara ligeramente para alcanzar el punto de ajuste más rapidamente; sin embargo, algunos sistemas no pueden aceptar el sobreimpulso, en cuyo caso se requiere un sistema de lazo cerrado *sobre-amortiguado*, que requerirá una P significativamente menor que la mitad de la configuración de P que causa oscilación.

4.1.5. S-Curve Trajectory Planning

S-curve trajectory planning limits jerk (the rate of change of acceleration) to provide smoother motion. This can reduce machine vibration and improve surface finish, but requires tuning additional parameters.

4.1.5.1. Enabling

Configurar en el archivo INI:

```
[TRAJ]
PLANNER_TYPE = 1              # 0=trapezoidal (default), 1=S-curve
MAX_LINEAR_JERK = 1000.0      # Machine units/s^3

[JOINT_n]
MAX_JERK = 1000.0
```

S-curve planning is only active when `PLANNER_TYPE = 1` and `MAX_LINEAR_JERK > 0`.

nota
If `MAX_LINEAR_JERK` is not specified, it defaults to 1e9 (1 billion), which effectively disables jerk limiting while maintaining S-curve calculations. This produces motion similar to trapezoidal planning but not identical. The maximum allowed value is 1e9 to prevent numerical instability.

4.1.5.2. Tuning

Start with a conservative jerk value and increase gradually:

```
MAX_JERK b’’≈b’’ 10 to 100 × MAX_ACCELERATION
```

Typical values: 100-100,000 units/s^3 depending on machine rigidity and units (mm values are typically 1000x larger than inch values).

Increase `MAX_LINEAR_JERK` until motion becomes sluggish or following errors increase, then reduce slightly. Test with coordinated moves and arcs.

Values above 1e9 are automatically clamped to 1e9 to avoid numerical issues in the S-curve trajectory calculations.

4.1.6. RTAI

La Interfaz de Aplicación en Tiempo real (RTAI) se utiliza para proporcionar el mejor rendimiento en tiempo real (RT). El kernel parcheado RTAI le permite escribir aplicaciones con restricciones de tiempo estrictas. RTAI da la habilidad tener cosas como la generación de pasos software que requieren gran precisión de sincronización.

4.1.6.1. ACPI

La Interfaz Avanzada de Configuración y Energía (ACPI) tiene muchas funciones diferentes, la mayoría de las cuales interfieren con el rendimiento RT (por ejemplo: administración de energía, apagado de CPU, escala de frecuencia de CPU, etc.) El núcleo LinuxCNC (y probablemente todos los núcleos parcheados con RTAI) tiene ACPI deshabilitado. ACPI también se encarga de apagar el sistema después de que se ha iniciado el apagado, y es por eso que podría ser necesario presionar el botón de encendido para apagar completamente su computadora. El grupo RTAI ha estado mejorando esto en versiones recientes, por lo que su sistema LinuxCNC puede apagarse por sí mismo después de todo.

4.1.7. Opciones de hardware de interfaz computadora/máquina

4.1.7.1. litehm2/rv901t

Litehm2 es un puerto agnóstico de placa del firmware FPGA de HostMot2. La primera placa que soporta es la linsn rv901t, la cual fue construida originalmente como una placa controladora de LEDs, pero debido a las E/S disponibles queda bien para actuar como un controlador de máquina. Ofrece alrededor de 80 puertos de E/S de 5V con búfer y puede intercambiar todas las entradas y salidas. También es fácilmente modificable para dividir los puertos mitad y mitad entre entradas y salidas. La rv901lt se interconecta con la computadora por medio de Ethernet Gigabit o de 100Mbit.

Litehm2 se basa en el framework LiteX, el cual soporta una amplia variedad de placas FPGA. Actualmente solo se soporta la rv901t , pero está en desarrollo el soporte de más placas.

Se puede encontrar más información en https://github.com/sensille/litehm2.

4.2. Prueba de latencia

4.2.1. ¿Qué es latencia?

La latencia es cuánto tiempo le toma a la PC detener lo que esta haciendo y responder a una solicitud externa, como ejecutar uno de los hilos periódicos en tiempo real de LinuxCNC. Cuanto menor sea

la latencia, más rápido se ejecutarán los hilos en tiempo real y más suave será el movimiento (y potencialmente más rápido, en caso de pasos por software).

La latencia es mucho más importante que la velocidad de la CPU. Un humilde Pentium II que responde siempre a las interrupciones en 10 us puede dar mejores resultados que una bestia más reciente y rápida de P4 Hyperthreading.

La CPU no es el único factor para determinar la latencia. Las placas base, tarjetas de video, puertos USB y un número de otros factores puede dañar la latencia. La mejor manera de averiguar con cuánta se esta tratando, es ejecutar la prueba de latencia.

Generar pulsos de paso en software tiene una gran ventaja: es gratis. Casi todas las PC que tienen un puerto paralelo son capaces de emitir pulsos de pasos generados por el software.

However, software step pulses also have some disadvantages:

- velocidad de paso máxima limitada
- jitter (fluctuaciones) en los pulsos generados
- añade carga a la CPU

4.2.2. Pruebas de latencia

LinuxCNC incluye varias pruebas de latencia. Todas producen información equivalente. Ejecutar estas pruebas ayudará a determinar si una computadora es adecuada para controlar una máquina CNC.

nota
No ejecute LinuxCNC o StepConf mientras se ejecuta la prueba de latencia.

4.2.2.1. Prueba de latencia

The latency test can be run a few different ways.

If you are using PnCconf to configure your machine, you can launch the Latency Test by clicking the ”Test Base Period Jitter button’ during the 2nd step of the process.

If you are using StepConf to configure your machine, you can launch the Latency Test by clicking the ”Test Base Period Jitter button’ during the 2nd step of the process.

If you want to run the test from the command line, open a terminal window (in Ubuntu, from Applications → Accessories → Terminal) and run the following command:

```
latency-test
```

Esto iniciará la prueba de latencia con un período base de hilo de 25 µs y un período servo-hilo de 1 ms. Los tiempos de período pueden ser especificados en la línea de comandos:

```
latency-test 50000 1000000
```

Esto iniciará la prueba de latency con un período base de hilo de 50 µs y un período de servo-hilo de 1 ms.

Para opciones disponibles, ingresar en la línea de comandos:

```
latency-test -h
```

Después de iniciar la prueba de latencia deberías ver algo como esto:

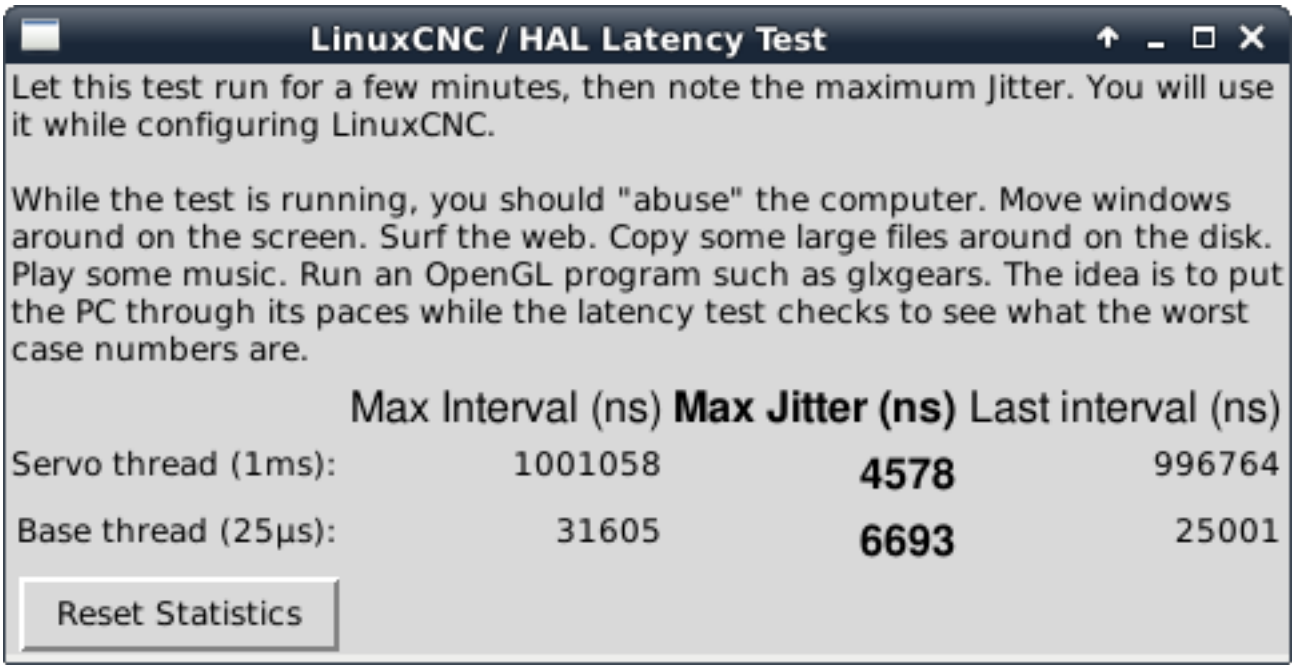


Figura 4.3: Prueba de latencia HAL

Mientras se ejecuta la prueba, debes *abusar* de la computadora. Mueve las ventanas alrededor de la pantalla. Navega por la web. Copia algunos archivos grandes en el disco. Pon música. Ejecuta un programa OpenGL como glxgears. La idea es poner la PC en aprietos mientras la prueba de latencia comprueba cuáles son los peores números.

Los números importantes son los de la columna *max jitter* (máxima fluctuación). En el ejemplo anterior, son 6693 nanosegundos (ns), o 6.693 microsegundos (µs). Anota este número e ingrésalo en StepConf cuando se solicite.

In the example above, latency-test only ran for a few seconds. You should run the test for at least several minutes; sometimes the worst case latency doesn't happen very often, or only happens when you do some particular action. For instance, one Intel motherboard worked pretty well most of the time, but every 64 seconds it had a very bad 300 µs latency. Fortunately that was fixable, see [?].

So, what do the results mean?

If your Max Jitter number is less than about 20,000 nanoseconds, the computer should give very nice results with software stepping or a dedicated hardware card such as a Mesa *Anything I/O* card.

If the Max Jitter number is between 20,000 and 50,000 nanoseconds, you can still get good results with software stepping, but your maximum step rate might be a little disappointing, especially if you use microstepping or have very fine pitch leadscrews. You can, however, achieve excellent results using a hardware card.

If the Max Jitter number is between 50,000 and 500,000 nanoseconds, you cannot use software stepping. You can, however, achieve acceptable results using a hardware card.

If the Max Jitter number is above 500,000 nanoseconds, you cannot use software stepping or a hardware card with LinuxCNC and achieve acceptable results.

---

nota
If you get high numbers, there may be ways to improve them. Another PC had very bad latency (several million nanoseconds) when using the onboard video. But a $5 used video card solved the problem. LinuxCNC does not require bleeding-edge hardware.

Para obtener más información sobre la afinación de pasos, consulta el capítulo Afinación de motores a pasos.

sugerencia
Hay herramientas adicionales disponibles desde la línea de comandos para examinar la latencia cuando LinuxCNC no está corriendo.

4.2.2.2. Gráfica de latencia

latency-plot crea una gráfica de barras registrando los hilos base y servo. Puede ser útil para ver picos de latencia cuando se inician o se usan otras aplicaciones. Uso:

```
latency-plot --help

Usage:
      latency-plot --help | -?
      latency-plot --hal [Options]

Options:
      --base ns  (base  thread interval in nanoseconds, default:   25000)
      --servo ns (servo thread interval in nanoseconds, default: 1000000)
      --time ms  (report interval in milliseconds, default: 1000)
      --relative (relative clock time (default))
      --actual   (actual clock time)
```

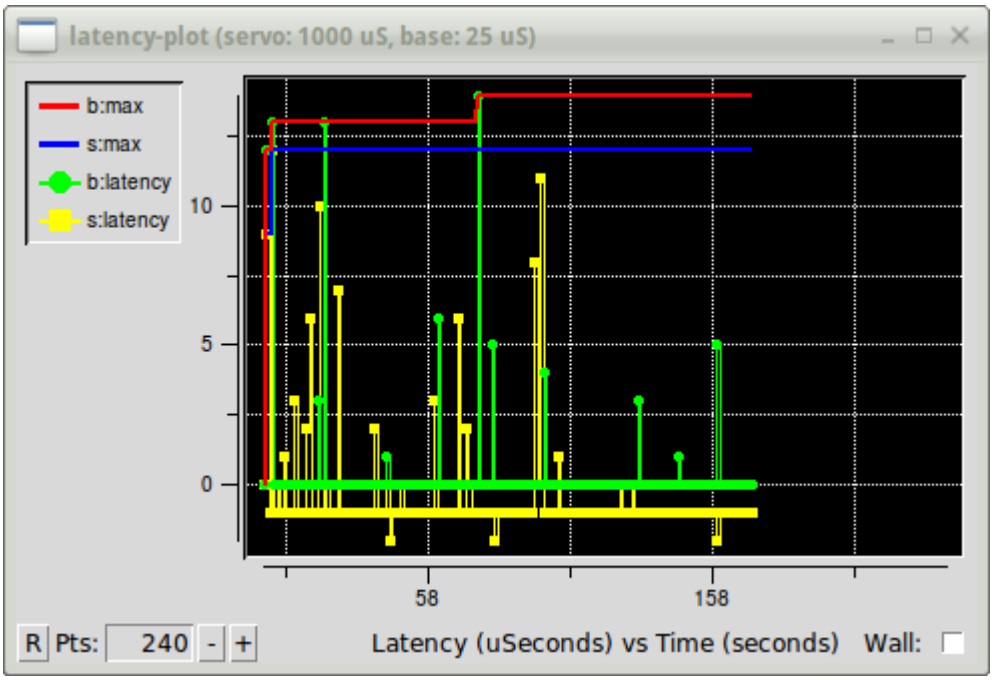


Figura 4.4: Ventana de `latency-plot`

4.2.2.3. Histograma de latencia

latency-histogram muestra un histograma de latencia (jitter) para hilos base y servo.

```
Usage:
   latency-histogram --help | -?
   latency-histogram [Options]

Options:
  --base      ns   (base  thread interval in nanoseconds, default:   25000, min:  5000)
  --servo     ns   (servo thread interval in nanoseconds, default: 1000000, min: 25000)
  --bbinsize  ns   (base  bin size in nanoseconds, default: 100
  --sbinsize  ns   (servo bin size in nanoseconds, default: 100
  --bbins     n    (base  bins, default: 200
  --sbins     n    (servo bins, default: 200
  --logscale  0|1  (y axis log scale, default: 1)
  --text      note (additional note, default: "" )
  --show           (show count of undisplayed bins)
  --nobase         (servo thread only)
  --verbose        (progress and debug)
  --nox            (no gui, display elapsed,min,max,sdev for each thread)
```

nota

When determining the latency, LinuxCNC and HAL should not be running, stop with `halrun -U`. Large number of bins and/or small binsizes will slow updates. For single thread, specify `--nobase` (and options for servo thread). Measured latencies outside the +/- bin range are reported with special end bars. Use `--show` to show count for the off-chart [pos|neg] bin.

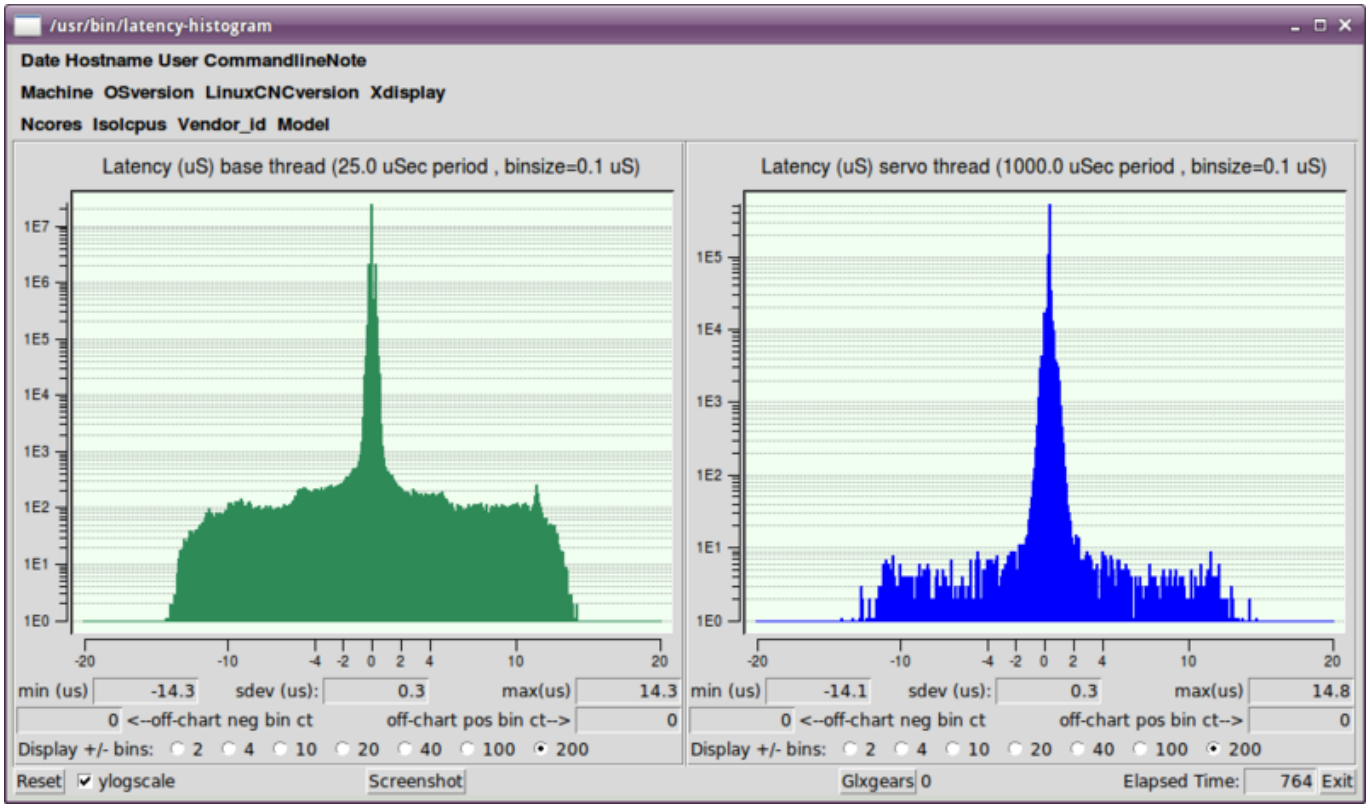


Figura 4.5: Ventana de `latency-histogram`

4.2.3. Ajuste de latencia

LinuxCNC puede correr en muchas plataformas de hardware distintas y con diferentes kernels de tiempo real, y todas ellas se pueden beneficiar del ajuste de latencia óptima.

El primer objetivo es ajustar el sistema para que LinuxCNC reserve un CPU para uso exclusivo de las tareas en tiempo real de LinuxCNC, para que otras tareas (tanto programas de usuario como hilos del kernel) no interfieran con los accesos al CPU de LinuxCNC.

Cuando se cree que opciones específicas de ajuste son universalmente de ayuda, LinuxCNC realiza automáticamente este ajuste en el arranque, pero muchas opciones de ajuste son específicas de la máquina y no pueden realizarse automáticamente. La persona que instale LinuxCNC necesita determinar experimentalmente el ajuste óptimo para su sistema.

4.2.3.1. Ajustar el BIOS para latencia

Los BIOSes de PCs varían salvajemente en su comportamiento de latencia.

Ajustar el BIOS es tedioso porque tienes que reiniciar la computadora, hacer un pequeño ajuste en el BIOS, arrancar Linux, y ejecutar la prueba de latencia (potencialmente por mucho tiempo) para ver qué efecto tuvo tu cambio en el BIOS. Luego repetir para todas las otras configuraciones de BIOS que quieras intentar.

Ya que los BIOSes son todos diferentes y no estándar, no es práctico proporcionar una guía detallada de ajuste de BIOS. En general, algunas cosas a intentar ajustar en el BIOS son:

- Deshabilitar ACPI, APM y cualquier otra prestación de ahorro de energía. Esto incluye cualquier cosa relacionada con ahorro de energía, suspensión, estados de reposo de la CPU, escalamiento de frecuencia de la CPU, etc.
- Deshabilitar modo ”turbo” de la CPU.
- Deshabilitar hyper-threading de la CPU.
- Deshabilitar (o de otro modo, controlar) Interrupción de Gerencia de Sistema (SMI).
- Deshabilitar cualquier hardware que no pretendas usar.

4.2.3.2. Ajustar Preempt-RT para latencia

El kernel Preempt-RT puede beneficiarse del ajuste con tal de proporcionar la mejor latencia para LinuxCNC. Se puede realizar el ajuste por medio de la línea de comandos de kernel, sysctl, y por medio de archivos en `/proc` y `/sys`.

Algunos parámetros de ajuste a considerar:

Línea de comando de kernel
Detalles aquí: https://www.kernel.org/doc/Documentation/admin-guide/kernel-parameters.txt

- `isolcpus`: Previene que la mayoría de procesos que no son de LinuxCNC usen esas CPUs, dejando más tiempo de CPU disponible para LinuxCNC.
- `irqaffinity`: Selecciona cuáles CPUs sirven interrupciones, para que los CPUs reservados para tiempo real de LinuxCNC no tengan que realizar esta tarea.
- `rcu_nocbs`: Previene que las retrollamadas RCU se ejecuten en estas CPUs.
- `rcu_nocb_poll`: Encuesta retrollamadas de RCU en lugar de usar dormir/despertar.
- `nohz_full`: Desabilita el tictac del reloj en estas CPUs.

Sysctl
Detalles aquí: https://www.kernel.org/doc/html/latest/scheduler/sched-rt-group.html

- `sysctl.kernel.sched_rt_runtime_us`: Establece a -1 para eliminar el límite de cuánto tiempo pueden usar las tareas en tiempo real.

4.3. Afinación de generadores de pasos

4.3.1. Obtener el máximo rendimiento del software generador de pasos

Generar pulsos de paso en el software tiene una gran ventaja: es gratis. Casi todas las PC tienen un puerto paralelo que es capaz de generar pulsos de paso por el software. Sin embargo, los pulsos de paso de software también tienen algunas desventajas:

- velocidad de paso máxima limitada
- jitter (fluctuaciones) en los pulsos generados
- añade carga a la CPU

Este capítulo tiene algunos pasos que pueden ayudar a obtener los mejores resultados de pasos generados por software.

4.3.1.1. Ejecutar una prueba de latencia

La CPU no es el único factor determinante de latencia. Placas madre, tarjetas gráficas, puertos USB y muchas otras cosas pueden degradarla. La mejor forma de saber qué se puede esperar de una PC es ejecutar las pruebas de latencia RT.

Ejecuta la prueba de latencia como se describe en el capítulo Prueba de latencia.

Mientras se ejecuta la prueba, debes *abusar* de la computadora. Mueve ventanas alrededor de la pantalla. Navega por la web. Copia algunos archivos grandes en el disco. Pon música. Ejecuta un programa OpenGL como glxgears. La idea es poner a la PC a un fuerte ritmo mientras la prueba de latencia verifica para ver cuáles son los peores números.

El último número en la columna etiquetada *Max Jitter* es el más importante. Anótalo, lo necesitarás más tarde. Contiene la peor latencia medida durante toda la ejecución de la prueba. En el ejemplo anterior, eso es 6693 nanosegundos, o 6.69 microsegundos, que es excelente. Sin embargo, el ejemplo solo se ejecutó durante unos segundos (imprime una línea cada segundo). Debes ejecutar la prueba durante al menos varios minutos; a veces, la peor latencia no ocurre muy a menudo, o solo sucede cuando se hace alguna acción particular. Cierta placa base Intel funcionó bastante bien la mayor parte del tiempo, pero cada 64 segundos tenía una muy mala latencia de 300 µs. Afortunadamente, eso es corrregible, ver Solucionando problemas SMI en la wiki de LinuxCNC

Entonces, ¿qué significan los resultados? Si tu número *Max Jitter* es menor que aproximadamente 15-20 microsegundos (15000-20000 nanosegundos), la computadora debería dar muy buenos resultados con pasos por software. Si la latencia máxima está entre 30-50 microsegundos, todavía puedes obtener buenos resultados, pero tu velocidad máxima de paso puede ser algo decepcionante, especialmente si utilizas micropasos o si tienes tornillos de avance muy finos. Si los números son 100 µs o más (100,000 nanosegundos), entonces la PC no es buena candidata para pasos por software. Números de más de 1 milisegundo (1,000,000 nanosegundos) significa que la PC no es una buena candidata para LinuxCNC, independientemente de que uses pasos por software o no.

Ten en cuenta que si obtienes números altos, pueden haber formas de mejorarlos. Por ejemplo, una PC tenía una latencia muy mala (varios milisegundos) usando el vídeo integrado. Pero una tarjeta de vídeo usada de $5 resolvió el problema, LinuxCNC no requiere hardware de última generación.

4.3.1.2. Descubre lo que esperan tus controladores

Las diferentes marcas de controladores de pasos tienen diferentes requisitos de tiempos en sus pasos y entradas de dirección. Por lo que necesitas desenterrar (o Googlear) la hoja de datos que tiene las especificaciones de tu unidad.

Del manual de Gecko G202:

```
Frecuencia de pasos: 0 a 200 kHz
Tiempo de pulso ”0” de paso: 0.5 µs mínimo (Paso en el flanco descendente)
Tiempo de pulso ”1” de paso: 4.5 µs mínimo
Configuración de dirección: 1 µs mínimo (20 µs tiempo de espera mínimo después del ↩
    flanco de paso)
```

Del manual de Gecko G203V:

```
Frecuencia de paso: 0 a 333 kHz
Tiempo de pulso ”0” de paso: 2.0 µs mín. (paso en flanco ascendente)
Tiempo de pulso ”1” de paso: 1.0 µs mín.

Configuración de dirección:
    200 ns (0.2 µs) antes del flanco ascendente del pulso del paso
    200 ns (0.2 µs) de espera después del flanco ascendente del pulso de paso
```

De la hoja de datos de Xylotex:

```
Tiempo de configuración de DIR mínimo antes del flanco ascendente del pulso de  ↩
   STEP 200 ns mín.
Tiempo de espera DIR después del flanco ascendente del pulso de STEP 200 ns
Tiempo mínimo de pulso alto de STEP 2.0 µs
Tiempo mínimo de pulso bajo de STEP 1.0 µs
El paso ocurre en el flanco ascendente
```

Una vez encontrados los números, anotarlos también; se necesitarán en el próximo paso.

4.3.1.3. Elegir el BASE_PERIOD

BASE_PERIOD es el *latido* de la computadora LinuxCNC. En cada período, el software generador de pasos decide si es el momento de otro pulso de paso. Un período más corto permitirá generar más pulsos por segundo, dentro de los limites. Pero si se queda demasiado corto, la computadora pasará tanto tiempo generando pulsos de paso que todo lo demás se ralentizará a paso de tortuga, o tal vez incluso se bloqueará. La latencia y los requisitos del controlador afectan el período más corto que se puede usar, como veremos en un minuto.

Veamos primero el ejemplo de Gecko. El G202 puede manejar pulsos de pasos que están en bajo por 0.5 µs y en alto por 4.5 µs. Se necesita que el pin de dirección este estable durante 1 µs antes del flanco descendente y permanecer estable 20 µs después del flanco descendente. El requisito de tiempo más largo es el tiempo de espera de 20 µs. Un enfoque simple sería establecer el período en 20 µs. Eso significa que todos los cambios en las líneas STEP y DIR están separados por 20 µs. Todo se ve bien, ¿verdad?

¡Incorrecto! Si hubiera latencia CERO, entonces todos los flancos estarían separados por 20 µs, y todo estaría bien. Pero todas las computadoras tienen alguna latencia, que significa retraso. Si la computadora tiene 11 µs de latencia, eso significa que a veces el software corre 11 µs más tarde de lo que se supone. Si una ejecución del software se hace 11 µs tarde, y la siguiente se hace a tiempo, el retraso de la primera a la segunda es solo 9 µs. Si el primero generó un impulso de paso, y el segundo cambia el bit de dirección, se acaba de violar el requisito de tiempo de espera de 20 µs del G202. Eso significa que su unidad podría haber dado un paso en la dirección incorrecta, y su pieza tendrá el tamaño incorrecto.

La parte realmente desagradable de este problema es que puede ser muy esporádico. Las peores latencias pueden ocurrir solo unas pocas veces por minuto, y las probabilidades de que ocurra mala latencia justo cuando el motor está cambiando de dirección son bajas. Así que muy de vez en cuando ocurrirán errores que puedan arruinar una pieza, y son imposibles de localizar para solucionarlos.

La forma más sencilla de evitar este problema es elegir un BASE_PERIOD que sea la suma de los requisitos de tiempo más largos de su controlador, y el peor caso de latencia de la computadora. Si está ejecutando un Gecko con un tiempo de espera requerido de 20 µs, y la prueba de latencia dice que tiene una latencia máxima de 11 µs, si se configura BASE_PERIOD a 20+11=31 µs (31000 nanosegundos en el archivo ini), se tiene la garantía de cumplir los requisitos de tiempo del controlador.

Pero queda un compromiso. Hacer un pulso de paso requiere al menos dos períodos. Uno para comenzar el pulso y otro para finalizarlo. Como el período es 31 µs, se necesita 2x31 = 62 µs para crear un pulso de paso. Eso significa que la velocidad máxima de paso es de solo 16129 pasos por segundo. No muy buena. Pero no te rindas todavía, aún tenemos algunos ajustes que hacer en la próxima sección

Para Xylotex los tiempos de activación y espera son muy cortos, 200 ns cada uno (0.2 µs). El tiempo más largo es el alto de 2 µs. Si se tiene una latencia de 11 µs, entonces puede establecerse un BASE_PERIOD bajo como 11+2=13 µs. ¡Deshacerse del largo tiempo de espera de 20 µs realmente ayuda!. Con un período de 13 µs, un paso completo tarda 2x13 = 26 µs, y la velocidad máxima de paso es ¡38,461 pasos por segundo!

Pero no se puede celebrar todavía. Consideremos que 13 µs es un período muy corto. Si intentamos correr el generador de pasos cada 13 µs, podría no quedar tiempo suficiente para ejecutar cualquier otra cosa, y la computadora podría bloquearse. Si el objetivo son períodos menores a 25 µs, debemos comenzar con 25 µs o más, ejecutar LinuxCNC y ver cómo responden las cosas. Si todo esta bien, podemos disminuir gradualmente el período. Si el puntero del mouse comienza a ser lento, o cualquier otra cosa en la PC se ralentiza, el período es cortito. Regresa al valor anterior que dejó a la computadora funcionar sin problemas.

En este caso, supongamos que se comenzó en 25 µs, tratando de llegar a 13 µs, pero se encuentra que el límite esta alrededor de 16 µs o algo menor y la computadora no responde muy bien. Entonces usamos 16 µs. Con un período de 16 µs y latencia de 11 µs, el tiempo de salida más corto será 16-11 = 5 µs. El controlador solo necesita 2 µs, así que se tiene un margen. El margen es bueno, no se quiere perder pasos por haber usado un tiempo demasiado corto.

¿Cuál es la velocidad de paso máxima? Recuerda, dos períodos para dar un paso. Se estableció en 16 µs para el período, por lo que un paso consume 32 µs. Eso trabaja a unos nada malos 31,250 pasos por segundo.

4.3.1.4. Uso de steplen, stepspace, dirsetup y/o dirhold

En la última sección, obtuvimos para el controlador Xylotex un período de 16 µs y una velocidad máxima de 31250 pasos por segundo. Pero el Gecko estaba atascado en 31 µs y unos no tan buenos 16129 pasos por segundo. El ejemplo de Xylotex es lo mejor que podemos hacer con él. Pero el Gecko se puede mejorar.

El problema con el G202 es el requisito de tiempo de espera de 20 µs. Esto, y la latencia de 11 µs es lo que nos obliga a usar un período lento de 31 µs. Pero el generador de pasos por software de LinuxCNC tiene algunos parámetros que permiten aumentar diferentes tiempos en uno o varios períodos. Por ejemplo, si se cambia steplen de 1 a 2, entonces habrá dos períodos entre el comienzo y final del pulso de paso. Del mismo modo, si se cambia dirhold de 1 a 3, habrá al menos tres períodos entre el pulso de paso y un cambio del pin de dirección.

Si podemos usar dirhold para cumplir con el requisito de 20 µs de tiempo de espera, entonces el siguiente tiempo más largo es el tiempo alto de 4.5 µs. Al agregar la latencia de 11 µs al tiempo alto de 4.5 µs, se obtiene un período mínimo de 15.5 µs. Cuando se intenta con 15.5 µs, encontraremos que la computadora es lenta, por lo que se establecen 16 µs. Si dejamos dirhold en 1 (valor predeterminado), entonces el tiempo mínimo entre paso y dirección es el período de 16 µs menos la latencia de 11 µs = 5 µs, que no es suficiente, necesitamos otros 15 µs. Como el período es 16 µs, necesitamos un período más. Entonces cambiamos dirhold de 1 a 2. Ahora el tiempo mínimo desde el final del pulso de paso hasta el pin de cambio de dirección es 5+16 = 21 µs, y no tenemos que preocuparnos de que Gecko dé pasos en dirección equivocada debido a la latencia.

Si la computadora tiene una latencia de 11 µs, entonces una combinación de un período base de 16 µs, y un valor dirhold de 2 asegura que siempre cumpliremos los requisitos de tiempo del Gecko. Para

pasos normales (sin dirección cambio), el valor aumentado de dirhold no tiene efecto alguno. Son dos períodos que totalizan 32 µs para dar cada paso, y tenemos la misma velocidad de 31250 pasos por segundo que obtuvimos con el Xylotex.

El número de latencia de 11 µs utilizado en este ejemplo es muy bueno. Si trabajas a través de estos ejemplos con mayor latencia, como 20 o 25 µs, la mayor velocidad de pasos tanto para Xylotex como para Gecko será menor. Pero las mismas fórmulas aplican para calcular el BASE_PERIOD óptimo, y para ajustar dirhold u otros parámetros del generador de pasos.

4.3.1.5. ¡Sin adivinar!

Para un sistema de pasos basado en software, rápido y confiable, no se puede simplemente adivinar períodos y otros parámetros de configuración. Se necesitan mediciones en la computadora, y hacer los cálculos para asegurar que los controladores obtengan las señales que necesitan.

Para facilitar las matemáticas, hemos creado una hoja de cálculo de Open Office Calculadora de tiempos de pasos. Tú ingresas el resultado de la prueba de latencia y los requisitos de tiempo del controlador de pasos y la hoja de cálculo calcula el BASE_PERIOD óptimo. A continuación, prueba el período para asegurar que no se ralentizará o bloqueará la PC. Finalmente, ingresa el período real y la hoja de cálculo dirá la configuración de parámetros de stepgen que se necesita para cumplir con los requisitos de tiempo de la unidad. También se calcula la velocidad máxima de paso que se podrá generar.

Se han agregado algunas cosas a la hoja de cálculo para calcular la velocidad máxima y cálculos eléctricos de generador de pasos.

4.4. Configuración de INI

4.4.1. Componentes del archivo INI

Un archivo INI típico sigue un diseño bastante simple, que incluye;

- comentarios
- secciones
- variables

Cada uno de estos elementos está separado en líneas individuales. Cada final de línea o el carácter de nueva línea crea un nuevo elemento.

4.4.1.1. Comentarios

A comment line is started with a ; or a # mark. When the INI reader sees either of these marks on a line, the rest of the line is ignored by the software. Comments can be used to describe what an INI element will do.

```
; Este es mi archivo de configuración de fresadora.
# Configurado el 12 de enero de 2012
```

Los comentarios también se pueden usar para *desactivar* una variable. Esto hace más fácil elegir entre diferentes variables.

```
DISPLAY = axis
# DISPLAY = touchy
```

En esta lista, la variable DISPLAY se establecerá en Axis porque el otro valor está comentado. Si alguien edita descuidadamente una lista como esta y deja dos de las líneas sin comentar, se utilizará la primera encontrada.

Note that inside a variable's value, the "#" and ";" characters are still comments. You should use double or single quoted strings if you need to embed # or ; characters:

```
# Below results in INCORRECT=value
# because comments are stripped
INCORRECT = value # and a comment

# Correct embedding
CORRECT = "value # and an embedded # char"
```

4.4.1.2. Secciones

Related parts of an INI file are separated into sections. A section name is enclosed in brackets like this: `[THIS_SECTION]`. The order of sections is unimportant. Sections begin at the section name and end at the next section name. Section identifiers are case senstive and can only contain letters A-Z, a-z, digits 0-9 and underscore (_). Additionally, section identifiers cannot start with a digit.

Las siguientes secciones son utilizadas por LinuxCNC:

- [EMC] información general
- [DISPLAY] configuraciones relacionadas con la interfaz gráfica de usuario
- [FILTER] configuración de los programas de filtro de entrada
- [RS274NGC] configuración utilizada por el intérprete de código G
- [EMCMOT] configuraciones utilizadas por el controlador de movimiento en tiempo real
- [TASK] configuraciones utilizadas por el controlador de tareas
- [HAL] especifica archivos .hal
- [HALUI] comandos MDI utilizados por HALUI
- [APPLICATIONS] otras aplicaciones que iniciará LinuxCNC
- [TRAJ] configuraciones adicionales utilizadas por el controlador de movimiento en tiempo real
- [JOINT_n] variables de articulaciones individuales
- [AXIS_l] variables de eje individuales
- [KINS] variables de cinemáticas
- [EMCIO] configuración utilizada por el controlador de E/S

4.4.1.3. Variables

A variable line is made up of a variable name, an equals sign (=), and a value. A variable identifier is case sensitive and may only consist of letters A-Z, a-z, digits 0-9 and underscore (_). Additionally, variable identifiers cannot start with a digit. White space at the beginning of the line and after the variable identifier, up to the equals sign, is ignored.

Ejemplo de variable

```
MACHINE = Mi máquina
```

A variable line may be extended to multiple lines with a terminal backslash (\) character. There may be white space following the trailing backslash character, but this is strongly discouraged.

Los identificadores de sección no pueden extenderse a varias líneas.

Ejemplo de variable con línea extendida

```
APP = sim_pin \
ini.0.max_acceleration \
ini.1.max_acceleration \
ini.2.max_acceleration \
ini.0.max_velocity \
ini.1.max_velocity \
ini.2.max_velocity
```

A specific variable in a specific sections is often denoted in the documentation as [SECTION]VARIABLE. This specification mirrors the same way they are specified in HAL files for expansion.

Variable values may be quoted to ensure proper space and special character embedding in literal or escaped forms. Both single and double quoted values are allowed. Mutiple quoted value segments are merged into one value. Embedding quotes in quoted values must use the backslash \\ character to escape the embedded quote if it is the same kind as the enclosing quotes. Double quotes values also support all common escape formats and full Unicode:

- control: \\[abfnrtv]
- octal: \\[0-2][0-7]{0,2}
-
-
-

The resulting value is always converted into UTF-8 and checked for validity. It is not allowed to embed NUL characters either literally or by using an escape.

Value Quote and Escape Example

```
STRING = ”Hello World!”
STRING = ’Hello World Too!’

# The following would become: Hello World!
STRING = ”Hello” \
” ” \
’World’ \
”!”

EMBED = ”Literal single ’ in double”
EMBED = ’Literal double ” in single’
EMBED = ”Literal double \” in double”
EMBED = ’Literal single \’ in single’

SMILE = ”\\370\\237\\230\\200 = b’’□b’’”
SMILE = ”\\xf0\\x9f\\x98\\x80 = b’’□b’’”
SMILE = ”\\ud83d\\ude00 = b’’□b’’”
SMILE = ”\\U0001f600 = b’’□b’’”
```

Variables’ value can have types associated when they are read by LinuxCNC. The types are enforced by the INI file reader when the appropriate function calls are performed. All values may always be read as *string*. Conversion into other types has restrictions. Possible types are:

- *string* - the value is taken as-is

- *boolean* - only boolean words are accepted
- *signed integer* - whole numbers in decimal, hexadecimal, octal or binary bases
- *unsigned integer* - whole numbers in decimal, hexadecimal, octal or binary bases
- *real* - floating point values (always using decimal point)
- *enumeration* - a restricted set of keys to represent a value or function

Signed and unsigned integers are read and converted as 64-bit numbers when marked as `s64` and `u64`. Plain old 32-bit integers are marked `int` and are always signed. The default number base is decimal. Alternative bases may be specified using a prefix. The complete list of valid integer numbers:

- \[0-9]+ - decimal
- 0x\[0-9A-Fa-f]+ - hexadecimal
- 0o\[0-7]+ - octal
- 0b\[01]+ - binary

Signed integers may be preceded by a plus (+) or minus (-) sign, regardless number base. Unsigned integers will generate a warning if they are preceded by a minus (-) sign upon conversion.

Boolean values are case insensitive and allow the following words:

- true/enabled - `TRUE`, `YES` `ON` or `1`
- false/disabled - `FALSE`, `NO` `OFF` or `0`

Enumerations are a set of keywords defined by LinuxCNC and interpreted to mean a setting, value or functionality. The exact values are declared in the individual variable description and the variables are marked with `enum`. Most enumerations are interpreted without case. It is noted in the variable description if case matters.

Some LinuxCNC variables are special in that their format is interpreted as a multi-valued entry. These variables have an appropriate description below of the format expected.

Variables that are used by LinuxCNC must always use the section names and variable names as shown. Any custom or private variables and sections are up to the user.

4.4.1.4. Secciones personalizadas y variables

La mayoría de las configuraciones de muestra utilizan secciones y variables personalizadas para poner todas las configuraciones en una ubicación para mayor comodidad.

Para agregar una variable personalizada a una sección LinuxCNC existente, simplemente incluye la variable en esa sección.

Ejemplo de variable personalizada, asignando el valor LINEAR a la variable TYPE, y el valor 16000 a la variable SCALE.

```
[JOINT_0]
TYPE = LINEAR
...
SCALE = 16000
```

Para introducir una sección personalizada con sus propias variables, agregar la sección y variables al archivo INI.

Ejemplo de sección personalizada

```
[PROBE]
Z_FEEDRATE = 50
Z_OFFSET = 12
Z_SAFE_DISTANCE = -10
```

Para usar las variables personalizadas en tu archivo HAL, coloca la sección y el nombre de la variable en lugar del valor.

Ejemplo de HAL

```
setp offset.1.offset [PROBE]Z_OFFSET
setp stepgen.0.position-scale [JOINT_0]SCALE
```

---

nota
El valor almacenado en la variable debe coincidir con el tipo especificado por el pin del componente.

Para usar las variables personalizadas en el código G, usar la sintaxis de variable global `#<_ini[sección]var`
El siguiente ejemplo muestra una simple rutina touch-off del eje Z para una ruteadora o fresadora que utiliza una placa de sonda.

Please note that G-code embedded ini variables are converted to upper case before they are searched in the INI file. The `#<_ini[section]variable>` should be called [SECTION]VARIABLE in the INI file.

Ejemplo de código G

```
G91
G38.2 Z#<_ini[probe]z_safe_distance> F#<_ini[probe]z_feedrate>
G90
G1 Z#5063
G10 L20 P0 Z#<_ini[probe]z_offset>
```

4.4.1.5. Inclusión de archivos

Un archivo INI puede incluir el contenido de otro archivo usando una directiva #INCLUDE.

Formato #INCLUDE

```
#INCLUDE archivo
```

El nombre del archivo se puede especificar como:

- un archivo en el mismo directorio que el archivo INI
- un archivo relativo al directorio de trabajo
- un nombre de archivo absoluto (comienza con /)
- a user-home-relative file name (starts with a ~/)

Se admiten varias directivas #INCLUDE.

Ejemplos de #INCLUDE

```
#INCLUDE joint_0.inc
#INCLUDE ../parallel/joint_1.inc
#INCLUDE below/joint_2.inc
#INCLUDE /home/myusername/myincludes/display.inc
#INCLUDE ~/linuxcnc/myincludes/rs274ngc.inc
```

The #INCLUDE directives are supported up to 16 levels — an included file may include additional files up to 16 levels deep. Recursive inclusion of files is detected and flagged as an error. The recommended file extension is *.inc*. Do *not* use a file extension of *.ini* for included files.

4.4.2. Secciones del archivo INI

Variables in each section have an associated type as denoted in parentheses after the variable.

4.4.2.1. Sección [EMC]

- `VERSION = 1.1` - (string) The format version of this configuration. Any value other than 1.1 will cause the configuration checker to run and try to update the configuration to the new style joint axes type of configuration.
- `MACHINE = My Controller` - (string) This is the name of the controller, which is printed out at the top of most graphical interfaces. You can put whatever you want here as long as you make it a single line long.
- `DEBUG = 0` - (u64) Debug level 0 means no messages will be printed when LinuxCNC is run from a terminal. Debug flags are usually only useful to developers. See src/emc/nml_intf/debugflags.h for other settings.
- `RCS_DEBUG = 1` (u64) RCS debug messages to show. Print only errors (1) by default if EMC_DEBUG_RCS and EMC_DEBUG_RCS bits in `DEBUG` are unset, otherwise print all (-1). Use this to select RCS debug messages. See src/libnml/rcs/rcs_print.hh for all MODE flags.
- `RCS_DEBUG_DEST = STDOUT` - (enum) how to output RCS_DEBUG messages (NULL, STDOUT, STDERR, FILE, LOGGER, MSGBOX).
- `RCS_MAX_ERR = -1` - (int) Number after which RCS errors are not reported anymore (-1 = infinite).
- `NML_FILE = /usr/share/linuxcnc/linuxcnc.nml` - (string) Set this if you want to use a non-default NML configuration file.

4.4.2.2. Sección [DISPLAY]

Los diferentes programas de interfaz de usuario utilizan diferentes opciones, y no todas las opciónes son compatibles con todas las interfaces de usuario. Hay varias interfaces como AXIS, GMOCCAPY, Touchy, QtDragon y Gscreen de QtVCP. AXIS es una interfaz para usar con computadora y monitor normales. Touchy es para usar con pantallas táctiles. GMOCCAPY se puede usar en ambos tipos y también ofrece muchas conexiones para controles de hardware. Las descripciones de las interfaces se encuentran en la sección Interfaces del manual de usuario.

- `DISPLAY = axis` - (string) The file name of the executable providing the user interface to use. Prominent valid options are (all in lower case): `axis`, `touchy`, `gmoccapy`, `gscreen`, `tklinuxcnc`, `qtvcp`, `qtvcp qtdragon` or `qtvcp qtplasmac`.
- `POSITION_OFFSET = RELATIVE` - (enum) The coordinate system (`RELATIVE` or `MACHINE`) to show on the DRO when the user interface starts. The RELATIVE coordinate system reflects the G92 and G5*x* coordinate offsets currently in effect.
- `POSITION_FEEDBACK = COMMANDED` - (enum) The coordinate value (`COMMANDED` or `ACTUAL`) to show on the DRO when the user interface starts. In AXIS this can be changed from the View menu. The COMMANDED position is the position requested by LinuxCNC. The ACTUAL position is the feedback position of the motors if they have feedback like most servo systems. Typically the COMMANDED value is used.
- `DRO_FORMAT_MM =%+08.6f` - (string) Override the default DRO formatting in metric mode (normally 3 decimal places, padded with spaces to 6 digits to the left). The example above will pad with zeros, display 6 decimal digits and force display of a + sign for positive numbers. Formatting follows Python practice: https://docs.python.org/2/library/string.html#format-specification-mini-language . An error will be raised if the format can not accept a floating-point value.

- `DRO_FORMAT_IN =% 4.1f` - (string) Override the default DRO formatting in imperial mode (normally 4 decimal places, padded with spaces to 6 digits to the left). The example above will display only one decimal digit. Formatting follows Python practice: https://docs.python.org/2/library/-string.html#format-specification-mini-language . An error will be raised if the format can not accept a floating-point value.
- `CONE_BASESIZE = .25` - (real) Override the default cone/tool base size of .5 in the graphics display. Valid values are between 0.025 and 2.0.
- `DISABLE_CONE_SCALING = TRUE` - (bool) Any non-empty value (including "0") will override the default behavior of scaling the cone/tool size using the extents of the currently loaded G-code program in the graphics display.
- `MAX_FEED_OVERRIDE = 1.2` - (real) The maximum feed override the user may select. 1.2 means 120 % of the programmed feed rate.
- `MIN_SPINDLE_OVERRIDE = 0.5` - (real) The minimum spindle override the user may select. 0.5 means 50 % of the programmed spindle speed. (This is used to set the minimum spindle speed.)
- `MIN_SPINDLE_0_OVERRIDE = 0.5` - (real) The minimum spindle override the user may select. 0.5 means 50 % of the programmed spindle speed. (This is used to set the minimum spindle speed.) On multi spindle machine there will be entries for each spindle number. Only used by the QtVCP based user interfaces.
- `MAX_SPINDLE_OVERRIDE = 1.0` - (real) The maximum spindle override the user may select. 1.0 means 100 % of the programmed spindle speed.
- `MAX_SPINDLE_0_OVERRIDE = 1.0` - (real) The maximum feed override the user may select. 1.2 means 120 % of the programmed feed rate. On multi spindle machine there will be entries for each spindle number. Only used by the QtVCP based user interfaces.
- `DEFAULT_SPINDLE_SPEED = 100` - Las RPM predeterminadas del husillo cuando se inicia en modo manual. Si esta configuración no está presente, el valor predeterminado es 1 RPM para AXIS y 300 RPM para GMOCCAPY.
  - *obsoleta* - Usar en cambio la sección [SPINDLE_n]
- `DEFAULT_SPINDLE_0_SPEED = 100` - Las RPM predeterminadas del husillo en modo manual. En una máquina con múltiples husillos habrá entradas para cada número de husillo. Solo usado por interfases de usuario basadas en QtVCP.
  - *obsoleta* - Usar en cambio la sección [SPINDLE_n].
- `SPINDLE_INCREMENT = 200` - Incremento utilizado al hacer clic en los botones de aumento/disminución. Utilizada sólo por las interfaces basadas en QtVCP.
  - *obsoleta* - Usar en cambio la sección [SPINDLE_n].
- `MIN_SPINDLE_0_SPEED = 1000` - Las RPM mínimas que pueden seleccionarse manualmente. En máquinas con múltiples husillo habrá entradas para cada número de husillo. Usada sólo por interfaces de usuario basadas en QtVCP.
  - *obsoleta* - Usar en cambio la sección [SPINDLE_n].
- `MAX_SPINDLE_0_SPEED = 20000` - Las RPM máximas que se pueden seleccionar manualmente. En una máquina con múltiples husillos habrá entradas para cada número de husillo. Solo usada por interfaces de usuario basadas en QtVCP.
  - *obsoleta* - Usar en cambio la sección [SPINDLE_n].

- PROGRAM_PREFIX = ~/linuxcnc/nc_files - La ubicación predeterminada para archivos de código G, subrutinas nombradas y códigos M definidos por el usuario. Se busca en el directorio PROGRAM_PREFIX antes que en los directorios listados en [RS274]SUBROUTINE_PATH y [RS274]USER_M_PATH.
- INTRO_GRAPHIC = emc2.gif - La imagen que se muestra en la pantalla de inicio.
- INTRO_TIME = 5 - El tiempo máximo a mostrar la pantalla de inicio, en segundos.
- CYCLE_TIME = 0.05 - Tiempo de ciclo de despliegue de la GUI. Dependiendo de la pantalla, puede ser un segundos o ms (de preferencia ms). A menudo es la tasa de actualización en vez de el tiempo de espera entre actualizaciones. Si no se configura correctamente el tiempo de actualización, puede que la pantalla no responda o se vea entrecortada. Un valor de 100 ms (0.1 s) es una configuración común, aunque en un rango entre 50 - 200 ms (.05 - .2 s) puede ser usable. Un CPU con poca energía puede ver mejoras con un valor más alto. Normalmente el valor predeterminado esta bien.
- PREVIEW_TIMEOUT = 5 - Tiempo de espera (en segundos) para cargar la previsualización gráfica de código G. Actualmente, sólo AXIS.
- HOMING_PROMPT = TRUE - Any non-empty value (including ”0”) will enable showing a prompt message with homing request, when the Power On button is pressed in AXIS GUI. Pressing the ”Ok” button in prompt message is equivalent to pressing the ”Home All” button(or the Ctrl-HOME key).
- FOAM_W = 1.5 - Establece la altura de espuma W.
- FOAM_Z = 0 - Establece la altura de espuma Z.
- GRAPHICAL_MAX_FILE_SIZE = 20 - El tamaño más grande (en mega bytes) que se mostrará gráficamente. Si el programa es más grande que esto, se mostrará una caja delimitada. Predeterminadamente, ésta configuración está a 20 MB o 1/4 de la memoria del sistema, lo que sea más bajo. Un valor negativo se interpreta como ilimitado.

---

nota
GladeVCP y PyVCP utilizan los siguientes elementos [DISPLAY], consulte la sección Incrustando una pestaña del capítulo GladeVCP o el Capítulo PyVCP para más información.

- EMBED_TAB_NAME = GladeVCP demo
- EMBED_TAB_COMMAND = halcmd loadusr -Wn gladevcp gladevcp -c gladevcp -x {XID\} -u ./glad
  ./gladevcp/manual-example.ui

---

nota
Los diferentes programas de interfaz de usuario utilizan diferentes opciones, y no todas las opciones son compatibles con todas las interfaces de usuario. Consulte el documento GUI AXIS para obtener detalles sobre AXIS. Consulte el documento GMOCCAPY para obtener detalles sobre GMOCCAPY.

- DEFAULT_LINEAR_VELOCITY = .25 - La velocidad predeterminada para trotes lineales, en unidades máquina por segundo.
- MIN_VELOCITY = .01 - El valor más bajo aproximado del control deslizante de trote.
- MAX_LINEAR_VELOCITY = 1.0 - La velocidad máxima para trotes lineales, en unidades de máquina por segundo.
- MIN_LINEAR_VELOCITY = .01 - El valor más bajo aproximado del deslizante de trote.
- DEFAULT_ANGULAR_VELOCITY = .25 - La velocidad predeterminada para trotes angulares, en unidades máquina por segundo.

- `MIN_ANGULAR_VELOCITY = .01` - El valor más bajo aproximado del deslizante de trote angular.
- `MAX_ANGULAR_VELOCITY = 1.0` - La velocidad máxima para trotes angulares, en unidades de máquina por segundo.
- `INCREMENTS = 1 mm, .5 in, ...` - Define los incrementos disponibles para trotes incrementales. Los INCREMENTS se pueden usar para ajustar los valores predeterminados. Los valores pueden ser números decimales (p. ej. 0.1000) o números fraccionarios (p. ej. 1/16), opcionalmente seguidos por una unidad (cm, mm, um, inch, in o mil). Si no se especifica una unidad, se supone la unidad de máquina. Las distancias métricas e imperiales se pueden mezclar: INCREMENTS = 1 inch, 1 mil, 1 cm, 1 mm, 1 um es una entrada válida.
- `GRIDS = 10 mm, 1 in, ...` - Define los valores preestablecidos para las líneas de cuadrícula. El valor se interpreta de la misma manera que `INCREMENTS`.
- `OPEN_FILE = /ruta/completa/a/archivo.ngc` - El archivo que se mostrará en la gráfica de vista previa cuando se inicie AXIS. Una cadena en blanco "" no cargará ningún archivo en el arranque. GMOCCAPY no usará esta configuración, ya que ofrece una entrada correspondiente en su página de configuración.
- `EDITOR = gedit` - El editor que se usará al seleccionar Archivo> Editar para editar código G desde el menú de AXIS. Esto debe configurarse para que este elemento de menú trabaje. Otra entrada válida es `gnome-terminal -e vim`. Esta entrada no se aplica a GMOCCAPY ya que esta GUI tiene un editor integrado.
- `TOOL_EDITOR = tooledit` - El editor que se utilizará al editar la tabla de herramientas (por ejemplo, al seleccionar "Archivo > Editar datos de herramienta ..." en AXIS). Otras entradas validas son `gedit`, `gnome-terminal -e vim` y `gvim`. Esta entrada no se aplica a GMOCCAPY ya que esta GUI tiene un editor integrado.
- `PYVCP = /archivo.xml` - El archivo de descripción del panel PyVCP. Ver el << cha:pyvcp,capítulo PyVCP >> para más información.
- `PYVCP_POSITION = BOTTOM` - La ubicación del panel PyVCP en la interfaz de usuario AXIS. Si se omite esta variable, el panel pasará por defecto al lado derecho. La única alternativa valida es `BOTTOM`. Ver el << cha:pyvcp,Capítulo PyVCP >> para más información.
- `LATHE = 1` - Cualquier valor no vacío (incluido "0") hace que Axis utilice el "modo torno" con una vista superior y con Radio y Diámetro en el DRO.
- `BACK_TOOL_LATHE = 1` - Cualquier valor no vacío (incluido "0") hace que Axis utilice el "modo torno de herramienta trasera" con el eje X invertido.
- `FOAM = 1` - Cualquier valor no vacío (incluido "0") hace que Axis cambie la visualización para el modo cortador por espuma.
- `GEOMETRY = XYZABCUVW` - Controla la vista **previa** y el **backplot** de movimiento giratorio. Este elemento consiste en una secuencia de letras de eje, opcionalmente precedidas por un signo "-":

  1. Las letras X,Y,Z especifican la traslación a lo largo de la coordenada nombrada.
  2. Las letras A,B,C especifican rotación alrededor de los correspondientes ejes X, Y, Z.
  3. Las letras U,V,W especifican traslación a lo largo de los ejes relacionados X, Y, Z.
  4. Cada letra especificada debe ocurrir en `[TRAJ]COORDINATES` para que tenga efecto.
  5. Un caracter especial "**-**" precediendo a cualquier letra invierte la dirección de la operación.
  6. Las operaciones de traslación y rotación se evalúan de **derecha a izquierda**. De tal manera que usar `GEOMETRY=ZYZBC` especifica una rotación C seguida de una rotación B seguida por traslaciones Z, Y, X. El ordenamiento consecutivo de letras de traslación es inmaterial.
  7. La cadena apropiada de GEOMETRY depende de la configuración de la máquina y de las cinemáticas usadas para controlarla. El orden de las letras es importante. Por ejemplo, rotar alrededor de C y luego B es diferente que rotar alrededor de B y luego C.

8. Las rotaciones se aplican predeterminadamente con respecto al origen de máquina. Ejemplo: `GEOMETRY=CXYZ` primero traslada al punto de control X, Y, Z y luego realiza una rotación C alrededor del eje Z centrado en el origen de máquina.
9. Ejemplo de traslación UVW: `GEOMETRY=XYZUVW` provoca mover UVW en el sistema de coordenadas de la herramienta y mover XYZ en el sistema de coordenadas del material.
10. Las máquinas de corte de espuma (`FOAM = 1`) deben especificar ”XY;UV” o dejar el valor en blanco aunque este valor se ignore actualmente en el modo de cortador de espuma. Una versión futura puede definir qué significa ”;”, pero si lo hace ”XY;UV” significará lo mismo que el predeterminado para espuma actual.
11. Experimental: Si se incluye el signo de exclamación (`!`) en la cadena GEOMETRY, muestra los puntos para las rotaciones A, B, C respecto a los offsets X, Y, Z establecidos por los códigos G5x, G92. Ejemplo: Usando `GEOMETRY = !CXZ` para una máquina con `[TRAJ]COORDINATES=XZC`. Esta provisión aplica sólo para gráficos en vivo (previsualizaciones de código G deben hacerse con offsets G5x, G92 cero. Esto se puede facilitar configurando offsets en programas sólo cuando la tarea se ejecuta como se indica en #<_task> == 1. Si existen offsets distintos de cero en el arranque por persistencia, se debe poner en cero los offsets y recargar la vista previa.

---

nota
Si no se incluye `[DISPLAY]GEOMETRY` en el archivo INI, un predeterminado es proporcionado por el programa GUI `[DISPLAY]DISPLAY` (típicamente ”XYZABCUVW”).

- `ARCDIVISION = 64` - Establece la calidad de la vista previa de los arcos. Los arcos se previsualizan dividiéndolos en una serie de líneas rectas; un semicírculo se divide en **ARCDIVISIÓN** partes. Los valores más grandes dan una vista previa más precisa, pero tardan más tiempo en cargar y dan como resultado una pantalla más lenta. Los valores más pequeños dan una vista previa menos precisa, pero tarda menos tiempo en cargar y puede resultar en un despliegue más rápido. El valor predeterminado de 64 significa que un círculo de hasta 3 pulgadas se mostrará con precision de 1 mil (.03 %).
- `MDI_HISTORY_FILE =` - El nombre del archivo de historial MDI local. Si no se especifica, AXIS guardará el historial MDI en **.axis_mdi_history** en el directorio de usuario. Esto es útil si tiene múltiples configuraciones en una computadora.
- `JOG_AXES =` - El orden en que se asignan las teclas de trote a las letras del eje. Las flechas izquierda y derecha se asignan a la letra del primer eje, arriba y abajo a la segunda, página arriba/página abajo a la tercera, y corchetes izquierdo y derecho a la cuarta. Si no se especifica, el valor predeterminado se determina a partir de los valores de `[TRAJ]COORDINATES`, `[DISPLAY]LATHE` y `[DISPLAY]FOAM`.
- `JOG_INVERT =` - Para cada letra de eje, se invierte la dirección del trote. El valor predeterminado es ”X” para tornos y en blanco en el resto.

---

nota
La configuración de `JOG_AXES` y `JOG_INVERT` se aplican al trote modo universal por letra de eje de coordenadas y están vigentes mientras se encuentra en modo universal después de un homing exitoso. Cuando se opera en modo articulación anterior al homing, las teclas de trote del teclado se asignan en una secuencia fija: izquierda/derecha: joint0, arriba/abajo: joint1, página arriba/ página abajo: joint2, corchete izquierdo/derecho: joint3

- `USER_COMMAND_FILE = mycommands.py` - El nombre de un archivo Python opcional, específico de configuración originado por la GUI AXIS en lugar del archivo específico del usuario `~/.axisrc`.

---

nota
El siguiente elemento `[DISPLAY]` es utilizado únicamente por la interfaz TKLinuxCNC.

- `HELP_FILE = tklinucnc.txt` - Ruta al archivo de ayuda.

4.4.2.3. Sección [FILTER]

AXIS y GMOCCAPY tienen la capacidad de enviar archivos cargados a través de un programa de filtro. Este filtro puede hacer cualquier tarea deseada: algo tan simple como asegurarse el archivo termina con M2, o algo tan complicado como detectar si la entrada es una imagen de profundidad y generar código G para fresar la forma definida. La sección `[FILTER]` del archivo INI controla cómo trabajan los filtros. Primero, para cada tipo de archivo, escribe una línea `PROGRAM_EXTENSION`. Luego, especifica el programa a ejecutar para cada tipo de archivo. Este programa recibe como primer argumento el nombre del archivo de entrada, y debe escribir código RS274NGC en la salida estándar. Esta salida es lo que se mostrará en el área de texto, se previsualizará en el área de pantalla y será ejecutado por LinuxCNC cuando se ordene.

- `PROGRAM_EXTENSION = .extensión Descripción`

Si su postprocesador genera archivos en mayúsculas, es posible que desee agregar la siguiente linea:

```
PROGRAM_EXTENSION = .NGC Postprocesador XYZ
```

Las siguientes líneas agregan soporte para el convertidor de imagen a código G incluido con LinuxCNC.

```
PROGRAM_EXTENSION = .png,.gif,.jpg # Imagen de profundidad de escala de grises
  png = image-to-gcode
  gif = image-to-gcode
  jpg = image-to-gcode
```

Un ejemplo de un convertidor de código G personalizado ubicado en el directorio linuxcnc.

```
PROGRAM_EXTENSION = .gcode 3D Printer
  gcode = /home/mill/linuxcnc/convert.py
```

---

nota
El archivo de programa asociado con una extensión debe tener la ruta completa al programa o estar ubicado en un directorio que se encuentre en la ruta del sistema.

También es posible especificar un intérprete:

```
PROGRAM_EXTENSION = .py Python Script
  py = python
```

De esta manera se puede abrir cualquier script de Python y su salida es tratada como código G. Un script de ejemplo de este tipo está disponible en nc_files/holecircle.py. Este script crea código G para perforar una serie de agujeros a lo largo de una circunferencia. Hay muchos generadores más de código G en el sitio Wiki de LinuxCNC https://wiki.linuxcnc.org/.

Los filtros de Python deben usar la función de impresión para enviar el resultado a AXIS.

Este programa de ejemplo filtra un archivo y agrega un eje W para que coincida con el eje Z. Depende de que haya un espacio entre cada palabra de eje para que funcione.

```
#!/usr/bin/env python3

import sys

def main(argv):

  openfile = open(argv[0], 'r')
  file_in = openfile.readlines()
```

```
  openfile.close()

  file_out = []
  for line in file_in:
    # print(line)
    if line.find('Z') != -1:
      words = line.rstrip('\n')
      words = words.split(' ')
      newword = ''
      for i in words:
        if i[0] == 'Z':
          newword = 'W'+ i[1:]
      if len(newword) > 0:
        words.append(newword)
        newline = ' '.join(words)
        file_out.append(newline)
    else:
      file_out.append(line)
  for item in file_out:
    print("%s" % item)

if __name__ == "__main__":
  main(sys.argv[1:])
```

- `FILTER_PROGRESS=%d`
  Si se establece la variable de entorno AXIS_PROGRESS_BAR, entonces las líneas escritas a stderr de la forma de arriba establecen la barra de progreso de AXIS en el porcentaje dado. Esta característica debería ser utilizada por cualquier filtro que se ejecute por un tiempo prolongado.

4.4.2.4. Sección [RS274NGC]

- `PARAMETER_FILE = myfile.var` - El archivo ubicado en el mismo directorio que el archivo INI que contiene los parámetros utilizados por el intérprete (guardado entre ejecuciones).
- `ORIENT_OFFSET = 0` - Un valor float agregado al parámetro R de una operación M19 Orientar husillo. Se usa para definir una posición cero arbitraria independientemente de la orientación de montaje del codificador.
- `RS274NGC_STARTUP_CODE = G17 G20 G40 G49 G64 P0.001 G80 G90 G92 G94 G97 G98` - Una cadena de códigos NC que con la que se inicializa el intérprete. Esto no es un sustituto para especificar códigos G modales en la parte superior de cada archivo NGC, porque los códigos modales de las máquinas difieren, y pueden ser cambiados por el código G interpretado anteriormente en la sesión.
- `SUBROUTINE_PATH = ncsubroutines:/tmp/testsubs:lathesubs:millsubs` - Especifica una lista separada por dos puntos (:) de hasta 10 directorios a buscar cuando se especifican subrutinas de un solo archivo en código G. Se busca en estos directorios después de buscar `[DISPLAY] PROGRAM_PREFIX` (si está especificado) y antes de buscar `[WIZARD]WIZARD_ROOT` (si se especifica). Se busca en las rutas en el orden que están listadas. Se usa el primer archivo de subrutina coincidente encontrado en la búsqueda. Los directorios se especifican en relación con el directorio actual para el archivo INI o como rutas absolutas. La lista no debe contener espacios en blanco intermedios.
- `G64_DEFAULT_TOLERANCE` = *n* (Predeterminado: 0) Valor P predeterminado para G64 si P no es llamado.
- `G64_DEFAULT_NAIVETOLERANCE` = *n* (Predeterminado: 0) Valor Q predeterminado para G64 si Q no es llamado.
- `CENTER_ARC_RADIUS_TOLERANCE_INCH` = *n* (Predeterminado 0.00005)

- CENTER_ARC_RADIUS_TOLERANCE_MM = *n* (Predeterminado 0.00127)
- USER_M_PATH = myfuncs:/tmp/mcodes:experimentalmcodes - Especifica una lista de directorios separados por dos puntos (:) para funciones definidas por el usuario. Los directorios se especifican relativos al directorio actual del archivo INI o como rutas absolutas. La lista no debe contener espacios en blanco intermedios.

  Se realiza una búsqueda para cada posible función definida por el usuario, típicamente (M100-M199). El orden de búsqueda es:

  1. [DISPLAY]PROGRAM_PREFIX (si se especifica)
  2. Si no se especifica [DISPLAY]PROGRAM_PREFIX, busca en la ubicación predeterminada: nc_files
  3. Luego busca en cada directorio de la lista [RS274NGC]USER_M_PATH.
     El primer ejecutable M1*xx* encontrado en la búsqueda se usa para cada M1*xx*.

  nota
  El número máximo de directorios USER_M_PATH se define en tiempo de compilación (typ: USER_DEFINED_FUNCTION_MAX_DIRS == 5).

- INI_VARS = 1 (Predeterminado: 1)
  Permite que los programas de código G lean valores del archivo INI usando el formato #<_ini[sección]nom
  Ver parámetros del código G.
- HAL_PIN_VARS = 1 (Predeterminado: 1)
  Permite que los programas de código G lean los valores de los pines HAL usando el formato #<_hal[elemen
  HAL]>. El acceso a esta variable es de solo lectura. Consultar parámetros de código G para obtener más detalles y una advertencia importante.
- RETAIN_G43 = 0 (Predeterminado: 0)
  Cuando está configurado, puede activar G43 después de cargar la primera herramienta, y luego despreocuparse de eso durante el transcurso del programa. Cuando usted finalmente descargue la última herramienta, el modo G43 es cancelado.
- OWORD_NARGS = 0 (Predeterminado 0)
  Si esta función está habilitada, una subrutina llamada puede determinar el número de parámetros posicionales reales pasados al inspeccionar el parámetro #<n_args>.
- NO_DOWNCASE_OWORD = 0 (Predeterminado: 0)
  Conservar mayúsculas y minúsculas en los nombres de palabra O dentro de los comentarios si está configurado, permite leer elementos HAL en mayúsculas y minúsculas en comentarios estructurados como (debug, #<_hal[ElementoConMayMin]).
- OWORD_WARNONLY = 0 (Predeterminado: 0)
  Advertir en lugar de error en caso de errores en las subrutinas O-word.
- DISABLE_G92_PERSISTENCE = 0 (Predeterminado: 0) Permite limpiar el offset G92 automáticamente al iniciar la configuración.
- DISABLE_FANUC_STYLE_SUB = 0 (Predeterminado: 0) Si hay alguna razón para desactivar subrutinas Fanuc asignarle 1.
- *G73_PECK_CLEARANCE = .020* (default: Metric machine: 1mm, imperial machine: .050 inches) Chip breaking back-off distance in machine units
- *G83_PECK_CLEARANCE = .020* (default: Metric machine: 1mm, imperial machine: .050 inches) Clearance distance from last feed depth when machine rapids back to bottom of hole, in machine units.

nota
Las seis opciones anteriores fueron controladas por la máscara de bits FEATURES en versiones de LinuxCNC anteriores a 2.8. Esta etiqueta INI ya no trabaja.
Para referencia:

```
FEATURES & 0x1  -> RETAIN_G43
FEATURES & 0x2  -> OWORD_NARGS
FEATURES & 0x4  -> INI_VARS
FEATURES & 0x8  -> HAL_PIN_VARS
FEATURES & 0x10 -> NO_DOWNCASE_OWORD
FEATURES & 0x20 -> OWORD_WARNONLY
```

nota
`[WIZARD]WIZARD_ROOT` es una ruta de búsqueda válida, pero el asistente no se ha implementado por completo y los resultados de su uso son impredecibles.

- `LOG_LEVEL = 0` - Especifica el nivel de registro en bitácora (predeterminado: 0)
- `LOG_FILE = file-name.log`
  Para especificar el archivo a usar para registrar datos de bitácora.
- `REMAP=M400 modalgroup=10 argspec=Pq ngc=myprocedure` - Ver el capítulo Extensión de código G por remapeo para más detalles.
- `ON_ABORT_COMMAND=O <on_abort> call`- Ver el capítulo Extensión de código G por remapeo para más detalles.

4.4.2.5. Sección [EMCMOT]

Esta es una sección personalizada y LinuxCNC no la utiliza directamente. Muchas configuraciones utilizan valores de esta sección para cargar el controlador de movimiento. Para obtener más información sobre el controlador de movimiento, ver la sección Movimiento.

- `EMCMOT = motmod` - El nombre del controlador de movimiento generalmente se usa aquí.
- `BASE_PERIOD = 50000` - El período de tarea *Base* en nanosegundos.
- `SERVO_PERIOD = 1000000` - Este es el período de tarea ”Servo” en nanosegundos.
- `TRAJ_PERIOD = 100000` - Este es el período de la tarea *Planificador de trayectoria* en nanosegundos.
- `COMM_TIMEOUT = 1.0` - Número de segundos para esperar a Motion (la parte en tiempo real del controlador de movimiento) para acusar recibo de mensajes desde Task (la parte en tiempo no-real del controlador de movimiento).
- `HOMEMOD` = *alternate_homing_module* [home_parms=value] La variable HOMEMOD es opcional. Si se especifica, usa un módulo (construido por el usuario) en lugar del predeterminado (homemod). Los parámetros del módulo (home_parms) pueden incluirse si los soporta el módulo especificado. La configuración puede ser anulada desde la línea de comandos usando la opción -m ($ linuxcnc -h).

4.4.2.6. Sección [TASK]

- `TASK = milltask` - Especifica el nombre de la *tarea* ejecutable. La *tarea* ejecutable hace varias cosas, como

- Comunicar con la interfaz de usuario por medio de NML,
- comunicar con el planificador de movimiento en tiempo real por medio de memoria compartida no-HAL, y
- interpretar código G. Actualmente sólo hay una tarea ejecutable que tiene sentido para el 99.99 % de los usuarios: milltask.

- `CYCLE_TIME = 0.010` - The period, in seconds, at which TASK will run. This parameter affects the polling interval when waiting for motion to complete, when executing a pause instruction, and when accepting a command from a user interface. There is usually no need to change this number. Defaults to 0.100 if omitted.

4.4.2.7. Sección [HAL]

- `HALFILE = example.hal` - Ejecuta el archivo *example.hal* al arranque.

  Si se especifica `HALFILE` varias veces, los archivos se interpretan en el orden en que aparecen en el archivo INI. Los archivos HAL son descriptivos, la ejecución de lo descrito en el archivo se dispara por hilos a los que se les cuelgan funciones, no al leer el archivo HAL. Casi todas las configuraciones tendrán al menos un `HALFILE`, y los sistemas paso a paso suelen tener dos de estos archivos, uno que especifica la configuración paso a paso genérica (*core_stepper.hal*) y uno que especifica los pines de la máquina (*xxx_pinout.hal*).

  Los archivos HAL especificados en la variable `HALFILES` se encuentran con una búsqueda. Si el archivo nombrado se encuentra en el directorio que contiene el archivo INI, se utiliza. Si el archivo nombrado no se encuentra en este directorio de archivos INI, se realiza una búsqueda utilizando una biblioteca de sistema de archivos HAL.

  Si se arranca LinuxCNC con el script `linuxcnc` usando la opción "`-H _nombredirectorio_`", se prepone el nombre de directorio especificado a la búsqueda descrita arriba, de tal manera que se busca primero en _nombredirectorio_. La opción "-H *nombredirectorio*" puede ser especificada mas de una vez, los directorios se preponen en orden.

  Un HALFILE también se puede especificar como una ruta absoluta (cuando el nombre comienza con un caracter /). Las rutas absolutas no son recomendables ya que su uso puede limitar la reubicación de configuraciones.

- `HALFILE = texample.tcl` [*arg1* [*arg2*] ...] - Ejecuta el archivo tcl *texample.tcl* al inicio con arg1, arg2, etc. como lista argv. Los archivos con un sufijo .tcl son procesados como se indica arriba, pero son procesados por haltcl. Ver el capitulo HALTCL para más información.

- `HALFILE = LIB:sys_example.hal` - Ejecuta el archivo de la biblioteca de sistema *sys_example.hal* al inicio. El uso explícito del prefijo LIB: provoca el uso de la biblioteca del sistema HALFILE sin buscar en el directorio de archivos INI.

- `HALFILE = LIB:sys_texample.tcl` [*arg1* [*arg2* ...]] - Ejecuta el archivo de biblioteca de sistema *sys_texample.tcl* al inicio. El uso explícito del prefijo LIB: provoca el uso de la biblioteca del sistema HALFILE sin buscar en el directorio de archivos INI.

HALFILE items specify files that loadrt HAL components and make signal connections between component pins. Common mistakes are:

1. omisión de la sentencia addf necesaria para agregar funciones de componente a un hilo,
2. especificación incompleta de señal (red).

La omisión de las sentencias addf requeridas es casi siempre es un error. Las señales generalmente incluyen una o más conexiones de entrada y una sola salida (pero ambas no son estrictamente necesarias). Se proporciona un archivo de biblioteca de sistema para verificar estas condiciones e informar a stdout y en una ventana emergente de GUI:

```
HALFILE = LIB:halcheck.tcl [nopopup]
```

nota
La línea LIB:halcheck.tcl debería ser la última [HAL]HALFILE. Especificar la opción *nopopup* para suprimir el mensaje emergente y permitir el inicio inmediato. Las conexiones realizadas con un POSTGUI_HALFILE no serán verificadas.

- `TWOPASS = ON` - Utilice el procesamiento de dos pasos para cargar componentes HAL. Con el procesamiento TWOPASS, las líneas `[HAL]HALFILE` se procesan en dos pasadas. En el primer pase (pass0), se leen todos los HALFILES y se acumulan múltiples apariciones de comandos loadrt y loadusr. Estos comandos de carga acumulados se ejecutan al final de pass0. Esta acumulación permite especificar más de una vez líneas de carga para un componente dado (siempre que los nombres names= utilizados sean únicos en cada uso). En el segundo pase (pass1) los HALFILES son releídos y todos los comandos, excepto los comandos de carga ejecutados previamente, son ejecutados.
- `TWOPASS = nodelete verbose` - La funcionalidad `TWOPASS` se puede activar con cualquier cadena no nula que incluya las palabras clave verbose y nodelete. La palabra clave verbose provoca la impresión de detalles en la salida estándar. La palabra clave nodelete preserva archivos temporales en /tmp.

Para obtener más información, ver el capítulo HAL TWOPASS.

- `HALCMD = _comando_` - Ejecuta *comando* como un solo comando HAL. Si se especifica `HALCMD` varias veces, los comandos se ejecutan en el orden en que aparecen en el archivo INI. Las líneas `HALCMD` se ejecutan después de todas las líneas `HALFILE`.
- `SHUTDOWN = shutdown.hal` - Ejecuta el archivo *shutdown.hal* cuando se sale de LinuxCNC. Dependiendo de los controladores de hardware utilizados, esto puede permitir configurar salidas a valores definidos cuando LinuxCNC sale normalmente. Sin embargo, ya que no se garantiza que este archivo será ejecutado (por ejemplo, en el caso de un bloqueo de la computadora) no es un reemplazo para una cadena de parada de emergencia adecuada u otras protecciones contra fallos de software.
- `POSTGUI_HALFILE = example2.hal` - Ejecuta *example2.hal* después de que la GUI haya creado sus pines HAL. Algunas GUI crean pines HAL y admiten el uso de un archivo HAL postgui para usarlos. Las GUI que admiten archivos HAL postgui incluyen Touchy, AXIS, Gscreen y GMOCCAPPY.
  Ver la sección PyVCP con AXIS para más información.
- `HALUI = halui` - Agrega los pines de la interfaz de usuario de HAL.
  Para mas información, ver el capítulo Interfaz de usuario HAL.

4.4.2.8. Sección [HALUI]

- `MDI_COMMAND = G53 G0 X0 Y0 Z0` - Se puede ejecutar un comando MDI utilizando `halui.mdi-command-0`
  Incremente el número para cada comando que se enumera en la sección [HALUI]. También es posible iniciar subrutinas. `MDI_COMMAND = o<tu-subrutina> CALL [#<tu-variable>]`

4.4.2.9. Sección [APPLICATIONS]

LinuxCNC puede iniciar otras aplicaciones antes de que se inicie la GUI especificada. Las aplicaciones se pueden iniciar después de un retraso especificado para permitir acciones dependientes de la GUI (como crear pines HAL específicos de GUI).

- `DELAY = _valor_` - Segundos de espera antes de comenzar otras aplicaciones. Puede ser necesario un retraso si una aplicación tiene dependencias en acciones `[HAL]POSTGUI_HALFILE` o pines HAL creados por GUI (predeterminado `DELAY= 0`).
- 'APP =` *appname* [arg1 [arg2 ...]]' - Aplicación que se iniciará. Esta especificación se puede incluir varias veces. El nombre de la aplicación puede ser dado explícitamente como un nombre de archivo especificado absoluto o especificado con tilde (primer carácter es / o ~), un nombre de archivo relativo (los primeros caracteres del nombre de archivo son ./), o como un archivo en el directorio INI. Si no se encuentra ningún archivo ejecutable usando estos nombres, se utiliza la ruta de búsqueda del usuario para encontrar la aplicación.
  Ejemplos:
  - Simular las entradas a los pines HAL para prueba (usando sim_pin, una GUI simple para configurar entradas a parámetros, pines desconectados o señales sin escritores):

    ```
    APP = sim_pin motion.probe-input halui.abort motion.analog-in-00
    ```

  - Invocar halshow con una lista de observación previamente guardada. Como LinuxCNC establece el directorio de trabajo al directorio del archivo INI, puedes hacer referencia a los archivos en ese directorio (ejemplo: my.halshow):

    ```
    APP = halshow my.halshow
    ```

  - Alternativamente, se podría especificar un archivo de lista de observación identificado con un nombre de ruta completo:

    ```
    APP = halshow ~/saved_shows/spindle.halshow
    ```

  - Abrir halscope usando una configuración previamente guardada:

    ```
    APP = halscope -i my.halscope
    ```

4.4.2.10. Sección [TRAJ]



aviso
El nuevo Planificador de trayectoria (TP) está activado de forma predeterminada. Si no tiene configuraciones TP en su sección [TRAJ] - LinuxCNC por defecto hace:
ARC_BLEND_ENABLE = 1
ARC_BLEND_FALLBACK_ENABLE = 0
ARC_BLEND_OPTIMIZATION_DEPTH = 50
ARC_BLEND_GAP_CYCLES = 4
ARC_BLEND_RAMP_FREQ = 100

La sección [TRAJ] contiene parámetros generales para el módulo de planificación de trayectoria en *motion*.

- `ARC_BLEND_ENABLE = 1` - Activa el nuevo TP. Si se establece en 0, TP utiliza mezcla parabólica (1 segmento adelantado). (Predeterminado: 1).
- `ARC_BLEND_FALLBACK_ENABLE = 0` - Recurrir opcionalmente a mezclas parabólicas si la velocidad estimada es más rápida. Sin embargo, esta estimación es aproximada y parece que deshabilitarlo proporciona un mejor rendimiento (Predeterminado: 0).
- `ARC_BLEND_OPTIMIZATION_DEPTH = 50` - Profundidad de anticipación en cantidad de segmentos.
  Para ampliar esto un poco, se puede elegir este valor de forma algo arbitraria. Aquí hay una fórmula para estimar cuánta *profundidad* se necesita para una configuración determinada:

```
# n = v_max / (2.0 * a_max * t_c)
# dónde:
# n = profundidad de optimización
# v_max = velocidad máxima del eje (UU/seg)
# a_max = aceleración máxima del eje (UU/seg)
# t_c = período servo (segundos)
```

Por lo tanto, una máquina con una velocidad máxima del eje de 10 IPS, una aceleración máxima de 100 IPS^2, y un período servo de 0.001 s necesitaría:

10 / (2.0 * 100 * 0.001) = 50 segmentos para alcanzar siempre la velocidad máxima a lo largo del eje más rápido.

En la práctica, afinar este número no es tan importante, ya que la anticipación rara vez necesita toda la profundidad a menos que se tengan muchos segmentos muy cortos. Si durante la prueba se notan ralentizaciones extrañas y no se puede averiguar de dónde vienen, primero intentar aumentar esta profundidad usando la fórmula anterior.

Si aún se ven ralentizaciones extrañas, pueden deberse a que se tienen segmentos cortos en el programa. Si este es el caso, intentar agregar una pequeña tolerancia para la detección Naive CAM. Una buena regla general es esta:

```
# min_length ~= v_req * t_c
# dónde:
# v_req = velocidad deseada en UU/seg
# t_c = servo período (segundos)
```

Si deseas recorrer una ruta a 1 IPS = 60 IPM, y tu período servo es de 0.001 segundos, entonces cualquier segmento más corto que min_length ralentizará la trayectoria. Si configuras la tolerancia Naive CAM a aproximadamente esta longitud mínima, los segmentos demasiado cortos se combinarán para eliminar este cuello de botella. Por supuesto, establecer la tolerancia demasiado alta significa grandes desviaciones de ruta, por lo que debes jugar un poco para encontrar un buen valor. Como consejo, comienza con 1/2 de min_length, luego continúa ajustando según sea necesario. * `ARC_BLEND_GAP_CYCLES = 4`- Qué tan corto debe ser el segmento anterior antes de que el planificador de trayectorias lo *consuma*.

A menudo, una combinación de arco circular dejará segmentos de línea cortos entre mezclas. Como la geometría tiene que ser circular, no podemos mezclar toda una línea si la siguiente es un poco más corta. Puesto que el planificador de trayectoria tiene que tocar cada segmento al menos una vez, significa que segmentos muy pequeños ralentizarán las cosas significativamente. Mi solución a esto es ”consumir” el segmento corto haciéndolo parte del arco de mezcla. Ya que la línea + mezcla es un segmento, no tenemos que reducir la velocidad para alcanzar el segmento muy corto. Probablemente, no necesitarás tocar esta configuración. * `ARC_BLEND_RAMP_FREQ = 20` - Esta es una frecuencia de *corte* para usar velocidades en rampa.

Velocidad en rampa en este caso solo significa aceleración constante sobre el segmento entero. Esto es menos óptimo que un perfil de velocidad trapezoidal, ya que la aceleración no está maximizada. Sin embargo, si el segmento es lo suficientemente corto, no hay tiempo suficiente para acelerar mucho antes de alcanzar el segmento siguiente. Recordemos los segmentos de línea corta de los ejemplos anteriores. Como son líneas, no hay aceleración en esquinas, así que somos libres de acelerar hasta la velocidad solicitada. Sin embargo, si esta línea está entre dos arcos, entonces tendrá que desacelerar rápidamente de nuevo para estar dentro de la velocidad máxima del siguiente segmento. Esto significa tener un pico de aceleración y luego un pico de desaceleración, causando un gran tirón, para muy poco aumento de rendimiento. Esta configuración es una forma de eliminar este tirón para segmentos cortos.

Básicamente, si un segmento se completa en menos tiempo que 1 / ARC_BLEND_RAMP_FREQ, no nos molestamos con un perfil de velocidad trapezoidal en ese segmento, y usaríamos aceleración constante. (Ajustar `ARC_BLEND_RAMP_FREQ = 1000` es equivalente a usar siempre aceleración trapezoidal, si el bucle servo es de 1 kHz).

Se puede caracterizar el peor caso de pérdida de rendimiento comparando la velocidad que alcanza un perfil trapezoidal contra la rampa:

```
# v_ripple = a_max / (4.0 * f)
# donde:
# v_ripple = velocidad promedio ”perdida” debido a la rampa
# a_max = aceleración máxima del eje
# f = frecuencia de corte del INI
```

Para la máquina mencionada, la ondulación para una frecuencia de corte de 20 Hz es 100 / (4 * 20) = 1.25 IPS. Esto parece alto, pero ten en cuenta que es solo una estimación del peor de los casos. En realidad, el perfil de movimiento trapezoidal está limitado por otros factores, como la aceleración normal o velocidad solicitada, por lo que la pérdida de rendimiento real debería ser mucho menor. Aumentar la frecuencia de corte puede sacar más rendimiento, pero hace que el movimiento sea más duro debido a discontinuidades de aceleración. Un valor en el rango de 20 Hz a 200 Hz debería ser razonable para comenzar.

Finalmente, ninguna cantidad de ajustes acelerará una trayectoria de herramienta con muchas esquinas pequeñas y estrechas, ya que estás limitado por la aceleración en las esquinas.

- `SPINDLES = 3` - El número de husillos a soportar. Es imperativo que este número coincida con el parámetro ”num_spindles” pasado al módulo motion.
- `COORDINATES = X Y Z` - Los nombres de los ejes que se controlan. Solo son válidos X, Y, Z, A, B, C, U, V, W. En código G solo se aceptan ejes nombrados en `COORDINATES`. Está permitido escribir un nombre de eje más de una vez (p. ej. X Y Y Z para una máquina de pórtico). Para las *cinemáticas trivkins* comunes, los números de articulación se asignan en secuencia de acuerdo con el parámetro trivkins *coordinates=*. Por tanto, para trivkins *coordinates = xz*, la articulación 0 corresponde a X y la articulación 1 corresponde a Z. Consulte la página de manual de cinemáticas (*$ man kins*) para obtener información sobre trivkins y otros módulos de cinemáticas.
- `LINEAR_UNITS = <unidades>` - Especifica las *unidades máquina* para ejes lineales. Las opciones posibles son milímetros o pulgadas. Esto no afecta las unidades lineales en código NC (las palabras G20 y G21 hacen esto).
- `ANGULAR_UNITS = <unidades>` - Especifica las *unidades máquina* para ejes de rotación. Las opciones posibles son *deg*, *degree* (360 por círculo), *rad*, *radian* (2*π por círculo), *grad* o *gon* (400 por círculo). Esto no afecta las unidades angulares de código NC. En RS274NGC, las palabras A, B y C siempre se expresan en grados.
- `DEFAULT_LINEAR_VELOCITY = 0.0167` - La velocidad inicial para trotes de ejes lineales, en unidades máquina por segundo. El valor que se muestra en *AXIS* equivale a unidades máquina por minuto.
- `DEFAULT_LINEAR_ACCELERATION = 2.0` - En máquinas con cinemáticas no triviales, la aceleración utilizada para trotes ”teleop” (espacio cartesiano), en *unidades máquina* por segundo al cuadrado.
- `MAX_LINEAR_VELOCITY = 5.0` - La velocidad máxima para cualquier eje o movimiento coordinado, en *unidades máquina* por segundo. El valor mostrado es igual a 300 unidades por minuto.
- `MAX_LINEAR_ACCELERATION = 20.0` - La aceleración máxima para cualquier eje o movimiento coordinado, en *unidades máquina* por segundo cuadrado.
- `PLANNER_TYPE = 0` - Selects the trajectory planner type: 0 = trapezoidal (default), 1 = S-curve with jerk limiting. S-curve planning is only active when `PLANNER_TYPE = 1` AND `MAX_LINEAR_JERK > 0`.
- `MAX_LINEAR_JERK = 10000.0` - The maximum jerk (rate of change of acceleration) for coordinated moves, in *machine units* per second cubed. Default is 1e9 (1 billion) if not specified, which effectively disables jerk limiting while avoiding numerical instability. Values are clamped to a maximum of 1e9 to prevent numerical issues in S-curve calculations. When `PLANNER_TYPE = 1`, this enables S-curve trajectory planning. Note: Not specifying MAX_LINEAR_JERK (defaulting to 1e9) produces motion similar to trapezoidal planning (PLANNER_TYPE = 0) but not identical, as extremely high jerk still uses S-curve calculations.

- POSITION_FILE = *position.txt* - Si se establece en un valor no vacío, las posiciones articulares se almacenan entre ejecuciones en este archivo. Esto permite que la máquina comience con las mismas coordenadas que tenía en el apagado. Esto supone que no hubo movimiento de la máquina mientras estuvo apagada. Si no se establece, las posiciones no se almacenan y comenzará en 0 cada vez que se inicie LinuxCNC. Esto puede ayudar en máquinas pequeñas sin interruptores casa. Si usa la interfaz de resolver de Mesa, este archivo se puede usar para emular codificadores absolutos y eliminar la necesidad de homing (sin pérdida de precisión). Ver la página de manual de hostmot2 para más detalles.
- NO_FORCE_HOMING = 1 - (bool) The default behavior is for LinuxCNC to force the user to home the machine before any MDI command or a program is run. Normally, only jogging is allowed before homing. For configurations using identity kinematics, setting NO_FORCE_HOMING = 1 allows the user to make MDI moves and run programs without homing the machine first. Interfaces using identity kinematics without homing ability will need to have this option set to 1.



aviso
LinuxCNC no sabrá tus límites de carrera de articulación cuando se usa NO_FORCE_HOMING = 1.

- HOME = 0 0 0 0 0 0 0 0 0 - Se necesita una posición de inicio mundial para los módulos de cinemáticas que calculan las coordenadas mundiales usando kinematicsForward() al cambiar de modo articular a teleop. Pueden especificarse hasta nueve valores de coordenadas (X Y Z A B C U V W), los elementos finales no utilizados pueden omitirse. Este valor solo se usa para máquinas con cinemáticas no triviales. En máquinas con cinemáticas triviales (fresadoras, tornos, varios tipos de pórtico) este valor se ignora. Nota: la configuración sim de hexapod requiere un valor distinto de cero para la coordenada Z.
- TPMOD = *alternate_trajectory_planning module* [tp_parms=value]
  La variable TPMOD es opcional. Si se especifica, usa un módulo especificado (construido por el usuario) en lugar del predeterminado (tpmod). Los parámetros del módulo (tp_parms) pueden incluirse si los soporta el módulo especificado. La configuración puede anularse desde la línea de comandos usando la opción -t ($ linuxcnc -h).
- NO_PROBE_JOG_ERROR = 0 - Permite omitir la verificación de recorrido de sonda al hacer trote manual.
- NO_PROBE_HOME_ERROR = 0 - Permite omitir la verificación de recorrido de sonda cuando homing esta en progreso.

4.4.2.11. Sección [KINS]

- JOINTS = 3 - Especifica el número de articulaciones (motores) en el sistema. Por ejemplo, una máquina trivkins XYZ con un solo motor para cada eje tiene 3 articulaciones. Una máquina de pórtico con un motor en cada uno de los dos ejes, y dos motores en el tercer eje, tiene 4 articulaciones. (Esta variable de configuración puede ser utilizada por una interfaz gráfica de usuario para establecer el número de articulaciones (num_joints) especificado en el módulo de movimiento (motmod).)
- KINEMATICS = trivkins - Especifica un módulo de cinemáticas para el módulo motion. Las GUIs pueden usar esta variable para especificar la línea loadrt en archivos HAL para el módulo motmod. Para obtener más información sobre los módulos de cinemáticas, consulte la página de manual: $ man kins.

4.4.2.12. Sección [AXIS_<letra>_]

La <*letra*> especifica una de: X Y Z A B C U V W

- `TYPE = LINEAR` - (enum) The type of this axis, either `LINEAR` or `ANGULAR`. Required if this axis is not a default axis type. The default axis types are X,Y,Z,U,V,W = LINEAR and A,B,C = ANGULAR. This setting is effective with the AXIS GUI but note that other GUI's may handle things differently.
- `MAX_VELOCITY = 1.2` - (real) Maximum velocity for this axis in machine units per second.
- `MAX_ACCELERATION = 20.0` - (real) Maximum acceleration for this axis in machine units per second squared.
- `MAX_JERK = 0.0` - (real) Maximum jerk for this axis in machine units per second cubed. Used when S-curve trajectory planning is enabled. When set to 0 (default), no per-axis jerk limiting is applied.
- `MIN_LIMIT = -1000` - (real) The minimum limit (soft limit) for axis motion, in machine units. When this limit is exceeded, the controller aborts axis motion. The axis must be homed before `MIN_LIMIT` is in force. For a rotary axis (A,B,C typ) with unlimited rotation having no `MIN_LIMIT` for that axis in the `[AXIS_<letter>]` section a value of -1e99 is used.
- `MAX_LIMIT = 1000` - (real) The maximum limit (soft limit) for axis motion, in machine units. When this limit is exceeded, the controller aborts axis motion. The axis must be homed before MAX_LIMIT is in force. For a rotary axis (A,B,C typ) with unlimited rotation having no `MAX_LIMIT` for that axis in the `[AXIS_<letter>]` section a value of 1e99 is used.
- `WRAPPED_ROTARY = 1` - (bool) When this is set to 1 for an ANGULAR axis the axis will move 0-359.999 degrees. Positive Numbers will move the axis in a positive direction and negative numbers will move the axis in the negative direction.
- `LOCKING_INDEXER_JOINT = 4` - (int) This value selects a joint to use for a locking indexer for the specified axis <*letter*>. In this example, the joint is 4 which would correspond to the B axis for a XYZAB system with trivkins (identity) kinematics. When set, a G0 move for this axis will initiate an unlock with the `joint.4.unlock pin` then wait for the `joint.4.is-unlocked` pin then move the joint at the rapid rate for that joint. After the move the `joint.4.unlock` will be false and motion will wait for `joint.4.is-unlocked` to go false. Moving with other joints is not allowed when moving a locked rotary joint. To create the unlock pins, use the motmod parameter:

  ```
  unlock_joints_mask=jointmask
  ```

  Los bits jointmask son: (LSB) 0: articulación0, 1: articulación1, 2: articulación2, ...

  Ejemplo: `loadrt motmod ... unlock_joints_mask = 0x38` crea pines de desbloqueo para articulaciones 3,4,5.
- `OFFSET_AV_RATIO = 0.1` - (real) If nonzero, this item enables the use of HAL input pins for external axis offsets:

  ```
  axis.<letra>.eoffset-enable
  axis.<letra>.eoffset-count
  axis.<letra>.eoffset-scale
  ```

Consultar el capítulo: Offsets externos de ejes para información de su uso.

4.4.2.13. Secciones [JOINT_<número>]

<*número*> especifica el número de articulación 0 ... (no_articulaciones-1) El valor de *no_articulaciones* lo establece `[KINS]JOINTS=`.

Las secciones [JOINT_0], [JOINT_1], etc. contienen parámetros generales para los componentes individuales en el módulo de control de articulaciones. Los nombres en la sección comienzan a numerarse en 0 y llegan hasta el número de articulaciones especificado en la entrada [KINS]JOINTS menos 1.

Típicamente (para sistemas que usan *cinemáticas trivkins*, hay correspondencia 1:1 entre una articulación y una letra de eje de coordenadas):

- JOINT_0 = X
- JOINT_1 = Y
- JOINT_2 = Z
- JOINT_3 = A
- JOINT_4 = B
- JOINT_5 = C
- JOINT_6 = U
- JOINT_7 = V
- JOINT_8 = W

Otros módulos de cinemáticas con cinemáticas de identidad están disponibles para admitir configuraciones con conjuntos parciales de ejes. Por ejemplo, usando trivkins con coordenadas=XZ, las relaciones articulación-eje comunes son:

- JOINT_0 = X
- JOINT_1 = Z

Para obtener más información sobre los módulos cinemáticos, consultar la página de manual de *kins* (en una terminal de UNIX teclee man kins).

- TYPE = LINEAR - (enum) The type of joint, either LINEAR or ANGULAR.
- UNITS = INCH - (enum) If specified, this setting overrides the related [TRAJ] UNITS setting, e.g., [TRAJ]LINEAR_UNITS if the TYPE of this joint is LINEAR, [TRAJ]ANGULAR_UNITS if the TYPE of this joint is ANGULAR.
- MAX_VELOCITY = 1.2 - (real) Maximum velocity for this joint in machine units per second.
- MAX_ACCELERATION = 20.0 - (real) Maximum acceleration for this joint in machine units per second squared.
- MAX_JERK = 0.0 - (real) Maximum jerk for this joint in machine units per second cubed. Used when S-curve trajectory planning is enabled. When set to 0 (default), no per-joint jerk limiting is applied.
- BACKLASH = 0.0000 - (real) Backlash in machine units. Backlash compensation value can be used to make up for small deficiencies in the hardware used to drive an joint. If backlash is added to an joint and you are using steppers the STEPGEN_MAXACCEL must be increased to 1.5 to 2 times the MAX_ACCELERATION for the joint. Excessive backlash compensation can cause an joint to jerk as it changes direction. If a COMP_FILE is specified for a joint BACKLASH is not used.
- COMP_FILE = *file.extension* - (string) The compensation file consists of map of position information for the joint. Compensation file values are in machine units. Each set of values are are on one line separated by a space. The first value is the nominal value (the commanded position). The second and third values depend on the setting of COMP_FILE_TYPE. Points in between nominal values are interpolated between the two nominals. Compensation files must start with the smallest nominal and be in ascending order to the largest value of nominals. File names are case sensitive and can contain letters and/or numbers. Currently the limit inside LinuxCNC is for 256 triplets per joint.

  Si se especifica COMP_FILE para un eje, BACKLASH no se utiliza.

- `COMP_FILE_TYPE = 0` or `1` - (int) Specifies the type of compensation file. The first value is the nominal (commanded) position for both types.
  A `COMP_FILE_TYPE` must be specified for each `COMP_FILE`.

  - *Tipo 0:* El segundo valor especifica la posición real a medida que se mueve el eje en la dirección positiva (valor creciente). El tercer valor especifica la posición real a medida que el eje se mueve en la dirección negativa (valor decreciente).

    Ejemplo de tipo 0

    ```
    -1.000 -1.005 -0.995
    0.000 0.002 -0.003
    1.000 1.003 0.998
    ```

  - *Tipo 1:* El segundo valor especifica el offset positivo del nominal mientras se va en la dirección positiva. El tercer valor especifica el offset negativo del nominal mientras se va en una dirección negativa.

    Ejemplo de tipo 1

    ```
    -1.000 0.005 -0.005
    0.000 0.002 -0.003
    1.000 0.003 -0.004
    ```

- `MIN_LIMIT = -1000` - (real) The minimum limit for joint motion, in machine units. When this limit is reached, the controller aborts joint motion. For a rotary joint with unlimited rotation having no `MIN_LIMIT` for that joint in the `[JOINT_N]` section a the value -1e99 is used.
- `MAX_LIMIT = 1000` - (real) The maximum limit for joint motion, in machine units. When this limit is reached, the controller aborts joint motion. For a rotary joint with unlimited rotation having no `MAX_LIMIT` for that joint in the `[JOINT_N]` section a the value 1e99 is used.

---

nota
Para cinemáticas **identidad**, la configuraciones `[JOINT_N]MIN_LIMIT/MAX_LIMIT` deben ser iguales o exceder los límites `[AXIS_L]` correspondientes (uno a una identidad). Estas configuraciones se verifican al inicio cuando se especifican los módulos de cinemáticas trivkins.

---

nota
Las configuraciones `[JOINT_N]MIN_LIMIT/MAX_LIMIT` se hacen valer al trotar en modo articulación antes de homing. Después de homing, los límites de coordenadas `[AXIS_L]MIN_LIMIT/MAX_LIMIT` se usan como restricciones para trote de eje (letra de coordenada) y para movimientos en código G (programas y comandos MDI) por el planificador de trayectoria. El planificador de trayectoria funciona en el espacio cartesiano (XYZABCUVW) y no tiene información sobre el movimiento de articulaciones implementado por **cualquier** módulo de cinemáticas. Es posible que ocurran violaciones a los límites de articulación en código G que obedezca los límites de posición del planificador de trayectoria cuando se usan cinemáticas no identidad. El módulo de movimiento detecta siempre violaciones a los límites de posición de articulación y falla si ocurren durante la ejecución de comandos en código G. Ver el asunto #97 de GitHub relacionado.

- `MIN_FERROR = 0.010` - (real) This is the value in machine units by which the joint is permitted to deviate from commanded position at very low speeds. If MIN_FERROR is smaller than FERROR, the two produce a ramp of error trip points. You could think of this as a graph where one dimension is speed and the other is permitted following error. As speed increases the amount of following error also increases toward the `FERROR` value.

- FERROR = 1.0 - (real) FERROR is the maximum allowable following error, in machine units. If the difference between commanded and sensed position exceeds this amount, the controller disables servo calculations, sets all the outputs to 0.0, and disables the amplifiers. If MIN_FERROR is present in the INI file, velocity-proportional following errors are used. Here, the maximum allowable following error is proportional to the speed, with FERROR applying to the rapid rate set by [TRAJ]MAX_VELOCITY, and proportionally smaller following errors for slower speeds. The maximum allowable following error will always be greater than MIN_FERROR. This prevents small following errors for stationary axes from inadvertently aborting motion. Small following errors will always be present due to vibration, etc.
- LOCKING_INDEXER = 1 - (bool) Indicates the joint is used as a locking indexer.

Estos parámetros están relacionados con Homing; para una mejor explicación leer el capítulo Configuración de homing.

- HOME = 0.0 - (real) The position that the joint will go to upon completion of the homing sequence.
- HOME_OFFSET = 0.0 - (real) The joint position of the home switch or index pulse, in machine units. When the home point is found during the homing process, this is the position that is assigned to that point. When sharing home and limit switches and using a home sequence that will leave the home/limit switch in the toggled state, the home offset can be used define the home switch position to be other than 0 if your HOME position is desired to be 0.
- HOME_SEARCH_VEL = 0.0 - (real) Initial homing velocity in machine units per second. Sign denotes direction of travel. A value of zero means assume that the current location is the home position for the machine. If your machine has no home switches you will want to leave this value at zero.
- HOME_LATCH_VEL = 0.0 - (real) Homing velocity in machine units per second to the home switch latch position. Sign denotes direction of travel.
- HOME_FINAL_VEL = 0.0 - (real) Velocity in machine units per second from home latch position to home position. If left at 0 or not included in the joint rapid velocity is used. Must be a positive number.
- HOME_USE_INDEX = NO - (bool) If the encoder used for this joint has an index pulse, and the motion card has provision for this signal you may set it to yes. When it is yes, it will affect the kind of home pattern used. Currently, you can't home to index with steppers unless you're using StepGen in velocity mode and PID.
- HOME_INDEX_NO_ENCODER_RESET = NO - (bool) Use YES if the encoder used for this joint does not reset its counter when an index pulse is detected after assertion of the joint index_enable HAL pin. Applicable only for HOME_USE_INDEX = YES.
- HOME_IGNORE_LIMITS = NO - (bool) When you use the limit switch as a home switch and the limit switch this should be set to YES. When set to YES the limit switch for this joint is ignored when homing. You must configure your homing so that at the end of your home move the home/limit switch is not in the toggled state you will get a limit switch error after the home move.
- HOME_IS_SHARED = NO - (bool) If the home input is shared by more than one joint set to true to prevent homing from starting if the one of the shared switches is already closed. Set to false (the default) to permit homing if a switch is closed.
- HOME_ABSOLUTE_ENCODER = 0 | 1 | 2 - (int) Used to indicate the joint uses an absolute encoder. At a request for homing, the current joint value is set to the HOME_OFFSET value. If the HOME_ABSOLUTE_ENCODER setting is 1, the machine makes the usual final move to the HOME value. If the HOME_ABSOLUTE_ENCODER setting is 2, no final move is made.
- HOME_SEQUENCE = <*n*> - (int) Used to define the "Home All" sequence. <*n*> must start at 0 or 1 or -1. Additional sequences may be specified with numbers increasing by 1 (in absolute value). Skipping of sequence numbers is not allowed. If a HOME_SEQUENCE is omitted, the joint will not be

homed by the ”Home All” function. More than one joint can be homed at the same time by specifying the same sequence number for more than one joint. A negative sequence number is used to defer the final move for all joints having that (negative or positive) sequence number. For additional info, see: HOME SEQUENCE.

- `VOLATILE_HOME = 0` - (bool) When enabled (set to 1) this joint will be unhomed if the Machine Power is off or if E-Stop is on. This is useful if your machine has home switches and does not have position feedback such as a step and direction driven machine.

Estos parámetros son relevantes para las articulaciones controladas por servos.



aviso
Las siguientes son entradas de archivos INI personalizadas que se pueden encontrar en un archivo INI de muestra o en un archivo generado por asistente. Estos no son utilizados por el software LinuxCNC. Solo están ahí para poner todas las configuraciones en un solo lugar. Para más información sobre entradas de archivo INI personalizadas ver la subsección Secciones y variables personalizadas .

Los siguientes elementos pueden ser utilizados por un componente PID y se supone que la salida es en voltios.

- `DEADBAND = 0.000015` - (real) How close is close enough to consider the motor in position, in machine units.

  Esto a menudo se establece en una distancia equivalente a 1, 1.5, 2, o 3 conteos de codificador, pero no hay reglas estrictas. Las configuraciones más holgadas (grandes) permiten menos *hunting (caza)* de servos a expensas de una menor precisión. Las configuraciones más estrictas (pequeñas) intentan una mayor precisión a expensas de más *hunting*. ¿Es realmente más preciso si también es más incierto? Como regla general, es bueno evitar si puede, o al menos limitar, el *hunting* de servos.

  Ten cuidado de ir por debajo de 1 cuenta de codificador, ya que puedes crear una situación en la que tu servo no estará feliz en ningún lugar. Esto puede ir más allá de *hunting* (lento) a *nervous* (rápido), e incluso *squealing* (estrepitoso), que es fácil de confundir con la oscilación causada por una afinación incorrecta. Es mejor perder un conteo o dos al principio, hasta que hayas realizado una *afinación burda* por lo menos.

  Ejemplo de cálculo de unidades máquina por pulso de codificador a usar al decidir el valor de `DEADBAND`:

$$\frac{1\,revolution}{1000\,lines} X \frac{1\,line}{4\,pulse/line} X \frac{0.2\,units}{1\,revolution} = \frac{0.200\,units}{4000\,pulses} = \frac{0.00005\,units}{1\,pulse}$$

- `BIAS = 0.000` - Esto es utilizado por hm2-servo y algunos otros. BIAS es una cantidad constante que se agrega a la salida. En la mayoría de los casos, debe dejarse en cero. Sin embargo, a veces puede ser útil para compensar servoamplificadores, o para equilibrar el peso de un objeto que se mueve verticalmente. BIAS se desactiva cuando el bucle PID está desactivado, al igual que todos los demás componentes de la salida.
- `P = 50` - La ganancia proporcional para el servo. Este valor multiplica el error entre la posición ordenada y la real en unidades máquina, lo que resulta en una contribución a la tensión calculada para el amplificador del motor. Las unidades en la ganancia P son voltios por unidad máquina, p. ej. $\frac{volts}{unit}$

- `I = 0` - La ganancia integral para el servo. El valor multiplica el error acumulativo entre la posición ordenada y la real en unidades máquina, lo que resulta en una contribución a la tensión calculada para el amplificador de motor. Las unidades en la ganancia I son voltios por unidad máquina por segundo, p. ej., $\frac{volts}{unit\ second}$
- `D = 0` - La ganancia derivativa para el servo. El valor multiplica la diferencia entre el error actual y los anteriores, lo que resulta en una contribución a la tensión calculada para el amplificador del motor. Las unidades en la ganancia D son voltios por unidad de máquina por segundo, p. ej., $\frac{volts}{unit\ second}$
- `FF0 = 0` - Ganancia de avance de orden 0. Este número es multiplicado por el posición ordenada, lo que resulta en una contribución a la tensión calculada para el amplificador del motor. Las unidades en la ganancia FF0 son voltios por unidad máquina, p. ej., $\frac{volts}{unit}$
- `FF1 = 0` - La ganancia de avance de 1er orden. Este número es multiplicado por el cambio en la posición ordenada por segundo, lo que resulta en una contribución al voltaje calculado para el amplificador del motor. Las unidades en la ganancia FF1 son voltios por unidad máquina por segundo, p. ej., $\frac{volts}{unit\ second}$
- `FF2 = 0` - La gganancia de avance de 2do orden. Este número es multiplicado por el cambio en la posición ordenada por segundo por segundo, lo que resulta en un contribución a la tensión calculada para el amplificador del motor. Las unidades en la ganancia FF2 son voltios por unidad máquina por segundo por segundo, p. ej., $\frac{volts}{unit\ second^2}$
- `OUTPUT_SCALE = 1.000`
- `OUTPUT_OFFSET = 0.000`

  Estos dos valores son los factores de escala y el offset para la salida de la articulación al amplificador del motor.

  El segundo valor (offset) se resta de la salida calculada (en voltios), y se divide por el primer valor (factor de escala), antes de ser escrito en los convertidores D/A. Las unidades de los valores de escala están en voltios verdaderos por voltios de salida DAC. Las unidades del valor de offset está en voltios. Estos se pueden usar para linealizar un DAC. Específicamente, al escribir salidas, LinuxCNC primero convierte la salida deseada en unidades cuasi-SI a valores brutos de actuador, p. ej., voltios para un amplificador DAC. Esta escala se parece a: $raw = \frac{output - offset}{scale}$

  El valor de la escala se puede obtener analíticamente haciendo un análisis de unidades, es decir, las unidades son [unidades SI de salida]/[unidades de actuador]. Por ejemplo, en una máquina con un amplificador de modo de velocidad tal que 1 voltio da como resultado una velocidad de 250 mm/seg.

$$amplifier[volts] = (output[\frac{mm}{sec}] - offset[\frac{mm}{sec}])/250\ \frac{mm}{secvolt}$$

  Tener en cuenta que las unidades del offset están en unidades máquina, p. ej., mm/s, y se restan previamente de las lecturas del sensor. El valor para este offset se obtiene al encontrar el valor de tu salida que produce 0.0 para la salida del actuador. Si el DAC está linealizado, este offset es normalmente 0.0.

  La escala y el offset también se pueden usar para linealizar el DAC, resultando en valores que reflejan los efectos combinados de la ganancia del amplificador, no linealidad del DAC, unidades DAC, etc.

  Para hacer esto, seguir este procedimiento.

1. Cree una tabla de calibración para la salida, alimentando el DAC con el voltaje deseado y midiendo el resultado.
2. Hacer un ajuste lineal de mínimos cuadrados para obtener los coeficientes a, b tales como $measured = a * raw + b$
3. Tenga en cuenta que queremos una salida en bruto de modo que nuestro resultado medido sea idéntico a la salida ordenada. Esto significa
   a. $command = a * raw + b$
   b. $raw = (command - b) / a$
4. Como resultado, los coeficientes a y b del ajuste lineal pueden ser utilizados como la escala y el offset para el controlador directamente.

La siguiente tabla es un ejemplo de mediciones de voltaje.

Cuadro 4.1: Mediciones del voltaje de salida

Bruto	Medido
-10	-9.93
-9	-8.83
0	-0.03
1	0.96
9	9.87
10	10.87

- `MAX_OUTPUT = 10` - El valor máximo para la salida de la compensación PID que se escribe en el amplificador del motor, en voltios. El valor calculado de salida está sujeto a este límite. El límite se aplica antes de escalar a unidades de salida en bruto. El valor se aplica simétricamente tanto al lado positivo como al negativo.
- `INPUT_SCALE = 20000` - En configuraciones de muestra
- `ENCODER_SCALE = 20000` - En configuraciones construidas con PnCconf

Especifica el número de pulsos que corresponde a un movimiento de una unidad máquina como se establece en la sección `[TRAJ]`. Para una articulación lineal, una unidad máquina será igual a la configuración de `LINEAR_UNITS`. Para una articulación angular, una unidad es igual a la configuración en `ANGULAR_UNITS`. Un segundo número, si se especifica, se ignora. Por ejemplo, en un codificador de 2000 cuentas por revolución, y una transmisión de 10 revoluciones por pulgada y unidades deseadas en pulgadas, tenemos:

$$input\,scale = 2000\,\frac{counts}{rev} * 10\,\frac{rev}{inch} = 20000\,\frac{counts}{inch}$$

Estos parámetros son relevantes para las articulaciones controladas por motores a pasos.



aviso
Las siguientes son entradas de archivos INI personalizadas que se pueden encontrar en un archivo INI de muestra o un archivo generado por asistente. Estos no son utilizados por el software LinuxCNC y solo están ahí para poner todas las configuraciones en un solo lugar. Para más información sobre entradas de archivo INI personalizadas ver la subsección Secciones y variables personalizadas .

Los siguientes elementos pueden ser utilizados por un componente StepGen.

- `SCALE = 4000` - En configuraciones de muestra
- `STEP_SCALE = 4000` - En configuraciones construidas con PnCconf

Especifica el número de pulsos que corresponden a un movimiento de una unidad máquina como se establece en la sección `[TRAJ]`. Para sistemas paso a paso, esto es el número de pulsos de paso emitidos por unidad máquina. Para una articulación lineal una unidad de máquina será igual a la configuración de `LINEAR_UNITS`. Para una articulación angular una unidad es igual a la configuración en `ANGULAR_UNITS`. Para servo sistemas, este es el número de pulsos de retroalimentación por unidad máquina. Un segundo número, si se especifica, se ignora.

Por ejemplo, en un motor paso a paso de 1.8 grados con semipasos, y transmisión de 10 revoluciones por pulgada, y unidades de máquina deseadas en pulgadas, tendríamos:

$$input\,scale = \frac{2\,steps}{1.8\,degrees} * 360\,\frac{degree}{rev} * 10\,\frac{rev}{inch} = 4000\,\frac{steps}{inch}$$

---

nota
Los archivos INI y HAL antiguos usaban `INPUT_SCALE` para este valor.

- `ENCODER_SCALE = 20000` (Opcionalmente utilizado en configuraciones construidas con PnCconf) - Especifica el número de pulsos que corresponden a un movimiento de una unidad máquina como se establece en la sección `[TRAJ]`. Para una articulación lineal, una unidad máquina será igual a la configuración de `LINEAR_UNITS`. Para una articulación angular, una unidad es igual a la configuración en `ANGULAR_UNITS`. Un segundo número, si se especifica, se ignora. Por ejemplo, en un conteo de 2000 por revolucion de codificador, transmisión de 10 revoluciones por pulgada, y unidades deseadas en pulgadas, se tiene:

$$input\,scale = 2000\,\frac{counts}{rev} * 10\,\frac{rev}{inch} = 20000\,\frac{counts}{inch}$$

- `STEPGEN_MAXACCEL = 21.0` - Límite de aceleración para el generador de pasos. Esto debería ser entre 1 % y 10 % más grande que `MAX_ACCELERATION` de la articulación. Este valor mejora la afinación de ”bucle de posición” de StepGen. Si has agregado compensación de holgura a una articulación, entonces esto debería ser 1.5 a 2 veces mayor que `MAX_ACCELERATION`.
- `STEPGEN_MAXVEL = 1.4` - Los archivos de configuración antiguos tienen también un límite de velocidad para el generador de pasos. Si se especifica, también debería ser entre 1 % y 10 % más grande que `MAX_VELOCITY` de la articulación. Pruebas posteriores han demostrado que el uso de `STEPGEN_MAXVEL` no mejora el ajuste del bucle de posición de StepGen.

4.4.2.14. Sección(es) [SPINDLE_<número>]

El <*número*> especifica el número de husillo 0 ... (no_husillos-1)
El valor de *no_husillos* se configura en `[TRAJ]SPINDLES=`.
Predeterminadamente la velocidad máxima del husillo en avance y reversa es aproximadamente 2147483000 RPM.
Predeterminadamente la velocidad mínima del husillo en avance y reversa es 0 RPM.
Predeterminadamente el incremento es 100 RPM.
Se pueden modificar estos valores predeterminados configurando las variables INI siguientes:

nota
Estas configuraciones son para el componente controlador de movimiento. Las pantallas de control pueden limitar estas configuraciones más adelante.

- `MAX_FORWARD_VELOCITY = 20000` - (real) The maximum spindle speed (in rpm) for the specified spindle. Optional. This will also set MAX_REVERSE_VELOCITY to the negative value unless overridden.
- `MIN_FORWARD_VELOCITY = 3000` - (real) The minimum spindle speed (in rpm) for the specified spindle. Optional. Many spindles have a minimum speed below which they should not be run. Any spindle speed command below this limit will be /increased/ to this limit.
- `MAX_REVERSE_VELOCITY = 20000` - (real) This setting will default to `MAX_FORWARD_VELOCITY` if omitted. It can be used in cases where the spindle speed is limited in reverse. Set to zero for spindles which must not be run in reverse. In this context ”max” refers to the absolute magnitude of the spindle speed.
- `MIN_REVERSE_VELOCITY = 3000` - (real) This setting is equivalent to `MIN_FORWARD_VELOCITY` but for reverse spindle rotation. It will default to the MIN_FORWARD_VELOCITY if omitted.
- `INCREMENT = 200` - (real) Sets the step size for spindle speed increment / decrement commands. This can have a different value for each spindle. This setting is effective with AXIS and Touchy but note that some control screens may handle things differently.
- `HOME_SEARCH_VELOCITY = 100` - (real) FIXME: Spindle homing not yet working. Sets the homing speed (rpm) for the spindle. The spindle will rotate at this velocity during the homing sequence until the spindle index is located, at which point the spindle position will be set to zero. Note that it makes no sense for the spindle home position to be any value other than zero, and so there is no provision to do so.
- `HOME_SEQUENCE = 0` - (int) FIXME: Spindle homing not yet working Controls where in the general homing sequence the spindle homing rotations occur. Set the `HOME_SEARCH_VELOCITY` to zero to avoid spindle rotation during the homing sequence.

4.4.2.15. Sección [EMCIO]

- `TOOL_TABLE = tool.tbl` - (string) The file which contains tool information, described in the User Manual.
- `DB_PROGRAM = db_program` - (string) Path to an executable program that manages tool data. When a DB_PROGRAM is specified, a TOOL_TABLE entry is ignored.
- `TOOL_CHANGE_POSITION = 0 0 2` - Especifica la ubicación XYZ a moverse al realizar un cambio de herramienta si se utilizan tres dígitos. Especifica la ubicación XYZABC cuando se usan 6 dígitos. Especifica la ubicación XYZABCUVW cuando se utilizan 9 dígitos. Los cambios de herramienta se pueden combinar. Por ejemplo, si combina la pinola con la posición de cambio, puede mover primero la Z y luego la X e Y.
- `TOOL_CHANGE_WITH_SPINDLE_ON = 1` - (bool) The spindle will be left on during the tool change when the value is 1. Useful for lathes or machines where the material is in the spindle, not the tool.
- `TOOL_CHANGE_QUILL_UP = 1` - (bool) The Z axis will be moved to machine zero prior to the tool change when the value is 1. This is the same as issuing a `G0 G53 Z0`.
- `TOOL_CHANGE_AT_G30 = 1` - (bool) The machine is moved to reference point defined by parameters 5181-5186 for G30 if the value is 1. For more information see G-code Parameters and G-code G30-G30.1.
- `RANDOM_TOOLCHANGER = 1` - (bool) This is for machines that cannot place the tool back into the pocket it came from. For example, machines that exchange the tool in the active pocket with the tool in the spindle.

4.5. Configuración de homing

4.5.1. Descripción general

Homing establece el origen cero de las coordenadas máquina G53. Los límites suaves se definen en relación con el origen de la máquina. Los límites suaves automáticamente desaceleran y detienen los ejes antes de que toquen los interruptores de límites. Una máquina configurada y funcionando correctamente no se moverá más allá de los límites suaves (software) y tendrá el origen de la máquina configurado tan repetible como lo permita el mecanismo de interruptor/índice de casa. Linuxcnc puede ser llevado a casa a ojo (marcas de alineación), con interruptores, con interruptores e índice de codificador, o utilizando codificadores absolutos. El recorrido de homing parece bastante simple: simplemente mueve cada articulación a una ubicación conocida, y establece las variables internas de LinuxCNC en consecuencia. Sin embargo, diferentes máquinas tienen diferentes requisitos, y el recorrido de homing es realmente bastante complicado.

---

nota
Si bien es posible usar LinuxCNC sin interruptores, procedimientos de homing o interruptores de límite, la seguridad adicional de los límites suaves no es suficiente.

4.5.2. Prerrequisitos

Homing se basa en algunos supuestos fundamentales de la máquina.

- Las direcciones negativa y positiva se basan en Movimientos de la herramienta que puede ser diferentes del movimiento real de la máquina. Es decir, en una fresadora, la mesa se mueve en lugar de la herramienta.
- Todo está referenciado desde el origen G53, cero de la máquina. El origen puede estar en cualquier lugar (incluso fuera de donde puede moverse)
- El origen cero de la máquina G53 está típicamente dentro del área de límites suaves, pero no necesariamente.
- El offset del interruptor de casa establece dónde está el origen, pero incluso esto se referencia desde el origen.
- Cuando se usa homing de índice de codificador, el offset del interruptor de casa se calcula a partir de la posición de referencia del codificador, después de que el interruptor de casa ha sido activado.
- Los límites suaves (software) negativos son lo máximo que puede mover en la dirección negativa después de homing. (pero pueden no ser negativos en sentido absoluto)
- Los límites suaves (software) positivos son lo máximo que se puede mover en la dirección positiva después de homing. (pero podrían no ser positivos en sentido absoluto, aunque es habitual establecerlo como un número positivo)
- Los límites suaves (software) están dentro del área de interruptores de límite.
- La posición de casa (final) esta dentro del área de límite suave
- Si se utiliza homing basado en interruptores, los interruptores de homing utilizan los interruptores de límite (interruptor de casa / límite compartido), o cuando se usa un interruptor de casa separado, están dentro del área de interruptores de límite.

- Si usa un interruptor de casa independiente, es posible que se comience homing en el lado equivocado del interruptor, lo que combinado con la opción HOME_IGNORE_LIMITS, provocará un choque fuerte. Se puede evitar esto haciendo que el interruptor de casa conmute su estado cuando el disparador está en un lado en particular hasta que vuelva a pasar el punto nuevamente. Dicho de otra manera, el estado del interruptor de casa debe representar la posición del disparador con respecto al interruptor (es decir, *antes* o *después* del interruptor), y debe permanecer así incluso si el disparador pasa por el interruptor en la misma dirección.

---

nota
Aunque es posible usar LinuxCNC con el origen de máquina G53 fuera de los límites suaves de máquina, si se usa G28 o G30 sin configurar los parámetros, se va al origen predeterminadamente. Esto activará los interruptores de límite antes de alcanzar la posición.

4.5.3. Ejemplo de disposición de interruptor de casa separado

Este ejemplo muestra los interruptores de límite mínimo y máximo con un interruptor de casa separado.

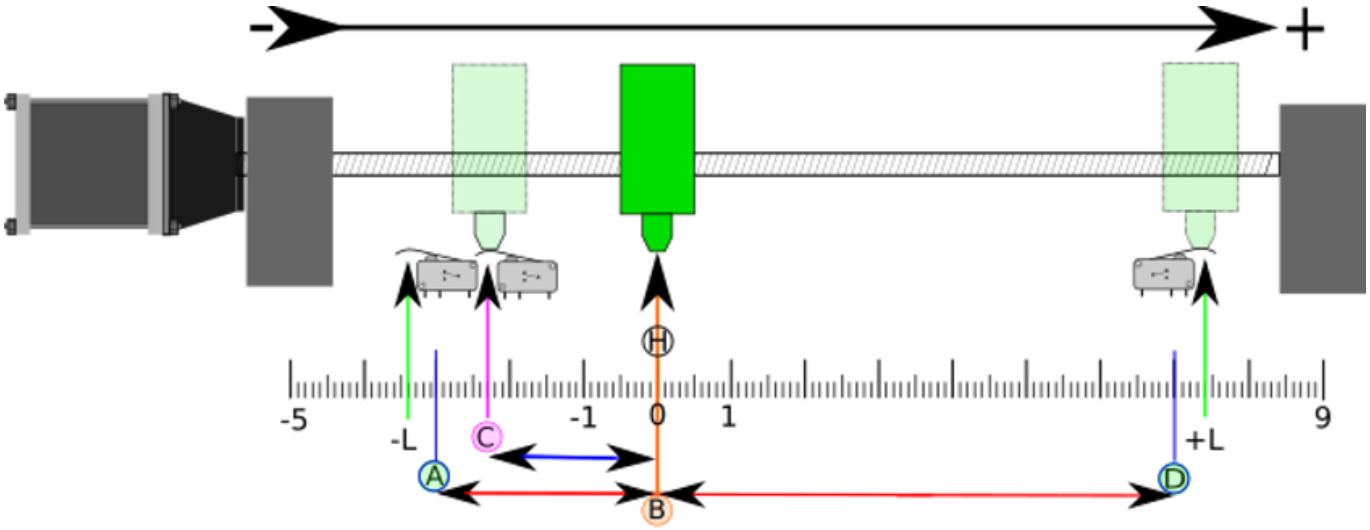


Figura 4.6: Disposición demostrativa de interruptor separado

- A es el límite suave negativo
- B es el origen de las coordenadas de máquina G53
- C es el punto de disparo del interruptor de casa
- D es el límite suave positivo
- H es la posición de casa final (HOME) = 0 unidades
- -L y +L son los puntos de disparo de los interruptores de límite
- A<->B son los límites suaves negativos (MIN_LIMITS) = -3 unidades
- B<->C es el home_offset (HOME_OFFSET) = -2.3 unidades
- B<->D son los límites suaves positivos (MAX_LIMITS) = 7 unidades
- A<->D es la carrera total = 10 unidades

- La distancia entre los interruptores de límite y los límites suaves (-L<->A y D<-+L) se amplía en este ejemplo
- Tenga en cuenta que hay una distancia entre los interruptores de límite y el contacto físico duro de arribo después de que el amplificador esté desactivado.

---

nota
Homing establece el sistema de coordenadas G53, mientras que el origen de la máquina (punto cero) puede estar en cualquier lugar, establecer el punto cero en el límite suave negativo hace que todas las coordenadas G53 sean positivas, lo cual es probablemente mas fácil de recordar. Haga esto configurando MIN_LIMIT = 0 y asegúrese de que MAX_LIMIT sea positivo.

4.5.4. Ejemplo de disposición de interruptor de límite/casa compartido

Este ejemplo muestra un interruptor de límite máximo con un interruptor combinado de límite mínimo/casa.

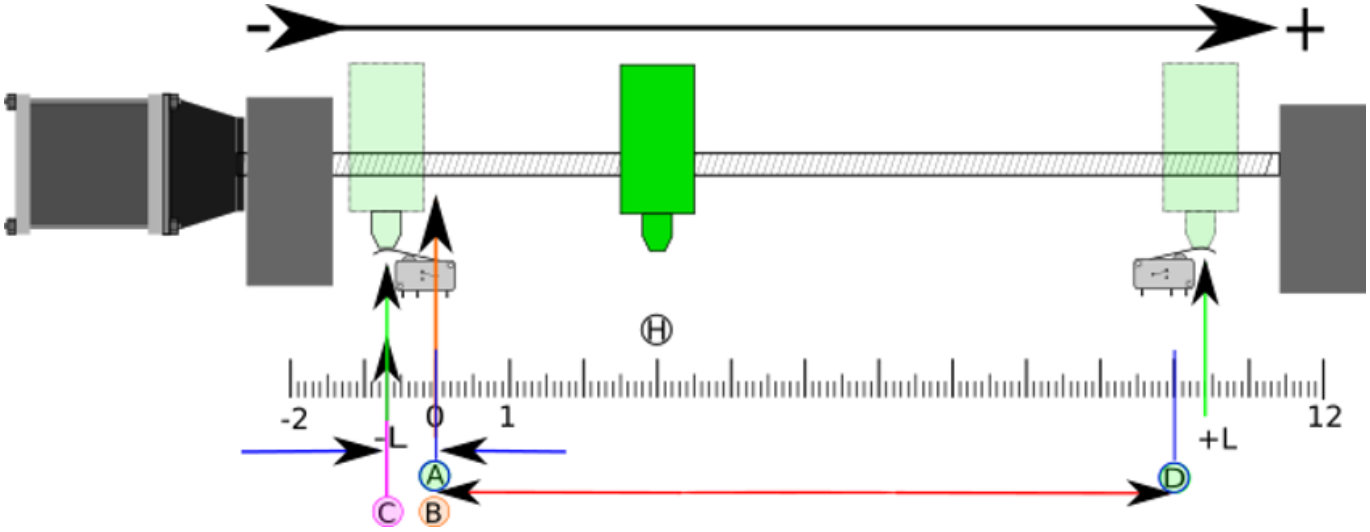


Figura 4.7: Disposición demostrativa de interruptor compartido

- A es el límite suave negativo.
- B es la coordenada Origen de máquina G53.
- C es el punto de disparo del interruptor de casa compartido con (-L) límite mínimo.
- D es el límite suave positivo.
- H es la posición final (HOME) = 3 unidades.
- -L y +L son los puntos de disparo del interruptor de límite.
- A<->B son los límites suaves negativos (MIN_LIMITS) = 0 unidades.
- B<->C es el home_offset (HOME_OFFSET) = -0.7 unidades.
- B<->D son los límites suaves positivos (MAX_LIMITS) 10 unidades.
- A<->D es el recorrido total = 10 unidades.

- La distancia entre los interruptores de límite y los límites suaves (-L<->A y D<->+L) se amplía en este ejemplo.
- Tenga en cuenta que hay una distancia entre los interruptores de límite y el contacto físico duro de arribo después de que el amplificador esté desactivado.

4.5.5. Secuencia de homing

Hay cuatro posibles secuencias homing definidas por el signo de HOME_SEARCH_VEL y HOME_LATCH_VEL junto con los parámetros de configuración asociados como se muestra en la siguiente tabla. Existen dos condiciones básicas, HOME_SEARCH_VEL y HOME_LATCH_VEL son del mismo signo o son signos opuestos. Para una descripción detallada de lo que hace cada parámetro de configuración, ver la sección siguiente.

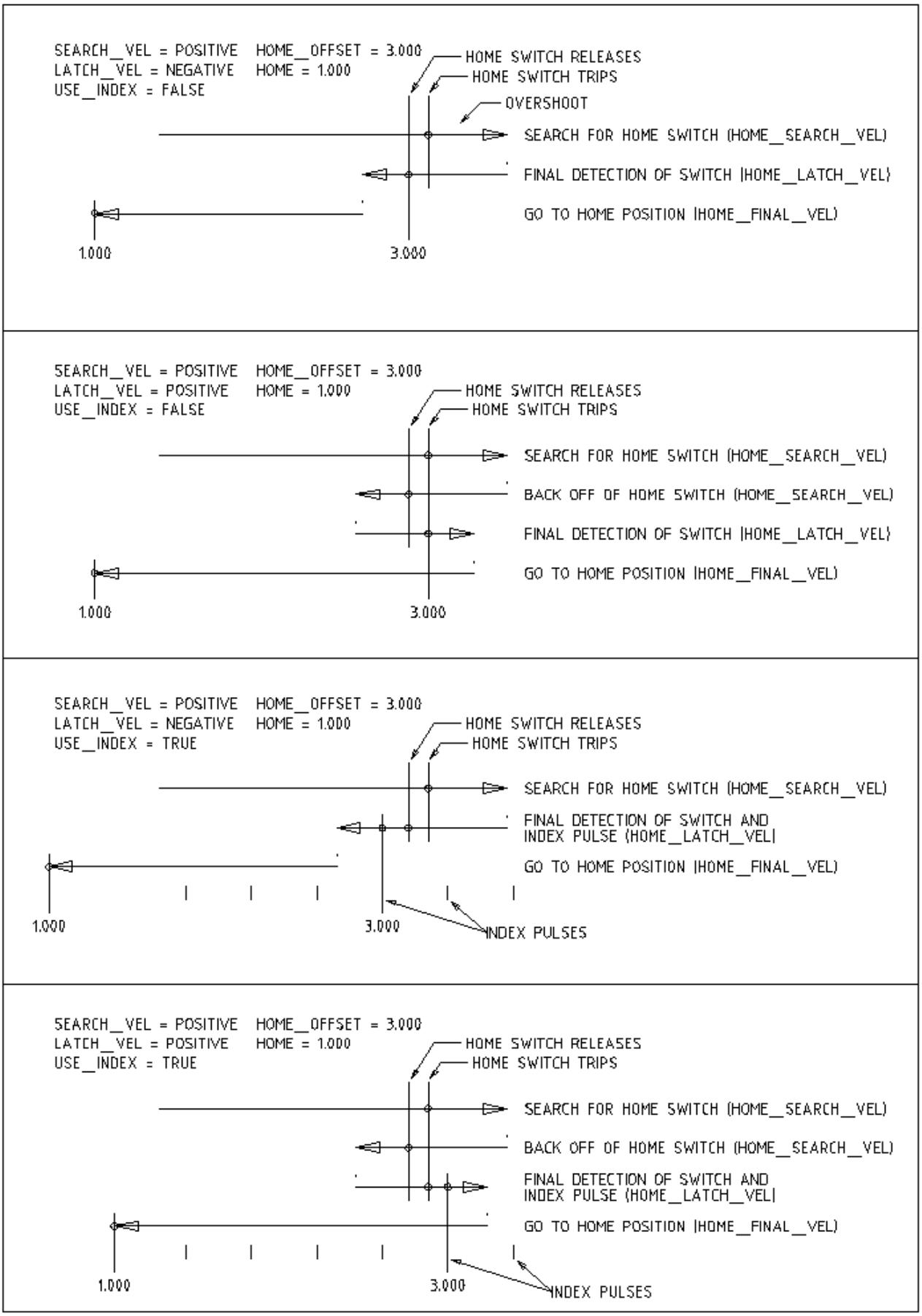


Figura 4.8: Secuencias homing

4.5.6. Configuración

Lo siguiente determina exactamente cómo se comporta la secuencia homing. Se definen en una sección [JOINT_n] del archivo INI.

Tipo de homing	HOME_SEARCH_VEL	HOME_LATCH_VEL	HOME_USE_INDEX
Inmediato	0	0	NO
Solo índice	0	no-cero	SI
Solo interruptor	no-cero	no-cero	NO
Interruptor e índice	no-cero	no-cero	SI

nota
Cualquier otra combinación puede provocar un error.

4.5.6.1. HOME_SEARCH_VEL

Esta variable tiene unidades de unidades máquina por segundo.

El valor predeterminado es cero. Un valor de cero hace que LinuxCNC asuma que no hay interruptor de casa; se omite la etapa de búsqueda de home.

Si HOME_SEARCH_VEL no es cero, entonces LinuxCNC supone que hay un interruptor de casa. Comienza comprobando si el interruptor de casa ya está activado. Si lo está, retrocede desde el interruptor a HOME_SEARCH_VEL. La dirección del retroceso es opuesta al signo de HOME_SEARCH_VEL. Después busca el interruptor de casa moviéndose en la dirección especificada por el signo de HOME_SEARCH_VEL, a una velocidad determinada por su valor absoluto. Cuando se detecta el interruptor de casa, la articulación se detendrá tan rápido posible, pero siempre habrá un exceso. La cantidad de exceso depende de la velocidad. Si es demasiado alto, la articulación podría sobrepasar lo suficiente para alcanzar un interruptor de límite o chocar contra el final de carrera. Por otro lado, si HOME_SEARCH_VEL es demasiado bajo, el homing puede tomar mucho tiempo.

4.5.6.2. HOME_LATCH_VEL

Esta variable tiene unidades de unidades máquina por segundo.

Especifica la velocidad y dirección que utiliza LinuxCNC cuando realiza su determinación final precisa del interruptor de casa (si está presente) y de la ubicación del pulso índice (si está presente). Por lo general, será más lento que la velocidad de búsqueda para maximizar la precisión. Si HOME_SEARCH_VEL y HOME_LATCH_VEL tienen el mismo signo, entonces la fase de enclavamiento se realiza mientras se mueve en la misma dirección que la fase de búsqueda (en ese caso, LinuxCNC primero retrocede desde el interruptor, antes de moverse hacia él nuevamente a la velocidad de enclavamiento). Si HOME_SEARCH_VEL y HOME_LATCH_VEL tienen signos opuestos, la fase de enclavamiento se realiza mientras se mueve en la dirección opuesta a la fase de búsqueda. Eso significa que LinuxCNC enclavará el primer pulso después de que salga del interruptor. Si HOME_SEARCH_VEL es cero (lo que significa que no hay un interruptor de casa), y este parámetro no es cero, LinuxCNC pasa a buscar el pulso de índice. Si HOME_SEARCH_VEL no es cero y este parámetro es cero, es un error y la operación de búsqueda fallará. El valor predeterminado es cero.

4.5.6.3. HOME_FINAL_VEL

Esta variable tiene unidades de unidades máquina por segundo.

Especifica la velocidad que utiliza LinuxCNC cuando realiza su movimiento desde HOME_OFFSET a la posición HOME. Si falta HOME_FINAL_VEL en el archivo INI, se usa la velocidad máxima de articulación para hacer este movimiento. El valor debe ser un número positivo.

4.5.6.4. HOME_IGNORE_LIMITS

Puede contener los valores YES / NO. El valor predeterminado para este parámetro es NO. Este indicador determina si LinuxCNC ignorará la entrada del interruptor de límite para esta articulación mientras hace homing. Esta configuración no ignorará las entradas de límite para otras articulaciones. Si no se tiene un interruptor de casa separado, configurarlo en YES y conectar la señal del interruptor de límite a la entrada del interruptor de casa de la articulación en HAL. LinuxCNC ignorará la entrada del interruptor de límite para esta articulación durante homing. Para usar solo una entrada para todo homing y límites, se tendrán que bloquear en HAL las señales de límite de las articulaciones que no están haciendo homing, y mandar a casa una articulación a la vez.

4.5.6.5. HOME_USE_INDEX

Especifica si hay o no un pulso de índice. Si la bandera es verdadera (HOME_USE_INDEX = YES), LinuxCNC se enclavará en el flanco ascendente del pulso índice. Si es falso, LinuxCNC enclavará en el flanco ascendente o descendente del interruptor de casa (dependiendo de los signos de HOME_SEARCH_VEL y HOME_LATCH_VEL). El valor predeterminado es NO.

---

nota
HOME_USE_INDEX requiere conexiones en su archivo HAL para `joint.n.index-enable` desde `encoder.n.index-enable`.

4.5.6.6. HOME_INDEX_NO_ENCODER_RESET

El valor predeterminado es NO. Utilizar YES si el codificador utilizado para esta articulación no restablece su contador cuando se detecta un pulso de índice después de la aserción del pin HAL de la articulación index_enable. Aplicable solo si HOME_USE_INDEX = YES.

4.5.6.7. HOME_OFFSET

Esto define la ubicación del punto cero de origen del sistema de coordenadas G53 de la máquina. Es la distancia (offset), en unidades articulares, desde el origen de la máquina hasta el punto de disparo del interruptor de casa o pulso índice. Después de detectar el punto de disparo del interruptor/pulso de índice, LinuxCNC establece la posición de la coordenada a HOME_OFFSET, definiendo así el origen, desde el cual hace referencia el limite suave. El valor por predeterminado es cero.

---

nota
La ubicación del interruptor de casa, como lo indica la variable HOME_OFFSET, puede estar dentro o fuera de los límites suaves. Se compartirán con o dentro de los interruptores de límite duro.

4.5.6.8. HOME

La posición a la que irá la articulación al finalizar la secuencia homing. Después de detectar el interruptor de casa o el interruptor de casa y luego el pulso de índice (según la configuración) y establecer la coordenada de ese punto en HOME_OFFSET, LinuxCNC se traslada a HOME como el paso final del proceso de homing. El valor predeterminado es cero. Tener en cuenta que incluso si este parámetro es igual que HOME_OFFSET, la articulación sobrepasará ligeramente la posición enclavada mientras para. Por lo tanto, siempre habrá un pequeño movimiento en este momento (a menos que HOME_SEARCH_VEL sea cero, y se omita toda la etapa de búsqueda/enclavamiento). Este movimiento final se realizará a la velocidad máxima de la articulación a menos que HOME_FINAL_VEL tenga valor.

nota
La distinción entre *HOME_OFFSET* y *HOME* es que *HOME_OFFSET* establece primero la ubicación de origen y la escala en la máquina aplicando el valor *HOME_OFFSET* a la ubicación donde se encontró casa, y luego *HOME* dice dónde debe moverse la articulación a esa escala.

4.5.6.9. HOME_IS_SHARED

Si no hay una entrada de interruptor de casa separada para esta articulación, sino un número de interruptores momentáneos conectados al mismo pin, establecer este valor en 1 para evitar que el homing se inicie si uno de los interruptores compartidos ya está cerrado. Establecer este valor en 0 para permitir homing incluso si el interruptor ya está cerrado.

4.5.6.10. HOME_ABSOLUTE_ENCODER

Usado con codificadores absolutos. Cuando se solicita llevar a a casa la articulación, la posición actual se establece en el valor *[JOINT_n]HOME_OFFSET*.

El movimiento final a la posición *[JOINT_n]HOME* es opcional según a la configuración *HOME_ABSOLUTE_E*

```
HOME_ABSOLUTE_ENCODER = 0 (predeterminado) la articulación no usa un codificador absoluto
HOME_ABSOLUTE_ENCODER = 1 Codificador absoluto, movimiento final a [JOINT_n]HOME
HOME_ABSOLUTE_ENCODER = 2 Codificador absoluto, SIN movimiento final a [JOINT_n]HOME
```

nota
Una configuración HOME_IS_SHARED se ignora silenciosamente.

nota
Una solicitud para volver a llevar a casa la articulación se ignora silenciosamente.

4.5.6.11. HOME_SEQUENCE

Se usa para definir una secuencia homing de múltiples articulaciones **HOME ALL** y aplicar un orden de referencia (p. ej., Z no puede llevarse a casa si X aún no lo está). Una articulación puede ser llevada a casa después de que todas las articulaciones con un valor más bajo (en valor absoluto) de HOME_SEQUENCE ya han sido llevadas a casa y están en HOME_OFFSET. Si dos articulaciones tienen la misma HOME_SEQUENCE, pueden ser llevadas a casa al mismo tiempo.

nota
Si HOME_SEQUENCE no se especifica, la articulación no será llevada a casa por la secuencia **HOME ALL** (pero si individualmente con comandos homing de la articulación).

El número inicial de HOME_SEQUENCE puede ser 0, 1 (o -1). El valor absoluto de los números de secuencia debe incrementarse en uno; saltar números de secuencia no está permitido. Si un número de secuencia se omite, **HOME ALL** se detendrá al finalizar el ultimo número de secuencia válido.

Los valores de HOME_SEQUENCE **negativos** indican que las articulaciones en la secuencia debería **sincronizar el movimiento final** a [JOINT_n]HOME esperando hasta que todas las articulaciones en la secuencia estén listas. Si alguna articulación tiene un HOME_SEQUENCE **negativo**,

todas las articulaciones con el mismo valor absoluto (positivo o negativo) del valor del elemento HOME_SEQUENCE se sincronizarán en el movimiento final.

Un HOME_SEQUENCE **negativo** también se aplica a los comandos para llevar a casa una sola articulación. Si el valor de HOME_SEQUENCE es **negativo**, todas las articulaciones que tengan el mismo valor absoluto de esa HOME_SEQUENCE serán **llevadas a casa juntas con un movimiento final sincronizado**. Si el valor de HOME_SEQUENCE es cero o positivo, un comando para llevar a casa la articulación solo afectará a la articulación especificada.

El trote en modo articulación de articulaciones con HOME_SEQUENCE negativa no está permitido. En aplicaciones de pórtico comunes, tal trote puede conducir a la desalineación. Hay que tener en cuenta que el trote convencional en coordenadas mundiales siempre está disponible una vez que la máquina este en casa.

Ejemplos para un sistema de 3 articulaciones

Dos secuencias (0,1), sin sincronización

```
[JOINT_0]HOME_SEQUENCE = 0
[JOINT_1]HOME_SEQUENCE = 1
[JOINT_2]HOME_SEQUENCE = 1
```

Dos secuencias, articulaciones 1 y 2 sincronizadas

```
[JOINT_0]HOME_SEQUENCE =  0
[JOINT_1]HOME_SEQUENCE = -1
[JOINT_2]HOME_SEQUENCE = -1
```

Con valores mixtos positivos y negativos, las articulaciones 1 y 2 sincronizadas

```
[JOINT_0]HOME_SEQUENCE =  0
[JOINT_1]HOME_SEQUENCE = -1
[JOINT_2]HOME_SEQUENCE =  1
```

Una secuencia, sin sincronización

```
[JOINT_0]HOME_SEQUENCE = 0
[JOINT_1]HOME_SEQUENCE = 0
[JOINT_2]HOME_SEQUENCE = 0
```

Una secuencia, todas las articulaciones sincronizadas

```
[JOINT_0]HOME_SEQUENCE = -1
[JOINT_1]HOME_SEQUENCE = -1
[JOINT_2]HOME_SEQUENCE = -1
```

4.5.6.12. VOLATILE_HOME

Si esta configuración es verdadera, esta articulacion queda fuera casa cada vez que la máquina pasa al estado OFF. Esto es apropiado para cualquier articulación que no mantenga la posición cuando el controlador de la articulación está apagado. Algunos controladores de pasos, especialmente los de micropasos, pueden necesitar esto.

4.5.6.13. LOCKING_INDEXER

Si esta articulación es un indexador rotativo con bloqueo, se desbloqueará antes de ir a casa y se bloqueará después.

4.5.6.14. Homing inmediato

Si una articulación no tiene interruptores de casa o no tiene una posición de casa lógica, como una articulación rotativa, y se desea que la casa de esa articulación sea la posición actual cuando se presione el botón ”Home All” en la GUI AXIS, se necesitan las siguientes entradas INI para esa articulación.

```
HOME_SEARCH_VEL = 0
HOME_LATCH_VEL = 0
HOME_USE_INDEX = NO
HOME_OFFSET = 0 (o el offset de la posición de casa (HOME))
HOME_SEQUENCE = 0 (u otro número de secuencia válido)
```

---

nota
Los valores predeterminados para HOME_SEARCH_VEL, HOME_LATCH_VEL, HOME_USE_INDEX, HOME y HOME_OFFSET son **cero**, por lo que pueden ser omitidos cuando se solicita homing inmediato. Normalmente debería incluirse un número válido de HOME_SEQUENCE ya que omitir una HOME_SEQUENCE elimina la articulación del **HOME ALL** como se indicó anteriormente.

4.5.6.15. Inhibiendo homing

Se proporciona un pin HAL (motion.homing-inhibit) para no permitir iniciación de homing tanto para ”Home todos” como para articulación individual.

Algunos sistemas aprovechan las disposiciones para sincronizar movimientos finales de homing articulares controlados por elementos negativos [JOINT_N]HOME_SEQUENCE = del archivo INI. Predeterminadamente, las disposiciones de sincronización no permiten trote de **articulación** antes de homing con el fin de evitar trotes de **articulación** que podrían desalinear la máquina (perpendicularidad de pórtico, por ejemplo).

El integrador del sistema puede permitir trote **articular** antes de homing con lógica HAL que cambie los elementos [JOINT_N]HOME_SEQUENCE. Esta lógica también debe evaluar el pin **motion.homing-inhibit** para garantizar que el homing no se inicie inadvertidamente cuando el trote **articular** está habilitado.

Ejemplo: articulaciones sincronizadas 0,1 utilizando una secuencia negativa (-1) para homing sincronizado con un interruptor (allow_jjog) que selecciona una secuencia positiva (1) para trote **articular** individual antes de homing (código HAL parcial):

```
loadrt mux2           names=home_sequence_mux
loadrt conv_float_s32 names=home_sequence_s32
setp home_sequence_mux.in0 -1
setp home_sequence_mux.in1  1
addf home_sequence_mux servo-thread
addf home_sequence_s32 servo-thread
...
net home_seq_float <= home_sequence_mux.out
net home_seq_float => home_sequence_s32.in
net home_seq_s32   <= home_sequence_s32.out
net home_seq_s32   => ini.0.home_sequence
net home_seq_s32   => ini.1.home_sequence
...
# allow_jjog: pin creado por un panel virtual o interruptor hardware
net hsequence_select <= allow_jjog
net hsequence_select => home_sequence_mux.sel
net hsequence_select => motion.homing-inhibit
```

nota
Los pines HAL INI (como ini.N.home_sequence) no están disponibles hasta que milltask comienza, por lo que la ejecución de los comandos HAL anteriores debe ser diferida utilizando un archivo HAL postgui o un script [APLICACIÓN] APP= retrasado.

nota
La sincronización en tiempo real del trote articular para múltiples articulaciones requiere conexiones HAL adicionales para pines de tipo de trote del generador de pulso manual (MPG) (`joint.N.enable`, `joint.N.scale`, `joint.N.counts`).

Un ejemplo de simulación de configuración (gantry_jjog.ini) que demuestra el trote articular cuando se utilizan secuencias de casa negativas se encuentra en el directorio: configs/sim/axis/gantry/.

4.6. Configuración de torno

4.6.1. Plano predeterminado

Cuando el intérprete de LinuxCNC se escribió por primera vez, fue diseñado para fresadoras. Es por eso que el plano predeterminado es XY (G17). Un torno normal solo usa el plano XZ (G18). Para cambiar el plano predeterminado, colocar la siguiente línea en el archivo INI en la sección RS274NGC.

```
RS274NGC_STARTUP_CODE = G18
```

Lo anterior se puede sobrescribir en un programa de código G, por lo que siempre se deben configurar las cosas importantes en el preámbulo del archivo de código G.

4.6.2. Configuración INI

Las siguientes configuraciones INI son necesarias para el modo torno en Axis, además de, o reemplazando, la configuración normal en el archivo INI. Estas configuraciones históricas usan cinemáticas de identidad (trivkins) y *tres* articulaciones (0,1,2) correspondientes a coordenadas x, y, z. En estas configuraciones históricas se requiere la articulación 1 para el eje Y no utilizado, aunque no se usa. Las configuraciones de torno simuladas pueden usar estos ajustes históricos. GMOCCAPY también usa las configuraciones mencionadas, pero ofrece configuraciones adicionales; verificar la sección GMOCCAPY para más detalles.

```
[DISPLAY]
DISPLAY = axis
LATHE = 1
...

[KINS]
KINEMATICS = trivkins
JOINTS = 3

[TRAJ]
COORDINATES = X Z
...

[JOINT_0]
...
[JOINT_2]
```

```
...
[AXIS_X]
...
[AXIS_Z]
...
```

Con la incorporación de joints_axes, se puede hacer una configuración más simple con solo las dos articulaciones requeridas especificando trivkins con el parámetro *coordinates*=:

```
[DISPLAY]
DISPLAY = axis
LATHE = 1
...

[KINS]
KINEMATICS = trivkins coordinates=xz
JOINTS = 2

[TRAJ]
COORDINATES = X Z
...

[JOINT_0]
...
[JOINT_1]
...
[AXIS_X]
...
[AXIS_Z]
...
```

4.7. Inicio rápido de motores a pasos

Esta sección asume que una instalación estándar a partir de un LiveCD ha sido realizada. Posterior a la instalación es recomendable conectar la computadora al internet y esperar por la aparición del gestor de actualizaciones y obtener las ultimas actualizaciones para LinuxCNC y Ubuntu antes de continuar.

4.7.1. Prueba de latencia

La prueba de latencia determina cuánto tiempo le toma al procesador de su computadora responder a una solicitud de procesamiento. Algunos hardware pueden interrumpir el procesamiento, lo que puede provocar la pérdida de algunos pasos cuando se opera una máquina CNC. Esta es la primera cosa que se requiere hacer posterior a la instalación. Siga las instrucciones de aquí para correr la prueba de latencia.

4.7.2. Sherline

Si tienes una máquina marca Sherline, varias configuraciones predefinidas están disponibles. Estas configuraciones se encuentran en el menú principal CNC/EMC, de ahí selecciona la configuración Sherline que sea compatible con tu máquina y guarda una copia.

4.7.3. Xylotex

Si tienes una máquina marca Xylotex, puedes saltar las siguientes secciones para pasar directamente al Asistente de configuración de motores a pasos. LinuxCNC provee una configuración rápida para las máquinas Xylotex.

4.7.4. Información de la máquina

Obtenga la información sobre cada eje de su máquina.

Los tiempos de los controladores están en nanosegundos. Si usted no esta seguro con respecto a los tiempos de su controlador de motor a pasos algunos tiempos específicos para controladores populares están incluidos en el asistente de configuración de motores a pasos. Nótese que algunos controladores marca Gecko de nueva generación tienen tiempos que difieren con los originales. Para mas controladores, hay una lista mantenida por usuarios en el sitio wiki del LinuxCNC.

Eje	Tipo de controlador	Tiempo de paso (ns)	Espacio de paso (ns)	Sostenimiento de dir. (ns)	Configuración de dir. (ns)
X					
Y					
Z					

4.7.5. Información de disposición de pines

Reúne información sobre las conexiones de la máquina al puerto paralelo de la computadora.

Pin de salida	Tipo de función	Si es distinto	Pin de entrada	Tipo de función	Si es distinto
1	Salida E-Stop		10	Límite/casa de X	
2	Paso X		11	Límite/casa de Y	
3	Dirección de X		12	Límite/casa de Z	
4	Paso Y		13	Límite/casa de A	
5	Dirección de Y		15	Entrada de sonda	
6	Paso Z				
7	Dirección de Z				
8	Paso A				
9	Dirección de A				
14	Husillo sentido horario				
16	Husillo PWM				
17	Habilitar amplificador				

Nótese que cualquier pin no usado debe ser definido como Unused en el menú desplegable de configuración. Esto puede cambiarse posteriormente ejecutando de nuevo StepConf.

4.7.6. Información mecánica

Reúne información de pasos y engranaje. El resultado de esto es pasos por unidad de usuario que se usan para SCALE en el archivo INI.

Eje	Pasos/rev.	Micropasos	Dientes de motor	Dientes de tornillo de avance	Paso de tornillo
X					
Y					
Z					

- *Pasos por revolución* - indica cuantos pasos del motor le toma a la flecha del motor completar una revolución completa, un valor típico es 200 Pasos/Rev.
- *Micropasos* - este parámetro indica cuántos pasos debe dar el controlador para mover un paso completo del motor. Si no se utilizan micropasos este número será 1. En caso de utilizar micropasos el valor dependerá del hardware controlador de pasos.
- *Dientes del motor y del tornillo de avance* - se utilizan si se tiene alguna reducción mecánica (engranes, cadenas, bandas de tiempo, etc.) entre el motor y el tornillo de avance. De lo contrario, poner ambos a 1.
- *Paso del tornillo de avance* - es cuánto movimiento ocurre (en las unidades de usuario) cuando el tornillo de avance da una vuelta completa. Si se utilizan pulgadas entonces son pulgadas por revolución. Si se utilizan milímetros entonces son milímetros por revolución.

El resultado neto de la combinación de parámetros que se busca es cuántos pasos producidos por el software CNC le tomará al eje moverse linealmente una unidad de usuario (pulgadas o mm).

Ejemplo 4.1 Unidades pulgadas

```
Motor a pasos             = 200 pasos por revolución
Controlador de motor      =  10 micro pasos por paso
Dientes del motor         =  20
Dientes del tornillo guía =  40
Paso del tornillo guía    =   0.2000 pulgadas por revolución
```

A partir de la información anterior, el tornillo de avance se mueve 0.200 pulgadas por vuelta. - El motor da 2.000 vueltas por una vuelta del tornillo de avance. - El controlador necesita 10 micropasos de entrada para hacer al motor dar un paso. - El controlador necesita 2000 pasos para hacer que el motor dé una revolución.

Por lo tanto la escala necesitada es:

$$\frac{200 motor steps}{1 motor rev} \times \frac{10 microsteps}{1 motor step} \times \frac{2 motor revs}{1 leadscrew rev} \times \frac{1 leadscrew revs}{0.2000 inch} = \frac{20{,}000 microsteps}{inch}$$

Ejemplo 4.2 Unidades mm

```
    Motor a pasos             = 200 pasos por revolución
    Controlador de motor      =   8 micropasos por paso
    Dientes del motor         =  30
    Dientes del tornillo guía =  90
    Paso del tornillo guía    =   5.00 mm por revolución
```

A partir de la información anterior: - El tornillo de avance se mueve 5.00 mm por vuelta. - El motor da 3.000 vueltas por 1 vuelta del tornillo de avance. - El controlador necesita 8 micropasos de entrada para hacer al motor dar un paso. - El controlador necesita 1600 pasos para hacer que el motor dé una revolución.

Por lo tanto la escala necesitada es:

$$\frac{200\,full\,steps}{1\,rev} x \frac{8\,microsteps}{1\,step} x \frac{3\,revs}{1\,leadscrew\,rev} x \frac{1\,leadscrew\,rev}{5.00\text{mm}} = \frac{960\,steps}{1\,mm}$$

4.8. Configuración de motor a pasos

4.8.1. Introducción

La forma preferida de configurar una máquina de pasos estándar es con el Asistente de configuración de pasos. Ver el capítulo Asistente de configuración Stepconf .

Este capítulo describe algunas de las configuraciones más comunes para sistemas basados en pasos. Estos sistemas utilizan motores paso a paso con controladores que aceptan señales de paso y dirección.

Es una de las configuraciones más simples, porque los motores funcionan en bucle abierto (sin retroalimentación que provenga de los motores), pero el sistema debe ser configurado correctamente para que los motores no se detengan o pierdan pasos.

La mayor parte de este capítulo se basa en una configuración de muestra publicada junto con LinuxCNC. La configuración se llama stepper_inch, y se puede encontrar ejecutando el Selector de configuración.

4.8.2. Velocidad de paso máxima

Con la generación de pasos por software, la velocidad de paso máxima es un paso por cada dos BASE_PERIODs para salida de paso y dirección. La velocidad de pasos máxima solicitada es el producto de una *MAX_VELOCITY* del eje y su *INPUT_SCALE*. Si la velocidad de pasos solicitada no es alcanzable, ocurrirán los siguientes errores, particularmente durante trotes rápidos y movimientos G0.

Si el controlador de pasos puede aceptar entrada en cuadratura, use este modo. Con una señal de cuadratura, es posible un paso para cada BASE_PERIOD, duplicando la velocidad de paso máxima.

Los otros remedios son disminuir uno o más de: BASE_PERIOD (establecerlo demasiado bajo hará que la máquina deje de responder o incluso se bloquee), INPUT_SCALE (si puede seleccionar diferentes tamaños de paso en su controlador de pasos, cambiar las relaciones de poleas o el paso del tornillo de avance), o MAX_VELOCITY y STEPGEN_MAXVEL.

Si no hay una combinación válida de BASE_PERIOD, INPUT_SCALE y MAX_VELOCITY aceptable, entonces considerar usar la generación de pasos por hardware (como con el Controlador de pasos universal compatible con LinuxCNC, las tarjetas Mesa y otros).

4.8.3. Asignación de pines

Uno de los principales defectos en EMC fue que no se podía especificar la asignación de pines sin recompilar el código fuente. LinuxCNC es mucho más flexible, y ahora (gracias a la Capa de Abstracción de Hardware o HAL) se puede especificar fácilmente qué señal va a dónde. Ver el Principios básicos de HAL para más información sobre HAL.

Tal como se describe en la Introducción y el tutorial de HAL, dentro del HAL tenemos señales, pines y parámetros.

nota
Mostramos un solo eje para ser concisos; todos los demás son similares.

Lo relevante para nuestra asignación de pines es:

```
señales: Xstep, Xdir y Xen
pines: parport.0.pin-XX-out y parport.0.pin-XX-in
```

Dependiendo de lo que se haya elegido en el archivo INI se utiliza standard_pinout.hal o xylotex_pinout.hal. Estos son dos archivos que le indican a HAL cómo vincular las diferentes señales y pines. Después investigaremos standard_pinout.hal.

4.8.3.1. Asignación de pines estándar HAL

Este archivo contiene varios comandos HAL, y generalmente se ve así:

```
# archivo de configuración de pines estándar para steppers de 3 ejes
# usando un puerto paralelo para E/S
#
# primero carga el controlador parport
loadrt hal_parport cfg = ”0x0378”
#
# a continuación, conecta las funciones de parport a hilos
# primero, leer entradas
addf parport.0.read base-thread 1
# al último, escribir salidas
addf parport.0.write base-thread -1
#
# finalmente, conectar los pines físicos a las señales
net Xstep => parport.0.pin-03-out
net Xdir => parport.0.pin-02-out
net Ystep => parport.0.pin-05-out
net Ydir => parport.0.pin-04-out
net Zstep => parport.0.pin-07-out
net Zdir => parport.0.pin-06-out

# crear una señal para el loopback de estop
net estop-loop iocontrol.0.user-enable-out iocontrol.0.emc-enable-in

# crear señales para el loopback de carga de herramientas
net tool-prep-loop iocontrol.0.tool-prepare iocontrol.0.tool-prepare
net tool-change-loop iocontrol.0.tool-change iocontrol.0.tool-changed

# conectar el pin del controlador de movimiento ”spindle on” a un pin físico
net spindle-on spindle.0.on => parport.0.pin-09-out

###
### Puede usar algo como esto para habilitar unidades chopper cuando la máquina está ↩
    ENCENDIDA
### la señal Xen se define en core_stepper.hal
###

# net Xen => parport.0.pin-01-out

###
```

```
### Si se desea activo bajo para este pin, invertirlo así:
###

# setp parport.0.pin-01-out-invert 1

###
### Un interruptor de casa de muestra en el eje X (eje 0). Hacer una señal
### y vincular el pin parport entrante a la señal, luego vincular la señal
### al pin de entrada del interruptor de casa del eje 0 de LinuxCNC
###

# net Xhome parport.0.pin-10-in => joint.0.home-sw-in

###
### ¿todos los interruptores de casa compartidos en un pin del puerto paralelo?
### de acuerdo, conectar la misma señal a todos los ejes, pero asegurando
### establecer HOME_IS_SHARED y HOME_SEQUENCE en el archivo INI.
###

# net homeswitches <= parport.0.pin-10-in
# net homeswitches => joint.0.home-sw-in
# net homeswitches => joint.1.home-sw-in
# net homeswitches => joint.2.home-sw-in

###
### Muestra de interruptores de límite separados en el eje X (eje 0)
###

# net X-neg-limit parport.0.pin-11-in => joint.0.neg-lim-sw-in
# net X-pos-limit parport.0.pin-12-in => joint.0.pos-lim-sw-in

###
### Al igual que en el ejemplo de los interruptores de casa compartidos, se pueden conectar ↩
     juntos
### interruptores de límite.  Tener cuidado si se activa uno, LinuxCNC se detendrá pero no  ↩
    puede saber
### qué interruptor/eje se ha disparado.  Hay que tener precaución en la recuperación de  ↩
    esta
### posición extrema para evitar una parada dura.
###

# net Xlimits parport.0.pin-13-in => joint.0.neg-lim-sw-in joint.0.pos-lim-sw-in
```

Las líneas que comienzan con # son comentarios, y su único propósito es guiar al lector a través del archivo.

4.8.3.2. Descripción general

Hay un par de operaciones que se ejecutan cuando se ejecuta/interpreta standard_pinout.hal:

- El controlador Parport se carga (ver el capítulo Controlador de puerto paralelo para más detalles).
- Las funciones de lectura y escritura del controlador parport se asignan al hilo base [5].
- Las señales de paso y dirección para los ejes X, Y, Z se vinculan a los pines en el parport.
- Se conectan más señales de E /S (loopback de estop, loopback del cambiador de herramientas).
- Se define una señal de husillo y se vincula a un pin parport.

[5] El subproceso más rápido en la configuración de LinuxCNC, generalmente su código se ejecuta cada pocas decenas de microsegundos.

4.8.3.3. Cambiar standard_pinout.hal

Si se desea cambiar el archivo standard_pinout.hal, todo lo que se necesita es un editor de texto. Abrir el archivo y localizar las partes que se desean cambiar.

Si se quiere, por ejemplo, cambiar los pines de señales de paso y dirección para el eje X , todo lo que se necesita hacer es cambiar el número en el nombre *parport.0.pin-XX-out*:

```
net Xstep parport.0.pin-03-out
net Xdir parport.0.pin-02-out
```

se puede cambiar a:

```
net Xstep parport.0.pin-02-out
net Xdir parport.0.pin-03-out
```

o básicamente cualquier otro pin *out* que guste.

Sugerencia: asegurarse de no tener más de una señal conectada al mismo pin.

4.8.3.4. Cambio de polaridad de una señal

Si el hardware externo espera una señal de ”activo bajo”, configurar el parámetro *-invert* correspondiente. Por ejemplo, para invertir la señal de control del husillo:

```
setp parport.0.pin-09-invert TRUE
```

4.8.3.5. Agregar control de velocidad PWM al husillo

Si el husillo puede ser controlado por una señal PWM, usar el componente *pwmgen* para crear la señal:

```
loadrt pwmgen output_type=0
addf pwmgen.update servo-thread
addf pwmgen.make-pulses base-thread
net spindle-speed-cmd spindle.0.speed-out => pwmgen.0.value
net spindle-on spindle.0.on => pwmgen.0.enable
net spindle-pwm pwmgen.0.pwm => parport.0.pin-09-out
setp pwmgen.0.scale 1800 # Change to your spindle’s top speed in RPM
```

Esto supone que la respuesta del controlador del husillo a PWM es simple: 0 % PWM da 0 RPM, 10 % PWM da 180 RPM, etc. Si hay un mínimo PWM que se requiere para que el husillo gire, siga el ejemplo en configuración de muestra *nist-lathe* para usar un componente *scale*.

4.8.3.6. Agregar una señal de habilitación

Algunos amplificadores (controladores) requieren una señal de habilitación antes de aceptar ordenes de movimiento de los motores. Por esta razón ya hay señales definidas llamadas *Xen*, *Yen*, *Zen*.

Para conectarlas usar el siguiente ejemplo:

```
net Xen parport.0.pin-08-out
```

Puede tener un solo pin que habilite todas los controladores o varios, dependiendo de la configuración que tenga. Tener en cuenta, sin embargo, que generalmente cuando falla un eje, todas los demás controladores también se desactivarán, por lo que tener una sola señal/pin de habilitación para todas las unidades es una práctica común.

4.8.3.7. Botón externo ESTOP

El archivo standard_pinout.hal supone que no hay un botón ESTOP externo. Para más información sobre un E-Stop externo, ver la página del manual de estop_latch.

4.9. Diagnóstico de motores a pasos

Muchas veces lo que se obtiene no es lo que espera; pero todo es cuestión de experiencia. Aprender de la experiencia aumenta la comprensión del todo. La mejor manera de diagnosticar problemas es dividir y conquistar. Con esto quiero decir que si se puede eliminar la mitad de las variables de la ecuación, cada vez se encontrará el problema más rápido. En el mundo real esto no siempre es el caso, pero generalmente es una buena manera de comenzar.

4.9.1. Problemas comunes

4.9.1.1. El motor se mueve solo un paso

La razón más común en una nueva instalación para que un motor a pasos haga esto es que se han intercambiado las señales de paso y dirección. Si se presionan las teclas de trote hacia adelante y hacia atrás alternadamente, y el motor se mueve un paso cada vez, y en la misma dirección, ahí está la prueba.

4.9.1.2. Los motores no se mueven en absoluto

Muchos controladores tienen un pin de activación o necesitan una bomba de carga para activar su salida.

4.9.1.3. Distancia incorrecta

Si se le ordena al eje que se mueva una distancia específica y no se mueve tal distancia, entonces la configuración de escala es incorrecta.

4.9.2. Mensajes de error

4.9.2.1. Error de seguimiento

El concepto de error de seguimiento es extraño cuando se habla de motores de pasos. Como son sistemas de lazo abierto, no hay retroalimentacion de posición para hacerle saber si realmente está fuera de rango. LinuxCNC calcula si puede mantener el cumplimiento del movimiento solicitado y, si no, entonces da un error de seguimiento. Los errores de seguimiento generalmente son el resultado de uno de los siguientes problemas en sistemas paso a paso.

- FERROR demasiado pequeño
- MIN_FERROR demasiado pequeño
- MAX_VELOCITY demasiado rápida
- MAX_ACCELERATION demasiado rápida
- BASE_PERIOD demasiado largo

- Holgura mecánica añadida a un eje

Cualquiera de los anteriores puede hacer que los pulsos en tiempo real no puedan mantenerse conforme a la tasa de pasos solicitada. Esto puede suceder si no se ejecutó la prueba de latencia el tiempo suficiente para obtener un buen número para el Asistente Stepconf, o si se configura la Velocidad máxima o la Aceleración máxima demasiado alta.

Si se agregó holgura mecánica, se debe aumentar STEPGEN_MAXACCEL hasta que duplique MAX_ACCELE en la sección AXIS del archivo INI para cada eje que se le agregó. LinuxCNC utiliza una ”aceleración adicional” en una reversa para tener en cuenta la holgura mecánica. Sin corrección de holgura, la aceleración del generador de pasos será solo un pequeño porcentaje mayor que la aceleración planificada del movimiento.

4.9.2.2. Error RTAPI

Cuando se reciba este error:

```
RTAPI: ERROR: Retraso inesperado en tiempo real en la tarea n
```

Rtapi genera este error basándose en una indicación de RTAI de que se violó el tiempo limite. Suele ser una indicación de que BASE_PERIOD en la sección [EMCMOT] del archivo ini está configurado demasiado bajo. Se debería correr la prueba de latencia durante un período prolongado de tiempo para ver si hay algún retraso que cause este problema. Si usaste el Asistente Stepconf, ejecútalo de nuevo y prueba el jitter del período base nuevamente, y ajusta Base Period Maximum Jitter en la página Información Básica de la máquina. Es posible que tengas que dejar la prueba en funcionamiento durante un período prolongado de tiempo para encontrar si algún hardware causa problemas intermitentes.

LinuxCNC rastrea el número de ciclos de CPU entre invocaciones de hilos en tiempo real. Si algún elemento del hardware está causando demoras o los hilos en tiempo real se configuran demasiado rápido, se obtendrá este error.

---

nota
Este error solo se muestra una vez por sesión. Si se tuviera un BASE_PERIOD demasiado bajo, se obtendrían cientos de miles de mensajes de error por segundo si se mostrara más de uno.

4.9.3. Pruebas

4.9.3.1. Tiempos de pasos

Si se observa un eje que termina en una ubicación incorrecta durante movimientos múltiples, es probable que no se tengan los tiempos de mantenimiento de dirección o de pasos correctos para los controladores. Cada cambio de dirección puede estar perdiendo un paso o más. Si los motores se saturan, también es posible que se tenga MAX_ACCELERATION o MAX_VELOCITY demasiado alto para ese eje.

El siguiente programa probará la configuración correcta del eje Z. Copia el programa a tu directorio ~/emc2/nc_files y asígnale un nombre como TestZ.ngc o similar. Pon a cero la máquina con Z = 0.000 en la parte superior de la mesa. Carga y ejecuta el programa. Hará 200 movimientos de ida y vuelta de 0.5 a 1”. Si tienes un problema de configuración, encontrarás que la posición final no terminará en las 0.500” que la ventana del eje muestra. Para probar otro eje, simplemente reemplaza la Z con la letra del eje a probar en las líneas G0.

```
(programa de prueba para ver si el eje Z pierde posición)
(msg, prueba 1 de configuración del eje Z)
G20 #1000=100 (bucle 100 veces)
```

```
(este bucle tiene retrasos después de los movimientos)
(prueba configuraciones de velocidad y aceleración)
o100 while [#1000]
   G0 Z1.000
   G4 P0.250
   G0 Z0.500
   G4 P0.250
   #1000 = [#1000 - 1]
o100 endwhile
(msg, prueba 2 de la configuración del eje Z, S para continuar)
M1 (para aquí)
#1000=100 (bucle 100 veces)
(el siguiente ciclo no tiene demoras después de los movimientos)
(prueba los tiempos de retención de dirección del controlador y también la aceleración  ↩
    máxima)
o101 while [#1000]
   G0 Z1.000
   G0 Z0.500
   #1000 = [#1000 - 1]
o101 endwhile
(msg, Listo ... Z debe estar exactamente .5” sobre la mesa)
M2
```

4.10. Programas filtro

4.10.1. Introducción

La mayoría de las pantallas de LinuxCNC tienen la capacidad de pasar archivos cargados por un *programa de filtro* o usar el programa filtro para crear código G. Este filtro puede realizar cualquier tarea deseada: algo tan simple como asegurarse que el archivo termina con *M2*, o algo tan complicado como generar código G a partir de una imagen.

4.10.2. Configurar el INI para programas filtro

La sección *[FILTER]* del archivo INI controla cómo funcionan los filtros. Primero, para cada tipo de archivo, se escribe una línea *PROGRAM_EXTENSION*. Luego, se especifica el programa a ejecutar para cada tipo de archivo. Este programa recibe el nombre del archivo de entrada como su primer argumento, y debe escribir código rs274ngc en la salida estándar. Esta salida es lo que se mostrará en el área de texto, se previsualizará en el área de visualización y será ejecutado por LinuxCNC con *Ejecutar*. Las líneas siguientes agregan soporte para el convertidor *image-to-gcode* incluido en LinuxCNC:

```
[FILTER]
PROGRAM_EXTENSION = .png,.gif Imagen de profundidad en escala de grises
png = image-to-gcode
gif = image-to-gcode
```

También es posible especificar un intérprete:

```
PROGRAM_EXTENSION = .py Script Python
py = python
```

De esta manera, cualquier script de Python se puede abrir, y su salida es tratada como código G. Un ejemplo de tal script está disponible en *nc_files/holecircle.py*. Este script crea código G para perforar una serie de agujeros a lo largo de la circunferencia de un círculo.

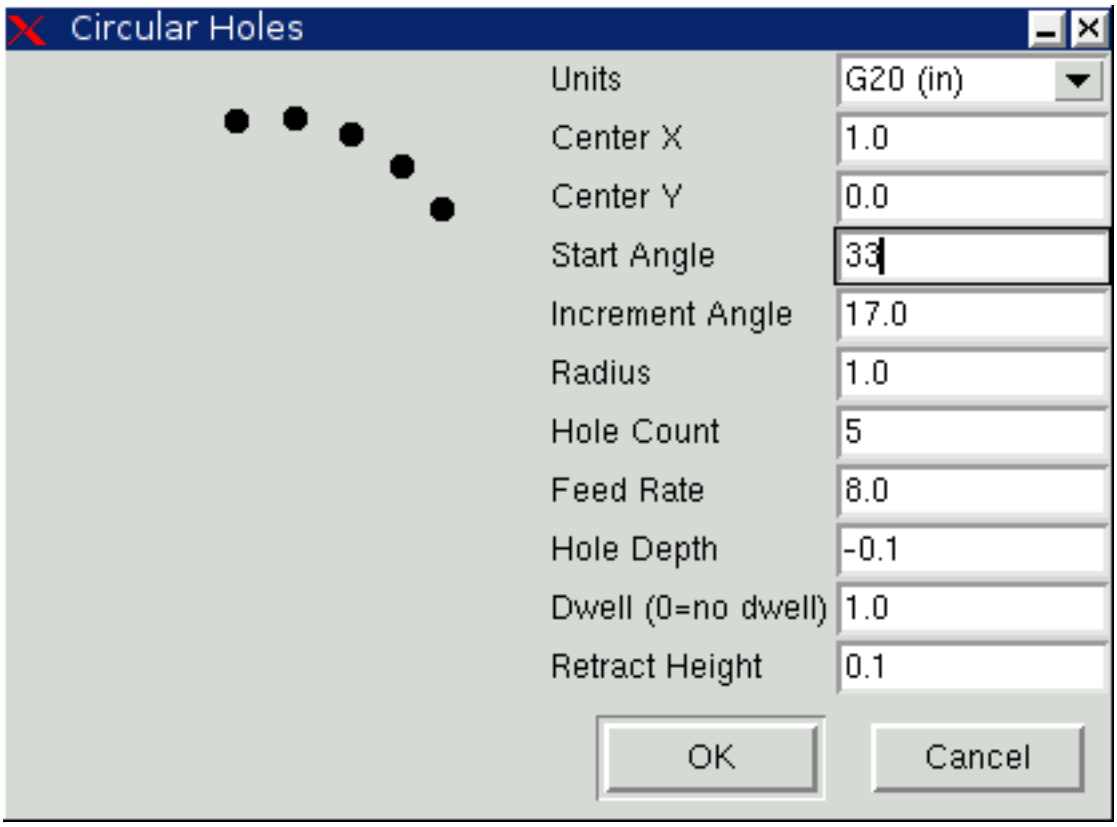


Figura 4.9: Agujeros circulares

Si un programa filtro envía líneas a stderr de la forma:

```
FILTER_PROGRESS=10
```

Establecerá la barra de progreso de la pantalla en el porcentaje dado (10 en este caso). Esta característica debería utilizarse en cualquier filtro que se ejecute durante mucho tiempo.

4.10.3. Crear programas filtro basados en Python

Aquí está un ejemplo muy básico de la mecánica de filtrado: Cuando se ejecuta mediante una pantalla de LinuxCNC que ofrezca filtrado por programa, producirá y escribirá una línea de código G cada 100ma de segundo en la salida estándar. También enviará un mensaje de progreso al flujo de error estándar de UNIX. Si hubiera un error publicará un mensaje de error y saldrá con un código de salida 1.

```
import time
import sys

for i in range(0,100):
    try:
        # simula tiempo de cálculo
        time.sleep(.1)

        # emite una línea código G
        print('G0 X1', file=sys.stdout)

        # actualiza el progreso
```

```
        print('FILTER_PROGRESS={}'.format(i), file=sys.stderr)
    except:
        # Esto provoca un mensaje de error
        print('Error; pero esto es solo una prueba', file=sys.stderr)
        raise SystemExit(1)
```

Aquí está un programa similar que en realidad puede filtrar. Coloca una ventana de diálogo PyQt5 con un botón de cancelación. Luego lee el programa línea por línea y las pasa a la salida estándar. Mientras transcurre, actualiza cualquier proceso escuchando la salida de error estándar.

```
#!/usr/bin/env python3

import sys
import os
import time

from PyQt5.QtWidgets import (QApplication, QDialog, QDialogButtonBox,
                             QVBoxLayout,QDialogButtonBox)
from PyQt5.QtCore import QTimer, Qt

class CustomDialog(QDialog):

    def __init__(self, path):
        super(CustomDialog, self).__init__(None)
        self.setWindowFlags(self.windowFlags() | Qt.WindowStaysOnTopHint)
        self.setWindowTitle("Prueba de filtrado con GUI")

        QBtn = QDialogButtonBox.Cancel

        self.buttonBox = QDialogButtonBox(QBtn)
        self.buttonBox.rejected.connect(self.reject)

        self.layout = QVBoxLayout()
        self.layout.addWidget(self.buttonBox)
        self.setLayout(self.layout)

        self.line = 0
        self._percentDone = 0

        if not os.path.exists(path):
            print("La ruta: '{}' no existe:".format(path), file=sys.stderr)
            raise SystemExit(1)

        self.infile = open(path, "r")
        self.temp = self.infile.readlines()

        # calcula el porcentaje del intervalo de actualización
        self.bump = 100/float(len(self.temp))

        self._timer = QTimer()
        self._timer.timeout.connect(self.process)
        self._timer.start(100)

    def reject(self):
        # Esto da un mensaje de error
        print('Solicitaste cancelar antes de terminar.', file=sys.stderr)
        raise SystemExit(1)

    def process(self):
        try:
            # obtiene la siguiente línea de código
            codeLine = self.temp[self.line]
```

```
            # procesa la línea de algún modo

            # saca el código procesado
            print(codeLine, file=sys.stdout)
            self.line +=1

            # actualiza el progreso
            self._percentDone += self.bump
            print('FILTER_PROGRESS={}'.format(int(self._percentDone)), file=sys.stderr)

            # si acabó termina sin error/mensaje de error
            if self._percentDone >= 99:
                print('FILTER_PROGRESS=-1', file=sys.stderr)
                self.infile.close()
                raise SystemExit(0)

        except Exception as e:
            # Esto da un mensaje de error
            print(('Sucedió algo malo:',e), file=sys.stderr)
            # esto señaliza que el mensaje de error debería mostrarse
            raise SystemExit(1)

if __name__ == "__main__":
    if (len(sys.argv)>1):
        path = sys.argv[1]
    else:
        path = None
    app = QApplication(sys.argv)
    w = CustomDialog(path=path)
    w.show()
    sys.exit( app.exec_() )
```

Capítulo 5

HAL (Hardware Abstraction Layer - Capa de Abstracción de Hardware)

5.1. Introducción a HAL

LinuxCNC trata la interacción con hardware. Pero pocos usuarios tienen exactamente las mismas especificaciones de hardware, similares mas no las mismas. E incluso para el exactamente mismo hardware, pueden haber maneras diferentes de usarlo, digamos para materiales diferentes o fresados diferentes, las cuales necesitarán adaptaciones para controlar un sistema que ya esté en funcionamiento. Fue necesaria una abstracción para hacer más fácil la configuración de LinuxCNC para una amplia variedad de dispositivos de hardware. Al nivel más alto, podría ser una manera simple de permitir que un número de *bloques de construcción* se carguen y se interconecten para ensamblar un sistema complejo.

Este capítulo da una introducción a la Capa de Abstracción de Hardware. Notarás que muchos de los bloques de construcción son de hecho, controladores de dispositivos hardware. Sin embargo, HAL puede hacer más que solo configurar controladores.

5.1.1. Descripción general

La Capa de Abstracción de Hardware (o con una referencia a la película 2001: Odisea del espacio sólo ”HAL”) es un software para

- proporcionar la infraestructura para la comunicación con y entre los muchos componentes de software y hardware del sistema.
- opcionalmente procesar y/o alterar esa información como fluya de componente a componente.

En sí mismo, este Middleware es agnóstico sobre su aplicación en CNC. Una búsqueda en internet, por ejemplo, arrojó una aplicación astronómica para controlar telescopios usando LinuxCNC. Motores mueven el telescopio en la posición correcta, y se necesita saber cómo mapear actividad del motor con el efecto de ese posicionamiento con el mundo real. Tal sincronización de posiciones de motor con posiciones del mundo real es una reminiscencia de lo que las máquinas CNC, o naves espaciales, necesitan hacer.

Cualquier controlador de máquina necesita saber:

- acerca de su estado interno y cómo se mapea esto en el entorno (coordenadas de máquina, estado de interruptores/reguladores),

- cómo se espera que los actuadores cambien ese estado,
- cómo permitir actualizaciones del estado interno por medio de sensores (codificadores, sondas).

La HAL consiste de partes (referidas como ”componentes”) que

- están conectados entre sí, p. ej. para actualizar datos de posición o para que el algoritmo de planificación le diga a los motores sobre el paso siguiente.
- pueden saber cómo comunicarse con el hardware,
- pueden simplemente procesar datos entrantes y proporcionar datos de salida para otros componentes,
- son siempre ejecutados periódicamente, ya sea
  - con una frecuencia muy alta o con unos cuantos microsegundos (µs) de tiempo de ejecución, llamado hilo base, p. ej. para
    1. darle a un motor de pasos un impulso a dar el paso, o para
    2. leer la posición que presenta un codificador.
  - con una frecuencia baja cada milisegundo (ms), p. ej. para
    1. ajustar la planeación de los movimientos siguientes para terminar una instrucción en código G.
  - como componentes en tiempo no-real ”espacio de usuario” que corren en un ”ciclo principal” tal y como otro software, y pueden ser interrumpidos o retrasados cuando el resto del sistema esta ocupado o sobrecargado.

En conjunto, HAL permite

1. programar una máquina que el programador no conoce directamente, pero puede confiar en una interfaz de programación con un efecto en la máquina bien especificado. Esa interfaz puede usarse para
   - decirle a la máquina qué hacer
   - escuchar lo que la máquina quiera decir sobre el estado en el que está.
2. Abstracciones verticales: El integrador de sistema humano de tal máquina usa HAL
   - para describir que busca la máquina y cómo lo busca y qué cable controla cuál motor que controla cuál eje.
   - La descripción de la máquina, las interfaces del programador y la interfaz del usuario de alguna manera se ”encuentran” en esa capa de abstracción.
3. Abstracciones horizontales:
   - No todas las máquina tienen todos los tipos de características
   - Fresadoras, tornos y robots comparten muchas
     - características (motores, articulaciones, …),
     - algoritmos de planificación para sus movimientos.

HAL no tiene interacción directa con el usuario. Pero se han proporcionado múltiples interfaces que permiten manipular a la HAL

- desde la línea de comandos usando el comando ”halcmd”.
- desde secuencias de comandos en Python y

- desde dentro de programas en C/C++,

pero ninguna de esas interfaces es *HAL en sí*.

HAL en sí mismo no es un programa, consiste en una o más listas de programas cargados (los componentes) que se ejecutan periódicamente (en secuencia estricta), y en un área de memoria compartida que usan esos componentes para intercambiar datos. La secuencia de comandos HAL principal se ejecuta solo una vez al arrancar la máquina, preparando los hilos de tiempo real y las ubicaciones de memoria compartida, cargando los componentes y preparando enlaces de datos entre ellos (las ”señales” y los ”pines”).

En principio, múltiples máquinas pueden compartir una HAL común para permitirles interoperar, sin embargo, la implementación actual de LinuxCNC esta limitada a un solo intérprete y un solo módulo de Tarea. Actualmente, casi siempre es un intérprete de código G y ”milltask” (el cual se encontró que también funciona bien para tornos y adecuadamente para robots), pero esos módulos son seleccionables en tiempo de carga. Con un interés creciente en el control de múltiples máquinas cooperantes, es probable que sobrepasar esta limitación sea uno de los pasos principales a abordar en desarrollos futuros de LinuxCNC.

HAL yace en el corazón de LinuxCNC y se usa y/o extiende por todas las partes de LinuxCNC, incluyendo las GUIs. El intérprete de código G (o lenguaje alternativo) sabe cómo interpretar el código G y lo traduce en operaciones de máquina al disparar señales en HAL. El usuario puede consultar la HAL de varias maneras para obtener información sobre su estado, el cual también representa el estado de la máquina. Durante el desarrollo de la versión 2.9, las GUIs aún son, por poco, una excepción a la regla y pueden saber algo que (sin necesitarlo) la HAL no.

5.1.2. Comunicación

Lo especial de la HAL es que puede comunicarse muy rápido

- con otros programas, pero en particular
- con sus componentes que típicamente corren en uno de los hilos de tiempo real.

Y mientras está en comunicación, la parte de LinuxCNC a la que le habla no necesita preparase para la comunicación: Todas esas acciones se realizan asíncronamente, es decir, ningún componente interrumpe su ejecución normal para recibir una señal, y las señales pueden enviarse inmediatamente, es decir, una aplicación puede esperar hasta que llegue un mensaje en particular -como una señal de activación, pero no necesita prepararse para recibir ese mensaje.

El sistema de comunicación

- representa y controla a todo el hardware conectado al sistema,
- inicia y detiene otros programas de comunicación.

La comunicación con el hardware de la máquina en sí, la realiza con los respectivos componentes HAL dedicados.

La HAL es un espacio compartido en el cual todas de las muchas partes que constituyen LinuxCNC intercambian información. Ese espacio dispone pines que son identificados por un nombre, aunque un ingeniero de LinuxCNC pueda preferir la asociación con un pin de un circuito electrónico. Esos pines pueden acarrear valores numéricos y lógicos, boleanos, de punto flotante y enteros con y sin signo. También hay un (relativamente nuevo) tipo de pin llamado hal_port destinado para flujos de bytes, y un framework para intercambiar datos mas complejos llamado hal_stream (el cual usa un área de memoria compartida privada, en lugar de un pin HAL). Los últimos dos tipos se usan relativamente infrecuentemente.

Con la HAL puedes enviar una señal a ese pin nombrado. Cada parte de la HAL puede leer ese pin que contiene ese valor de la señal. Eso hasta que una nueva señal se envía al mismo pin nombrado para sustituir el valor previo. El sistema central de intercambio de mensajes de la HAL es agnóstico de CNC, pero la HAL viene con un gran número de componentes que saben mucho de CNC y presentan esa información mediante pines. Hay pines que representan

- información estática sobre la máquina
- el estado actual de la máquina
  - interruptores de límite
  - posiciones contadas por motores a pasos o medidas por codificadores
- recipientes para instrucciones
  - control manual de posición de la máquina (trote o ”jogging”)
  - posiciones que el motor a pasos debería tomar después

En una analogía con cables electrónicos, los pines pueden cablearse, de tal manera que el valor que cambia en un pin sirve como entrada de otro pin. Los componentes HAL preparan tales pines de entrada y salida, y por lo tanto disparados automáticamente.

Componentes HAL Las muchas partes ”expertas” de LinuxCNC son típicamente implementadas como *componentes* de HAL, conceptualmente también referidos como *módulos*. Esos expertos implementados en computadora leen perpetuamente desde la HAL sobre un estado que la máquina debería esmerarse en alcanzar, y comparan ese estado deseado con el estado en el que esta actualmente la máquina. Cuando hay una diferencia entre lo que debería ser y el estado actual, es cuando se realizan algunas acciones para reducir tal diferencia, mientras se escriben actualizaciones perpetuamente de los estados actuales de vuelta al espacio de datos de la HAL.

Hay componentes especializados en cómo hablarle a motores a pasos, y otros componentes saben cómo controlar servos. En un nivel más alto, algunos componentes saben cómo están dispuestos los ejes de la máquina en 3D y otros saben cómo realizar un movimiento suave desde un punto en el espacio a otro. Tornos, fresadoras y robots diferirán en el componente de LinuxCNC que este activo, es decir, el que es cargado por un archivo de configuración HAL para esa máquina. Después de todo, dos máquinas pueden verse muy diferentes por haber sido construidas para propósitos distintos, pero cuando ambas usan servo motores, pueden usar el mismo componente servo de HAL.

Origen del incentivo para mover En el nivel mas bajo (cercano al hardware), p. ej. para motores de pasos, la descripción de un estado de ese motor es muy intuitiva: es el número de pasos en una dirección particular. Una diferencia entre la posición deseada y la real se traduce en un movimiento. Velocidades, aceleración y otros parámetros pueden estar internamente limitados dentro del componente mismo, o pueden estar opcionalmente limitados por componentes río arriba. (Por ejemplo, en la mayoría de los casos los valores de posición del eje momento a momento enviados a los componentes generadores de pasos ya han sido limitados y conformados para adecuarse a los límites de máquina configurados o a la velocidad de alimentación actual.)

Cualquier línea de código G es interpretada y dispara un conjunto de rutinas que en turno saben cómo comunicarse con componentes de la capa intermedia, p. ej. para crear un círculo.

Pines y señales HAL tiene un lugar especial en el corazón de sus programadores por la forma en que se representa el flujo de datos entre módulos. Cuando programadores tradiciones piensan en variables, direcciones o puertos de E/S, HAL refiere ”pines”. Y esos pines se conectan o se les asignan valores mediante señales. Muy similar a como un ingeniero eléctrico conectaría cables entre pines de componentes de una fresadora, un ingeniero HAL establece el flujo de datos entre pines de instancias de módulo.

Las GUIs de LinuxCNC (AXIS, GMOCCAPY, Touchy, etc.) representarán los estados de algunos pines (como los interruptores de límite) pero también existen otras herramientas gráficas para solucionar problemas y configuración: Halshow, Halmeter, Halscope y Halreport.

El resto de esta introducción presenta

- la sintaxis de cómo se conectan los pines de distintos componentes en los archivos de configuración de HAL, y
- software para inspeccionar los valores de los pines
  - en cualquier momento dado,
  - desarrollándose sobre el tiempo.

5.1.3. Diseño del sistema HAL

.HAL se basa en técnicas de diseño de sistemas tradicionales.

HAL se basa en los mismos principios que se usan para diseñar sistemas y circuitos de hardware, por lo que es útil examinar primero esos principios. Cualquier sistema, incluida una máquina CNC, consiste en componentes interconectados. Para una máquina CNC, esos componentes podrían ser el controlador principal, servoamplificadores o controladores de pasos, motores, codificadores, interruptores de límite, volantes con pulsadores (los populares MPG), quizás un VFD para el husillo, un PLC para manejar un cambiador de herramientas, etc. El constructor de la máquina debe seleccionar, montar y conectar estas piezas para formar un sistema completo.

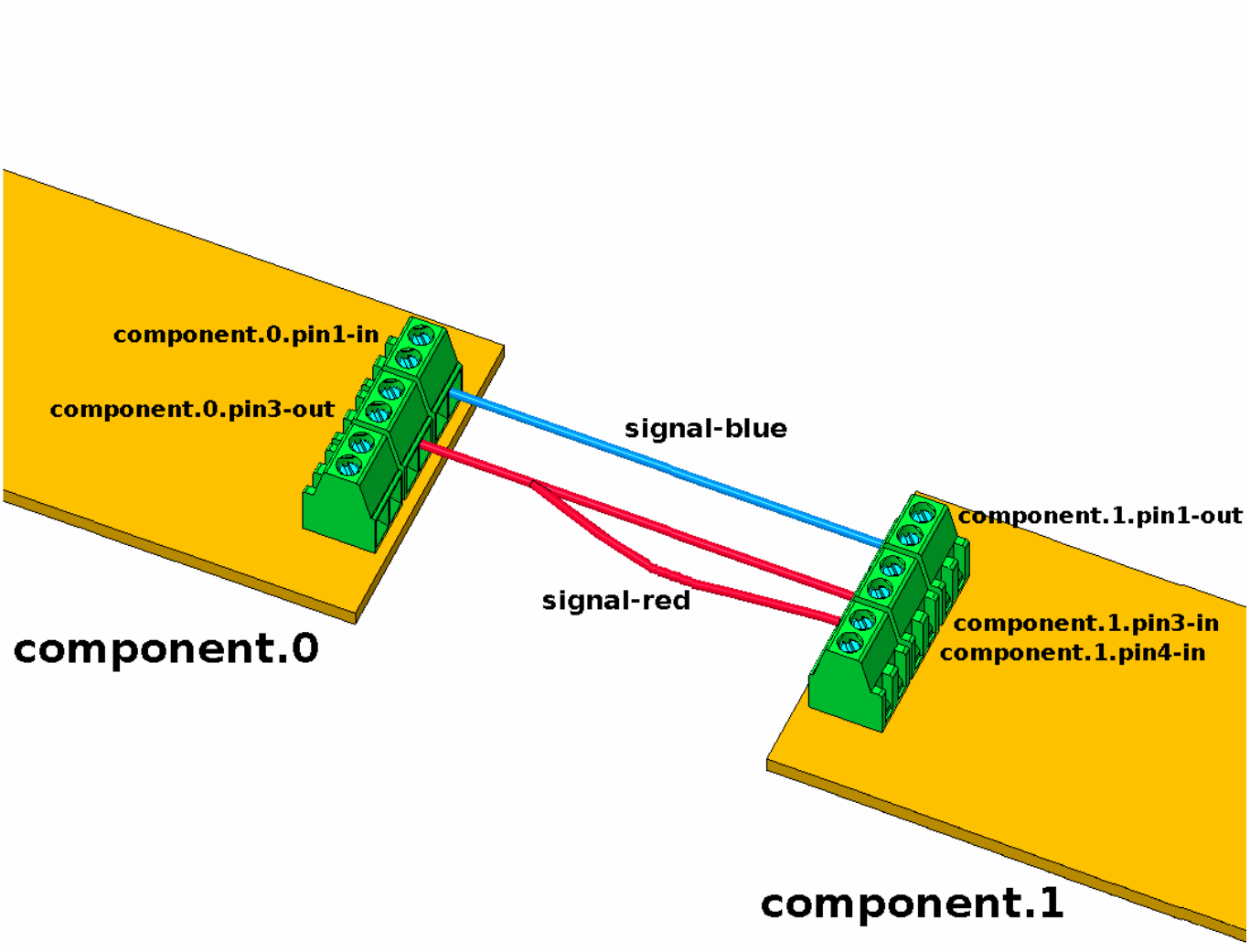


Figura 5.1: Concepto de HAL - Conexiones como en un circuito eléctrico.

La figura se transcribe a código HAL de esta manera:

```
net signal-blue    component.0.pin1-in      component.1.pin1-out
net signal-red     component.0.pin3-out     component.1.pin3-in     component.1.pin4-in
```

5.1.3.1. Selección de componentes

El constructor de una máquina no necesita preocuparse de cómo funciona cada componente individualmente. Los trata como cajas negras. Durante la etapa de diseño, él decide qué componentes va a usar: motores a pasos o servos, cuál marca de servo-amplificador, qué tipo de interruptores de límite y cuántos, etc. Las decisiones del integrador sobre qué componentes específicos usar se basarán en lo que hace ese componente y las especificaciones suministradas por el fabricante de dispositivo. El tamaño de un motor y la carga que debe mover afectará a la elección del amplificador necesario para manejarlo. La elección del amplificador puede afectar al tipo de retroalimentación necesario y a las señales de velocidad o posición que deben enviarse al propio amplificador desde el controlador.

De igual forma, en el mundo HAL el integrador debe decidir qué componentes HAL son necesarios. Por lo general, cada tarjeta de interfaz requerirá un controlador. Pueden ser necesarios componentes adicionales para la generación de pulsos de pasos por software, funcionalidad PLC, y una amplia variedad de otras tareas.

5.1.3.2. Diseño de interconexión

El diseñador de un sistema hardware no solo selecciona los componentes; también decide cómo interconectarlos. Cada caja negra tiene terminales, quizás solo dos como en un simple interruptor, o docenas para una unidad servo o un PLC. Los componentes necesitan estar interconectados. Los motores se conectan a los servo-amplificadores, los interruptores de límite se conectan al controlador, y así sucesivamente. Conforme el constructor de la máquina trabaja en el diseño, crea un diagrama de cableado completo que muestra cómo deben estar interconectados todos los componentes.

Cuando se usa HAL, los componentes están interconectados por señales. El diseñador debe decidir qué señales son necesarias y qué deberían conectar.

5.1.3.3. Implementación

Una vez que se completa el diagrama de cableado, es hora de construir la máquina. Las piezas deben ser adquiridas y montadas, y luego son interconectadas de acuerdo con el diagrama de cableado. En un sistema físico, cada interconexión es un trozo de cable con características determinadas, que debe conectarse a las terminales apropiadas.

HAL proporciona una serie de herramientas para ayudar a *construir* un sistema HAL. Algunas de las herramientas te permiten *conectar* (o desconectar) un solo *cable*. Otras herramientas te permiten guardar una lista completa de todos los componentes, cables y otra información sobre el sistema, para que pueda ser *reconstruido* con un solo comando.

5.1.3.4. Pruebas

Muy pocas máquinas funcionan bien la primera vez. Mientras hace las pruebas pertinentes, el constructor puede usar un tester para ver si un interruptor de límite está funcionando o para medir la tensión de CC que va a un servomotor. Puede conectar un osciloscopio para verificar la sintonización de un controlador, o para buscar ruido eléctrico. Puede encontrar un problema que requiera cambios en el diagrama de cableado; tal vez un componente debe estar conectado de manera diferente o ser reemplazado con algo completamente diferente.

HAL proporciona los equivalentes en software de un voltímetro, un osciloscopio, un generador de señales y otras herramientas necesarias para probar y ajustar un sistema. Los mismos comandos utilizados para construir el sistema se pueden usar para hacer cambios según sea necesario.

5.1.3.5. Resumen

Este documento está dirigido a personas que ya saben cómo hacer este tipo de integración de sistemas de hardware, pero que no saben cómo conectar el hardware a LinuxCNC. Vea la sección Ejemplo de arranque remoto en la documentación de Ejemplos UI HAL.

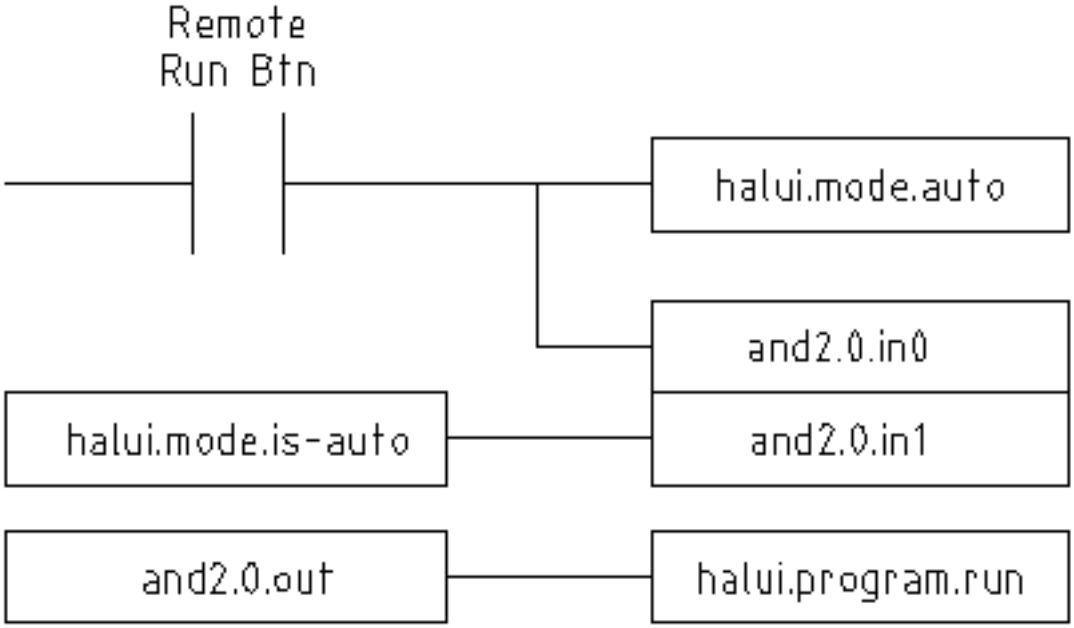


Figura 5.2: Ejemplo de arranque remoto (Esquema)

El diseño de hardware tradicional, como se ha descrito arriba, termina en el limite del control principal. Fuera del control (la PC) hay un montón de cajas relativamente simples, conectadas entre sí para hacer lo que sea necesario. En el interior, el control es un gran misterio; una enorme caja negra que esperamos funcione.

HAL extiende este método de diseño de hardware tradicional al interior de la gran caja negra. Hace de los controladores de dispositivos, e incluso algunas partes internas del controlador, cajas negras más pequeñas que se pueden interconectar, e incluso reemplazar al igual que el hardware externo. HAL permite al *diagrama de cableado del sistema* mostrar parte del controlador interno, en lugar de una gran caja negra. Y lo más importante, permite que el integrador pruebe y modifique el controlador utilizando los mismos métodos que usaría en el resto del hardware.

A la mayoría de los integradores le son familiares términos como motores, amplificadores y codificadores. Cuando hablamos de usar un cable blindado extra flexible de ocho conductores para conectar un codificador a la placa de entrada del servo en la computadora, el lector comprende de inmediato lo que es y le lleva a la pregunta *¿qué tipo de conectores necesitaré en cada extremo?*. El mismo tipo de pensamiento es esencial para HAL, pero ese tren de pensamiento específico puede tardar un poco en encarrilarse. Usando HAL, las palabras pueden parecer un poco extrañas al principio, pero el concepto de trabajar desde una conexión a la siguiente es el mismo.

Esta idea de extender el diagrama de cableado al interior del controlador es la adoptada por HAL. Si se siente cómodo con la idea de interconectar cajas negras de hardware, es probable que tenga pocos problemas al utilizar HAL para interconectar cajas negras de software.

5.1.4. HAL: Conceptos

Esta sección es un glosario que define los términos clave HAL, aunque es un poco diferente de un glosario tradicional porque estos términos no estan ordenados alfabéticamente sino por su relación o flujo dentro de la manera de hacer las cosas con HAL.

Componente:: Cuando hablamos de diseño de hardware, nos referimos a las piezas individuales como *partes*, *bloques de construcción*, *cajas negras*, etc. El equivalente en HAL es un *componente* o *componente HAL*. Este documento usa *componente HAL* cuando pueda haber confusión con otros tipos de componentes, pero normalmente usa solamente *componente*. Un componente HAL es una pieza de software con entradas, salidas y comportamiento bien definidos, que puede ser instalado e interconectado como se necesite. ++ Muchos componentes HAL modelan el comportamiento de una parte tangible de una máquina, y un **pin** puede, de hecho, estar destinado a conectarse a un **pin físico** del dispositivo con el que se comunicará, de ahí los nombres. Pero más a menudo, este no es el caso. Imagina la modernización de un torno/fresadora manual. Lo que implementa LinuxCNC es cómo la máquina se presenta por sí misma al mundo exterior, y es secundario si la implementación de cómo trazar un círculo ya esta implementada en la máquina o es proporcionada por LinuxCNC. Y es común agregar botones a la actualización imaginaria que **señaliza** una acción, como una parada de emergencia. LinuxCNC y la máquina se convierten en uno. Y todo eso mediante la HAL.

Parámetro:: Muchos componentes de hardware tienen mandos de ajustes que no están conectados a ningún otro componente, pero que deben tener alguna forma de acceso. Por ejemplo, los servoamplificadores a menudo tienen potenciómetros, para permitir ajustes de afinación, y puntos de prueba donde puede ser conectado un medidor u osciloscopio para ver los resultados del ajuste. Los componentes HAL también pueden tener tales cosas, que se conocen como *parámetros*. Hay dos tipos de parámetros: los parámetros de entrada son equivalentes a potenciómetros; son valores que pueden ser ajustados por el usuario, y permanecen fijos si no son ajustados de nuevo. Los parámetros de salida no pueden ser ajustados por el usuario; son equivalente a los puntos de prueba que permiten monitorizar las señales internas.

Pin:: Los componentes hardware tienen terminales que se utilizan para interconectarlos. El equivalente HAL es un *pin* o *pin HAL*. Se usa *pin HAL* cuando sea necesario para evitar confusiones. Todos los pines HAL tienen nombre y esos nombres de pines se usan para interconectarlos. Los pines HAL son entidades de software que existen solo dentro de la computadora.

Pin físico:: Muchos dispositivos de E/S tienen pines físicos reales o terminales que se conectan a hardware externo, por ejemplo, los pines del conector de puerto paralelo. Para evitar confusiones, estos se conocen como *pines físicos*. Son esas cosas que se *enchufan* en el mundo real.

---

nota
Podrías preguntarte qué relación hay entre los pines HAL, pines físicos y elementos externos como codificadores o tarjetas STG: estamos lidiando aquí con interfaces del tipo traslado/conversión de datos.

Señal:: En una máquina física, las terminales reales de los componentes en hardware están interconectados por cables. El equivalente HAL de un cable es una *señal* o *señal HAL*. Las señales HAL conectan los pines HAL según lo requiera el constructor de la máquina. Las señales HAL pueden ser desconectadas y reconectadas a voluntad (incluso cuando la máquina está en funcionamiento).

Tipo:: Cuando se usa hardware real, no se conectaría una salida de relé de 24 voltios a una entrada analógica +/-10 V de un servoamplificador. Los pines HAL tienen las mismas restricciones, las cuales se basan en su tipo. Tanto los pines como las señales tienen tipos, y las señales solo se pueden conectar a pines del mismo tipo. Actualmente hay 4 tipos, que son los siguientes:

+ - bit - un único valor, VERDADERO/FALSO o ENCENDIDO/APAGADO o 1/0 - float - un valor de punto flotante de 64 bits, con aproximadamente 53 bits de resolución y más de 1000 bits de rango dinámico. - u32 - un entero sin signo de 32 bits, los valores legales son de 0 a 4,294,967,295 - s32 - un entero de 32 bits con signo, los valores legales son de -2,147,483,647 a +2,147,483,647 - u64 - un entero sin

signo de 64 bits, los valores legales son de 0 a 18,466,744,073,709,551,615 - s64 - un entero de 64 bits con signo, los valores legales son de -9,223,372,036,854,775,808 a +9,223,372,036,854,775,807

Función:: Los componentes de hardware reales tienden a actuar de inmediato ante una entrada. Por ejemplo, si el voltaje de entrada a un servoamplificador cambia, la salida también cambia automáticamente. Sin embargo, los componentes de software no pueden actuar *automáticamente*. Cada componente tiene un código específico que se debe ejecutar para hacer lo que ese componente se supone debe hacer. En algunos casos, ese código simplemente se ejecuta como parte del componente. Por otra parte, en la mayoría de los casos, especialmente en componentes en tiempo real, el código debe ejecutarse en una secuencia específica y en intervalos específicos. Por ejemplo, las entradas deben leerse antes de realizar cálculos con los datos de entrada, y las salidas no deben escribirse hasta que los cálculos estén hechos. En estos casos, el código está disponible para el sistema en la forma de una o más *funciones*. Cada función es un bloque de código que realiza una acción específica. El integrador de sistema puede usar *hilos* para programar una serie de funciones que serán ejecutadas en un orden particular y en intervalos de tiempo específicos.

Hilo:: Un *hilo* es una lista de funciones que se ejecuta en intervalos específicos como parte de una tarea en tiempo real. Cuando un hilo es creado por primera vez, tiene un intervalo de tiempo específico (período), pero no tiene funciones. Las funciones se pueden agregar al hilo, y se ejecutarán en orden cada vez que se ejecuta el hilo.

Como ejemplo, supongamos que tenemos un componente de puerto paralelo llamado hal_parport. Ese componente define uno o más pines HAL para cada pin físico. Los pines se describen en la sección de documentación de ese componente: sus nombres, cómo se relaciona cada pin HAL con pines físicos, si están invertidos, si puedes cambiar su polaridad, etc. Pero eso por sí solo no hace que los datos de los pines HAL lleguen a los pines físicos. Se necesita código para hacer eso, y ahí es donde entran en escena las funciones. El componente hal_parport necesita al menos dos funciones: una para leer los pines de entrada física y actualizar los pines HAL, y otra para tomar datos de los pines HAL y escribirlos en pines de salida física. Ambas funciones son parte del controlador parport.

5.1.5. Componentes HAL

Cada componente HAL es una pieza de software con entradas, salidas y comportamiento bien definidos, que puede ser instalado e interconectado como sea necesario. La sección Lista de componentes HAL enlista todos los componentes disponibles y una descripción breve de lo que hace cada uno.

5.1.6. Problemas de sincronización en HAL

A diferencia de los modelos de cableado físico entre cajas negras en el que se dijo que está basado HAL, la simple conexión de dos pines con una señal HAL queda aún lejos de la acción del caso físico.

La lógica de relés consiste en relés conectados entre sí, y cuando el contacto se abre o se cierra, la corriente fluye (o se detiene) inmediatamente. Otras bobinas pueden cambiar de estado, etc., y todo simplemente *ocurre*. Pero un PLC de lógica de escalera no funciona de esa manera. Usualmente, en un solo pase a través de la escalera, cada escalón se evalúa en el orden en que aparece, y solo una vez por pase. Un ejemplo perfecto es una escalera de un solo escalón, con un contacto NC en serie con una bobina. El contacto y la bobina pertenecen al mismo relé.

Si se tratara de un relé convencional, tan pronto como se energiza la bobina, los contactos comienzan a abrirse y la des-energiza. Eso significa que los contactos cerrarán de nuevo, etc, etc. El relé se convierte en un zumbador.

Con un PLC, si la bobina está DESACTIVADA y el contacto está cerrado cuando el PLC comienza a evaluar el peldaño, cuando termina ese pase, la bobina estará ENCENDIDA. El hecho de que al encender la bobina se abre el contacto alimentado es ignorado hasta el próximo pase. En la siguiente pasada, el PLC ve que el contacto está abierto y des-energiza la bobina. Entonces el relé sigue cambiando rápidamente entre encendido y apagado, pero a un ritmo determinado por la frecuencia con la que el PLC evalúa el escalón.

En HAL, la función es el código que evalúa el(los) escalón(es). De hecho, la versión HAL en tiempo real de ClassicLadder exporta una función que hace exactamente eso. Entre tanto, un hilo es lo que permite la ejecución de la función en intervalos de tiempo específicos. Al igual que puedes elegir que un PLC evalúe todos sus escalones cada 10 ms, o cada segundo, puedes definir hilos HAL con períodos diferentes.

Lo que distingue a un hilo de otro *no* es lo que el hilo hace - eso está determinado por las funciones que son conectadas a él. La distinción real es simplemente qué tan seguido se ejecuta un hilo.

En LinuxCNC, puedes tener un hilo de 50 µs y un hilo de 1 ms. Estos se crearían con base en BASE_PERIOD y SERVO_PERIOD; los tiempos reales dependen de los valores en tu archivo INI.

El siguiente paso es decidir qué debe hacer cada hilo. Algunas de estas decisiones son las mismas en (casi) cualquier sistema LinuxCNC. Por ejemplo, la función motion-command-handler siempre se agrega al hilo servo.

Otras conexiones serán hechas por el integrador. Estas pueden incluir conectar las funciones de lectura de codificador de un controlador STG y las de escritura de un DAC al hilo servo, o enganchar la función de StepGen al hilo base, junto con la(s) función(es) parport para escribir pasos en el puerto.

5.2. HAL Basics

This document provides a reference to the basics of HAL.

5.2.1. HAL Commands

More detailed information can be found in the man page for halcmd: run *man halcmd* in a terminal window.

To see the HAL configuration and check the status of pins and parameters use the HAL Configuration window on the Machine menu in AXIS. To watch a pin status open the Watch tab and click on each pin you wish to watch and it will be added to the watch window.

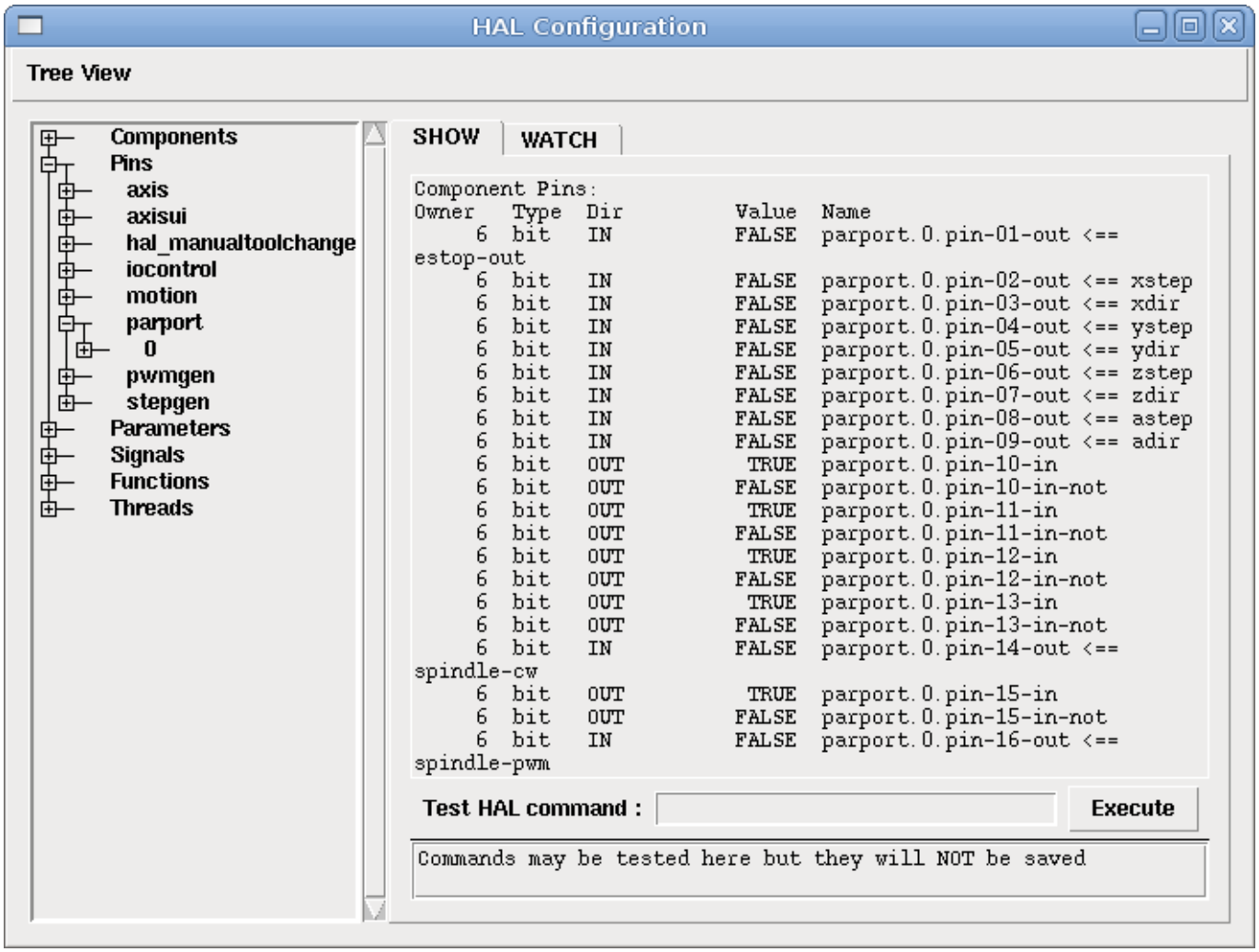


Figura 5.3: HAL Configuration Window

5.2.1.1. loadrt

The command `loadrt` loads a real time HAL component. Real time component functions need to be added to a thread to be updated at the rate of the thread. You cannot load a non-realtime component into the realtime space.

loadrt Syntax and Example

```
loadrt <componente> <opciones>
loadrt mux4 count=1
```

5.2.1.2. addf

The `addf` command adds a function to a real-time thread. If the StepConf wizard was used to create the configuration, two threads have been created (``base-thread`` and ``servo-thread``).

`addf` adds function *functname* to thread *threadname*. Default is to add the function in the order they are in the file. If *position* is specified, adds the function to that spot in the thread. A negative *position* indicates the position with respect to the end of the thread. For example *1* is start of thread, *-1* is the end of the thread, *-3* is third from the end.

Es importante cargar algunas funciones en un orden determinado, como por ejemplo las funciones de lectura y escritura de puerto paralelo. El nombre de la función suele ser el nombre del componente más un número. En el siguiente ejemplo se carga el componente *or2* y `show function` muestra el nombre de la función or2.

```
$ halrun
halcmd: loadrt or2
halcmd: show function
Exported Functions:
Owner   CodeAddr  Arg       FP   Users  Name
 00004  f8bc5000  f8f950c8  NO       0   or2.0
```

You have to add a function from a HAL real time component to a thread to get the function to update at the rate of the thread. Usually there are two threads as shown in this example.

---

nota
The FP column and the `uses_fp` parameter are deprecated. All threads now unconditionally save and restore floating point state. The `fp`/`nofp` distinction will be removed in a future version.

```
$ halrun
halcmd: loadrt motmod base_period_nsec=55555 servo_period_nsec=1000000 num_joints=3
halcmd: show thread
Realtime Threads:
     Period  FP     Name               (     Time, Max-Time )
     995976  YES          servo-thread (        0,        0 )
      55332  YES           base-thread (        0,        0 )
```

- base-thread (the high-speed thread): This thread handles items that need a fast response, like making step pulses, and reading and writing the parallel port.
- servo-thread (the slow-speed thread): This thread handles items that can tolerate a slower response, like the motion controller, ClassicLadder, and the motion command handler.

addf Syntax and Example

```
addf <función> <cuerda>
addf mux4.0 servo-thread
```

5.2.1.3. loadusr

The command `loadusr` loads a non-realtime HAL component. Non-realtime programs are their own separate processes, which optionally talk to other HAL components via pins and parameters. You cannot load realtime components into non-realtime space.

Flags may be one or more of the following:

-W	to wait for the component to become ready. The component is assumed to have the same name as the first argument of the command.
-Wn <name>	to wait for the component, which will have the given <name>. This only applies if the component has a name option.
-w	Esperar a que el programa termine
-i	ignorar el valor de retorno del programa (con -w)
-n	name a component when it is a valid option for that component.

Syntax and Examples of loadusr

```
loadusr <componente> <opciones>
loadusr halui
loadusr -Wn spindle gs2_vfd -n spindle
```

Esto ultimo significa: *loadusr espera la carga del componente spindle, que es un componente gs2_vfd, nombrado como spindle*.

5.2.1.4. net

The command `net` creates a *connection* between a signal and one or more pins. If the signal does not exist net creates the new signal. This replaces the need to use the command newsig. The optional direction arrows `<=`, `=>` and `<=>` make it easier to follow the logic when reading a `net` command line and are not used by the net command. The direction arrows must be separated by a space from the pin names.

Syntax and Examples of net

```
net nombre-señal nombre-pin <flecha-opcional> <nombre-de-segundo-pin-opcional>
net home-x joint.0.home-sw-in <= parport.0.pin-11-in
```

In the above example `home-x` is the signal name, `joint.0.home-sw-in` is a *Direction IN* pin, `<=` is the optional direction arrow, and `parport.0.pin-11-in` is a *Direction OUT* pin. This may seem confusing but the in and out labels for a parallel port pin indicates the physical way the pin works not how it is handled in HAL.

A pin can be connected to a signal if it obeys the following rules:

- An IN pin can always be connected to a signal.
- An IO pin can be connected unless there’s an OUT pin on the signal.
- Un pin OUT se puede conectar solo si no hay otros pines OUT o IO en la señal.

The same *signal-name* can be used in multiple net commands to connect additional pins, as long as the rules above are obeyed.

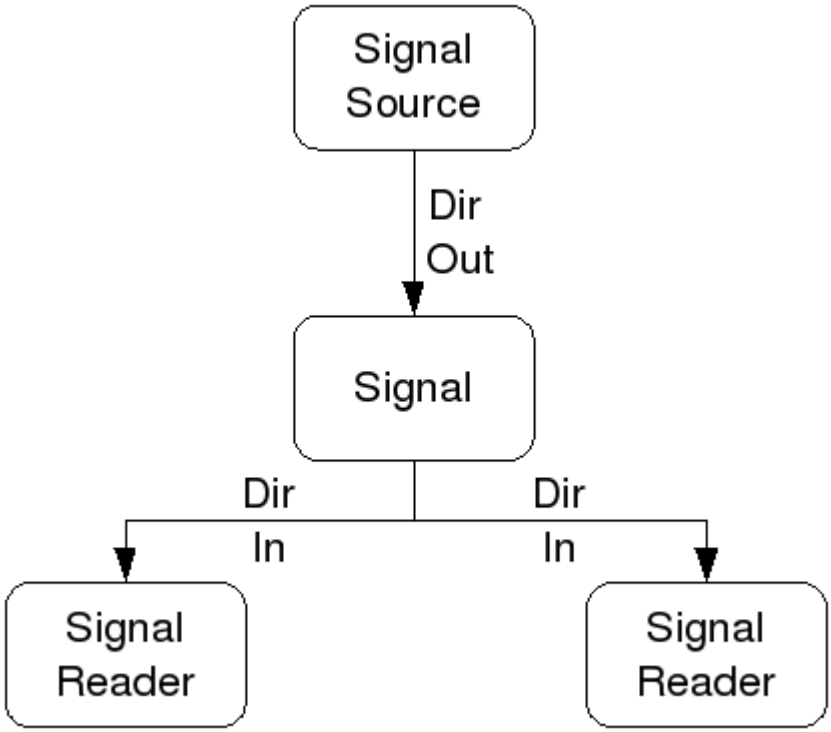


Figura 5.4: Signal Direction

This example shows the signal xStep with the source being `stepgen.0.out` and with two readers, `parport.0.pin-02-out` and `parport.0.pin-08-out`. Basically the value of `stepgen.0.out` is sent to the signal xStep and that value is then sent to `parport.0.pin-02-out` and `parport.0.pin-08-out`.

```
#   signal    source            destination          destination
net xStep stepgen.0.out => parport.0.pin-02-out parport.0.pin-08-out
```

Since the signal xStep contains the value of `stepgen.0.out` (the source) you can use the same signal again to send the value to another reader. To do this just use the signal with the readers on another line.

```
#   signal       destination2
net xStep => parport.0.pin-06-out
```

I/O pins An I/O pin like encoder.*N*.index-enable can be read or set as allowed by the component.

5.2.1.5. setp

The command `setp` sets the value of a pin or parameter. The valid values will depend on the type of the pin or parameter. It is an error if the data types do not match.

Some components have parameters that need to be set before use. Parameters can be set before use or while running as needed. You cannot use setp on a pin that is connected to a signal.

Syntax and Examples of setp

```
setp <nombre-pin/parámetro> <valor>
setp parport.0.pin-08-out TRUE
```

5.2.1.6. sets

The command `sets` sets the value of a signal.

Syntax and Examples of sets

```
sets <nombre-de-señal> <valor>
net mysignal and2.0.in0 pyvcp.my-led
sets mysignal 1
```

It is an error if:

- The signal-name does not exist
- If the signal already has a writer
- If value is not the correct type for the signal

5.2.1.7. unlinkp

The command `unlinkp` unlinks a pin from the connected signal. If no signal was connected to the pin prior running the command, nothing happens. The `unlinkp` command is useful for trouble shooting.

Syntax and Examples of unlinkp

```
unlinkp <nombre-de-pin>
unlinkp parport.0.pin-02-out
```

5.2.1.8. Obsolete Commands

The following commands are depreciated and may be removed from future versions. Any new configuration should use the net command. These commands are included so older configurations will still work.

The command `linksp` creates a *connection* between a signal and one pin.

Syntax and Examples of linksp

```
linksp <nombre-de-señal> <nombre-de-pin>
linksp X-step parport.0.pin-02-out
```

El comando `linksp` ha sido reemplazado por el comando `net`.

The command `linkps` creates a *connection* between one pin and one signal. It is the same as `linksp` but the arguments are reversed.

Syntax and Examples of linkps

```
linkps <nombre-de-pin> <nombre-de-señal>
linkps parport.0.pin-02-out X-Step
```

El comando `linkps` ha sido reemplazado por el comando `net`.

the command `newsig` creates a new HAL signal by the name <*signame*> and the data type of <*type*>. Type must be *bit*, *s32*, *u32*, *s64*, *u64* or *float*. Error if <*signame*> already exists.

Syntax and Examples of newsig

```
newsig <nombre-de-señal> <tipo>
newsig Xstep bit
```

More information can be found in the HAL manual or the man pages for `halrun`.

5.2.2. HAL Data

5.2.2.1. Bit

A bit value is an on or off.

- bit values = true or 1 and false or 0 (True, TRUE, true are all valid)

5.2.2.2. Float

A *float* is a floating point number. In other words the decimal point can move as needed.

- float values = a 64 bit floating point value, with approximately 53 bits of resolution and over 2^{10} (~ 1000) bits of dynamic range.

For more information on floating point numbers see:

https://en.wikipedia.org/wiki/Floating_point

5.2.2.3. s32

An *s32* number is a whole number that can have a negative or positive value.

- s32 values = integer numbers from -2147483648 to 2147483647

5.2.2.4. u32

A *u32* number is a whole number that is positive only.

- u32 values = integer numbers from 0 to 4294967295

5.2.2.5. s64

An *s64* number is a whole number that can have a negative or positive value.

- s64 values = integer numbers from -9,223,372,036,854,775,808 to +9,223,372,036,854,775,807

5.2.2.6. u64

A *u64* number is a whole number that is positive only.

- u64 values = integer numbers from 0 to 18,446,744,073,709,551,615

5.2.3. HAL Files

If you used the Stepper Config Wizard to generate your config you will have up to three HAL files in your config directory.

- *my-mill.hal* (if your config is named *my-mill*) This file is loaded first and should not be changed if you used the Stepper Config Wizard.
- *custom.hal* This file is loaded next and before the GUI loads. This is where you put your custom HAL commands that you want loaded before the GUI is loaded.
- *custom_postgui.hal* This file is loaded after the GUI loads. This is where you put your custom HAL commands that you want loaded after the GUI is loaded. Any HAL commands that use PyVCP widgets need to be placed here.

5.2.4. Parámetro

Two parameters are automatically added to each HAL component when it is created. These parameters allow you to scope the execution time of a component.

`.time`	Time is the number of CPU cycles it took to execute the function.
`.tmax`	Tmax is the maximum number of CPU cycles it took to execute the function.

`tmax` is a read/write parameter so the user can set it to 0 to get rid of the first time initialization on the function's execution time.

5.2.5. Basic Logic Components

HAL contains several real time logic components. Logic components follow a *Truth Table* that states what the output is for any given input. Typically these are bit manipulators and follow electrical logic gate truth tables.

For further components see HAL Components List or the man pages.

5.2.5.1. and2

The `and2` component is a two input and-gate. The truth table below shows the output based on each combination of input.

Sintaxis

```
and2 [count=N] | [names=name1[,name2...]]
```

Funciones

```
and2.n
```

Pines

```
and2.N.in0 (bit, in)
and2.N.in1 (bit, in)
and2.N.out (bit, out)
```

Cuadro 5.3: Truth Table of `and2`

in0	in1	out
False	False	False
True	False	False
False	True	False
True	True	True

5.2.5.2. not

The `not` component is a bit inverter.

Sintaxis

```
not [count=n] | [names=name1[,name2...]]
```

Funciones

```
not.all
not.n
```

Pines

```
not.n.in (bit, in)
not.n.out (bit, out)
```

Cuadro 5.4: Truth Table of `not`

in	out
True	False
False	True

5.2.5.3. or2

The `or2` component is a two input or-gate.

Sintaxis

```
or2[count=n] | [names=name1[,name2...]]
```

Funciones

```
or2.n
```

Pines

```
or2.n.in0 (bit, in)
or2.n.in1 (bit, in)
or2.n.out (bit, out)
```

Cuadro 5.5: or2 Truth Table

in0	in1	out
True	False	True
True	True	True
False	True	True
False	False	False

5.2.5.4. xor2

The `xor2` component is a two input xor (exclusive or)-gate.

Sintaxis

```
xor2[count=n] | [names=name1[,name2...]]
```

Funciones

```
xor2.n
```

Pines

```
xor2.n.in0 (bit, in)
xor2.n.in1 (bit, in)
xor2.n.out (bit, out)
```

Cuadro 5.6: xor2 Truth Table

in0	in1	out
True	False	True
True	True	False
False	True	True
False	False	False

5.2.6. Logic Examples

.Example using `and2`

```
loadrt and2 count=1
addf and2.0 servo-thread
net my-sigin1 and2.0.in0 <= parport.0.pin-11-in
net my-sigin2 and2.0.in1 <= parport.0.pin-12-in
net both-on parport.0.pin-14-out <= and2.0.out
```

In the above example one copy of `and2` is loaded into real time space and added to the servo thread. Next `pin-11` of the parallel port is connected to the `in0` bit of the and gate. Next `pin-12` is connected to the `in1` bit of the and gate. Last we connect the `and2` out bit to the parallel port `pin-14`. So following the truth table for `and2` if pin 11 and pin 12 are on then the output pin 14 will be on.

5.2.7. Conversion Components

5.2.7.1. weighted_sum

The weighted sum converts a group of bits into an integer. The conversion is the sum of the *weights* of the bits present plus any offset. It's similar to *binary coded decimal* but with more options. The *hold* bit interrupts the input processing, so that the *sum* value no longer changes.

Syntax for loading component weighted_sum

```
loadrt weighted_sum wsum_sizes=size[,size,...]
```

Creates groups of``weighted_sum``s, each with the given number of input bits (size).

To update the `weighted_sum`, the `process_wsums` must be attached to a thread.

Agrega process_wsums al servo-hilo

```
addf process_wsums servo-thread
```

Which updates the `weighted_sum` component.

In the following example, a copy of the AXIS HAL configuration window, bits *0* and *2* are TRUE, they have no offset. The weight (*weight*) of bit 0 is 1, that of bit 2 is 4, so the sum is 5.

Cuadro 5.7: Component pins of `weighted_sum`

Owner	Tipo	Dir	Valor	Name
10	bit	In	TRUE	`wsum.0.bit.0.in`
10	s32	I/O	1	`wsum.0.bit.0.weight`
10	bit	In	FALSE	`wsum.0.bit.1.in`
10	s32	I/O	2	`wsum.0.bit.1.weight`
10	bit	In	TRUE	`wsum.0.bit.2.in`
10	s32	I/O	4	`wsum.0.bit.2.weight`
10	bit	In	FALSE	`wsum.0.bit.3.in`
10	s32	I/O	8	`wsum.0.bit.3.weight`
10	bit	In	FALSE	`wsum.0.hold`
10	s32	I/O	0	`wsum.0.offset`
10	s32	Out	5	`wsum.0.sum`

5.3. HAL TWOPASS

5.3.1. TWOPASS

This section describes an option to have multiple load-commands for multiple instances of the same component at different positions in the file or among different files. Internally, this requires to read the HAL file twice, hence the name TWOPASS. Supported since LinuxCNC version 2.5, the TWOPASS processing of LinuxCNC configuration files helps with their modularization and readability. To recall, LinuxCNC configuration files are specified in a LinuxCNC INI file as `[HAL]HALFILE=filename`.

Normally, a set of one or more LinuxCNC configuration files must use a single, unique `loadrt` line to load a realtime component, which may create multiple instances of the component. For example, if you use a two-input AND gate component (and2) in three different places in your setup, you would need to have a single line somewhere to specify:

Example resulting in real-time components with default names and2.0, and2.1, and2.2.

```
loadrt and2 count=3
```

Configurations are more readable if you specify with the `names=` option for components where it is supported, e.g.:

Example load command resulting in explicitly named components aa, ab, ac.

```
loadrt and2 names=aa,ab,ac
```

It can be a maintenance problem to keep track of the components and their names, since when you add (or remove) a component, you must find and update the single loadrt directive applicable to the component.

El procesamiento TWOPASS se habilita al incluir un parámetro de archivo INI en la sección [HAL], donde "cualquier cadena" puede ser cualquier cadena no nula.

```
[HAL]

TWOPASS = anystring
```

Con TWOPASS habilitado, puede tener múltiples especificaciones como:

```
loadrt and2 names=aa
...
loadrt and2 names=ab,ac
...
loadrt and2 names=ad
```

These commands can appear in different HAL files. The HAL files are processed in the order of their appearance in the INI file, in multiple HALFILE assignments.

The TWOPASS option can be specified with options to add output for debugging (verbose) and to prevent deletion of temporary files (nodelete). The options are separated with commas.

Ejemplo

```
[HAL]
TWOPASS = on,verbose,nodelete
```

With TWOPASS processing, all [HAL]HALFILES are first read and multiple appearances of loadrt directives for each module are accumulated. Non-realtime components (loadusr) are loaded in order but no other LinuxCNC commands are executed in the initial pass.

---

nota
Non-realtime components should use the wait (-W) option to ensure the component is ready before other commands are executed.

After the initial pass, the realtime modules are loaded (loadrt) automatically

- with a number equal to the total number when using the *count*= option or
- with all of the individual names specified when using the *names*= option.

A second pass is then made to execute all of the other LinuxCNC instructions specified in the HALFILES. The addf commands that associate a component's functions with thread execution are executed in the order of appearance with other commands during this second pass.

While you can use either the *count*= or *names*= options, they are mutually exclusive — only one type can be specified for a given module.

TWOPASS processing is most effective when using the *names*= option. This option allows you to provide unique names that are mnemonic or otherwise relevant to the configuration. For example, if you use a derivative component to estimate the velocities and accelerations on each (x,y,z) coordinate, using the *count*= method will give arcane component names like ddt.0, ddt.1, ddt.2, etc.

Alternatively, using the *names*= option like:

```
loadrt ddt names=xvel,yvel,zvel
...
loadrt ddt names=xaccel,yaccel,zaccel
```

results in components sensibly named xvel, yvel, zvel, xaccel, yaccel, zaccel.

Many comps supplied with the distribution are created with the halcompile utility and support the *names*= option. These include the common logic components that are the glue of many LinuxCNC configurations.

User-created comps that use the halcompile utility automatically support the names= option as well. In addition to comps generated with the halcompile utility, numerous other comps support the names=option. Comps that support names= option include: at_pid, encoder, encoder_ratio, pid, siggen, and sim_encoder.

Two-step processing occurs before the GUI is loaded. When using a [HAL]POSTGUI_HALFILE, it is convenient to place all the [HAL]POSTGUI_HALFILE loadrt declarations for the necessary components in a preloaded HAL file.

Example of a HAL section when using a POSTGUI_HALFILE

```
[HAL]

TWOPASS = on
HALFILE = core_sim.hal
HALFILE = sim_spindle_encoder.hal
HALFILE = axis_manualtoolchange.hal
HALFILE = simulated_home.hal
HALFILE = load_for_postgui.hal <- loadrt lines for components in postgui.hal

POSTGUI_HALFILE = postgui.hal
HALUI = halui
```

5.3.2. Post GUI

Some GUIs support HAL files that are processed after the GUI is started in order to connect LinuxCNC pins that are created by the GUI. When using a postgui HAL file with TWOPASS processing, include all loadrt items for components added by postgui HAL files in a separate HAL file that is processed before the GUI. The `addf` commands can also be included in the file.

Ejemplo

```
[HAL]
TWOPASS = on
HALFILE = archivo_1.hal
...
HALFILE = archivo_n.hal
HALFILE = archivo_con_todas_las_cargas_para_postgui.hal
...
POSTGUI_HALFILE = el_archivo_postgui.hal
```

5.3.3. Excluding .hal files

TWOPASS processing converts *.hal* files to equivalent *.tcl* files and uses haltcl to find loadrt and addf commands in order to accumulate and consolidate their usage. Loadrt parameters that conform to the simple `names=` (or `count=`) parameters accepted by the HAL Component Generator (`halcompile`) are expected. More complex parameter items included in specialized LinuxCNC components may not be handled properly.

A *.hal* file may be excluded from TWOPASS processing by including a magic comment line anywhere in the *.hal* file. The magic comment line must begin with the string: `#NOTWOPASS`. Files specified with this magic comment are sourced by halcmd using the `-k` (keep going if failure) and `-v` (verbose) options.

This exclusion provision can be used to isolate problems or for loading any special LinuxCNC component that does not require or benefit from TWOPASS processing.

Ordinarily, the loadrt ordering of realtime components is not critical, but loadrt ordering for special components can be enforced by placing the such loadrt directives in an excluded file.

---

nota
While the order of loadrt directives is not usually critical, ordering of addf directives is often very important for proper operation of servo loop components.

Excluded HAL file example

```
$ cat twopass_excluded.hal
# The following magic comment causes this file to
# be excluded from twopass processing:
# NOTWOPASS

# debugging component with complex options:
loadrt mycomponent parm1=”abc def” parm2=ghi
show pin mycomponent

# ordering special components
loadrt component_1
loadrt component_2
```

---

nota
Case and whitespace within the magic comment are ignored. The loading of components that use `names=` or `count=` parameters (typically built by halcompile) should not be used in excluded files, as that would eliminate the benefits of TWOPASS processing. The LinuxCNC commands that create signals (`net`) and commands that establish execution order (`addf`) should not be placed in excluded files. This is especially true for addf commands since their ordering may be important.

5.3.4. Ejemplos

Examples of TWOPASS usage for a simulator are included in the directories:

```
configs/sim/axis/twopass/
configs/sim/axis/simtcl/
```

5.4. HAL Tutorial

5.4.1. Introducción

Configuration moves from theory to device — HAL device that is. For those who have had just a bit of computer programming, this section is the *Hello World* of the HAL.

`halrun` can be used to create a working system. It is a command line or text file tool for configuration and tuning. The following examples illustrate its setup and operation.

5.4.2. Halcmd

`halcmd` is a command line tool for manipulating HAL. A more complete man page exists for halcmd and installed together with LinuxCNC, from source or from a package. If LinuxCNC has been compiled as *run-in-place*, the man page is not installed but is accessible in the LinuxCNC main directory with the following command:

```
$ man -M docs/man halcmd
```

5.4.2.1. Notation

For this tutorial, commands for the operating system are typically shown without the prompt provided by the UNIX shell, i.e typically a dollar sign ($) or a hash/double cross (#). When communicating directly with the HAL through `halcmd` or `halrun`, the prompts are shown in the examples. The terminal window is in *Applications/Accessories* from the main Ubuntu menu bar.

Terminal Command Example - prompts

```
me@computer:~linuxcnc$ halrun
(will be shown like the following line)
halrun

(the halcmd: prompt will be shown when running HAL)
halcmd: loadrt counter
halcmd: show pin
```

5.4.2.2. Tab-completion

Your version of `halcmd` may include tab-completion. Instead of completing file names as a shell does, it completes commands with HAL identifiers. You will have to type enough letters for a unique match. Try pressing tab after starting a HAL command:

Tab-completion

```
halcmd: loa<TAB>
halcmd: load
halcmd: loadrt
halcmd: loadrt cou<TAB>
halcmd: loadrt counter
```

5.4.2.3. The RTAPI environment

RTAPI significa Interfaz de programación de Aplicaciones de Tiempo Real. Muchos componentes HAL funcionan en tiempo real, y todos los componentes HAL almacenan datos en memoria compartida para que los componentes en tiempo real puedan acceder a ella. El Linux normal no soporta programación en tiempo real o el tipo de memoria compartida que HAL necesita. Afortunadamente, hay sistemas operativos en tiempo real (RTOS) que proporcionan las extensiones necesarias para Linux. Lamentablemente, cada RTOS hace las cosas de manera un poco diferente.

To address these differences, the LinuxCNC team came up with RTAPI, which provides a consistent way for programs to talk to the RTOS. If you are a programmer who wants to work on the internals of LinuxCNC, you may want to study *linuxcnc/src/rtapi/rtapi.h* to understand the API. But if you are a normal person, all you need to know about RTAPI is that it (and the RTOS) needs to be loaded into the memory of your computer before you do anything with HAL.

5.4.3. A Simple Example

5.4.3.1. Loading a component

For this tutorial, we are going to assume that you have successfully installed the Live CD and, if using a RIP [1], invoke the *rip-environment* script to prepare your shell. In that case, all you need to do is load the required RTOS and RTAPI modules into memory. Just run the following command from a terminal window:

Loading HAL

```
cd linuxcnc
halrun
halcmd:
```

With the realtime OS and RTAPI loaded, we can move into the first example. Notice that the prompt is now shown as *halcmd:*. This is because subsequent commands will be interpreted as HAL commands, not shell commands.

For the first example, we will use a HAL component called *siggen*, which is a simple signal generator. A complete description of the *siggen* component can be found in the SigGen section of this Manual. It is a realtime component. To load the "`siggen`" component, use the HAL command `loadrt`.

Loading siggen

```
halcmd: loadrt siggen
```

5.4.3.2. Examining the HAL

Now that the module is loaded, it is time to introduce `halcmd`, the command line tool used to configure the HAL. This tutorial will introduce only a selection of halcmd features. For a more complete description try `man halcmd`, or see the reference in HAL Commands section of this document. The first halcmd feature is the *show* command. This command displays information about the current state of the HAL. To show all installed components:

Show Components with halrun/halcmd

```
halcmd: show comp

    Loaded HAL Components:
```

[1] Run In Place, when the source files have been downloaded to a user directory and are compiled and executed directly from there.

```
    ID     Type  Name                                PID   State
    3      RT    siggen                                    ready
    2      User  halcmd2177                          2177  ready
```

Since *halcmd* itself is also a HAL component, it will always show up in the list. The number after "`halcmd`" in the component list is the UNIX process ID. It is possible to run more than one copy of halcmd at the same time (in different terminal windows for example), so the PID is added to the end of the name to make it unique. The list also shows the *siggen* component that we installed in the previous step. The *RT* under *Type* indicates that *siggen* is a realtime component. The *User* under *Type* indicates it is a non-realtime component.

Next, let's see what pins `siggen` makes available:

Show Pins

```
halcmd: show pin

Component Pins:
Owner   Type   Dir        Value  Name
     3  float  IN             1  siggen.0.amplitude
     3  bit    OUT        FALSE  siggen.0.clock
     3  float  OUT            0  siggen.0.cosine
     3  float  IN             1  siggen.0.frequency
     3  float  IN             0  siggen.0.offset
     3  float  OUT            0  siggen.0.sawtooth
     3  float  OUT            0  siggen.0.sine
     3  float  OUT            0  siggen.0.square
     3  float  OUT            0  siggen.0.triangle
```

This command displays all of the pins in the current HAL. A complex system could have dozens or hundreds of pins. But right now there are only nine pins. Of these pins eight are floating point and one is bit (boolean). Six carry data out of the *siggen* component and three are used to transfer settings into the component. Since we have not yet executed the code contained within the component, some the pins have a value of zero.

The next step is to look at parameters:

Show Parameters

```
halcmd: show param

Parameters:
Owner   Type  Dir        Value   Name
     3  s32   RO             0   siggen.0.update.time
     3  s32   RW             0   siggen.0.update.tmax
```

The *show param* command shows all the parameters in the HAL. Right now, each parameter has the default value it was given when the component was loaded. Note the column labeled *Dir*. The parameters labeled *-W* are writable ones that are never changed by the component itself, instead they are meant to be changed by the user to control the component. We will see how to do this later. Parameters labeled *R-* are read only parameters. They can be changed only by the component. Finally, parameter labeled *RW* are read-write parameters. That means that they are changed by the component, but can also be changed by the user. Note: The parameters `siggen.0.update.time` and `siggen.0.update.tmax` are for debugging purposes and won't be covered in this section.

Most realtime components export one or more functions to actually run the realtime code they contain. Let's see what function(s) *siggen* exported:

Show Functions with halcmd`

```
halcmd: show funct

Exported Functions:
```

```
Owner   CodeAddr  Arg       FP   Users  Name
00003   f801b000  fae820b8  YES      0  siggen.0.update
```

The siggen component exported a single function. It is not currently linked to any threads, so *users* is zero [2].

5.4.3.3. Making realtime code run

To actually run the code contained in the function `siggen.0.update`, we need a realtime thread. The component called *threads* that is used to create a new thread. Lets create a thread called ”test-thread” with a period of 1 ms (1,000 µs or 1,000,000 ns):

```
halcmd: loadrt threads name1=test-thread period1=1000000
```

Let’s see if that worked:

Show Threads

```
halcmd: show thread

Realtime Threads:
     Period  FP     Name               (     Time, Max-Time )
     999855  YES    test-thread        (        0,        0 )
```

It did. The period is not exactly 1,000,000 ns because of hardware limitations, but we have a thread that runs at approximately the correct rate. The next step is to connect the function to the thread:

Add Function

```
halcmd: addf siggen.0.update test-thread
```

Up till now, we’ve been using `halcmd` only to look at the HAL. However, this time we used the `addf` (add function) command to actually change something in the HAL. We told *halcmd* to add the function `siggen.0.update` to the thread *test-thread*, and if we look at the thread list again, we see that it succeeded:

```
halcmd: show thread

Realtime Threads:
     Period  FP     Name               (     Time, Max-Time )
     999855  YES    test-thread        (        0,        0 )
                 1 siggen.0.update
```

There is one more step needed before the *siggen* component starts generating signals. When the HAL is first started, the thread(s) are not actually running. This is to allow you to completely configure the system before the realtime code starts. Once you are happy with the configuration, you can start the realtime code like this:

```
halcmd: start
```

Now the signal generator is running. Let’s look at its output pins:

```
halcmd: show pin

Component Pins:
Owner   Type  Dir         Value  Name
     3  float IN              1  siggen.0.amplitude
     3  bit   OUT         FALSE  siggen.0.clock
```

[2] CodeAddr and Arg fields were used during development and should probably disappear.

```
     3  float OUT    -0.1640929  siggen.0.cosine
     3  float IN              1  siggen.0.frequency
     3  float IN              0  siggen.0.offset
     3  float OUT    -0.4475303  siggen.0.sawtooth
     3  float OUT     0.9864449  siggen.0.sine
     3  float OUT            -1  siggen.0.square
     3  float OUT    -0.1049393  siggen.0.triangle
```

And let's look again:

```
halcmd: show pin

Component Pins:
Owner   Type  Dir         Value  Name
     3  float IN              1  siggen.0.amplitude
     3  bit   OUT         FALSE  siggen.0.clock
     3  float OUT     0.0507619  siggen.0.cosine
     3  float IN              1  siggen.0.frequency
     3  float IN              0  siggen.0.offset
     3  float OUT     -0.516165  siggen.0.sawtooth
     3  float OUT     0.9987108  siggen.0.sine
     3  float OUT            -1  siggen.0.square
     3  float OUT    0.03232994  siggen.0.triangle
```

We did two `show pin` commands in quick succession, and you can see that the outputs are no longer zero. The sine, cosine, sawtooth, and triangle outputs are changing constantly. The square output is also working, however it simply switches from +1.0 to -1.0 every cycle.

5.4.3.4. Changing Parameters

The real power of HAL is that you can change things. For example, we can use the `setp` command to set the value of a parameter. Let's change the amplitude of the signal generator from 1.0 to 5.0:

Set Pin

```
halcmd: setp siggen.0.amplitude 5
```

Check the parameters and pins again

```
halcmd: show param

Parameters:
Owner   Type  Dir         Value  Name
     3  s32   RO           1754  siggen.0.update.time
     3  s32   RW          16997  siggen.0.update.tmax

halcmd: show pin

Component Pins:
Owner   Type  Dir         Value  Name
     3  float IN              5  siggen.0.amplitude
     3  bit   OUT         FALSE  siggen.0.clock
     3  float OUT     0.8515425  siggen.0.cosine
     3  float IN              1  siggen.0.frequency
     3  float IN              0  siggen.0.offset
     3  float OUT      2.772382  siggen.0.sawtooth
     3  float OUT     -4.926954  siggen.0.sine
     3  float OUT             5  siggen.0.square
     3  float OUT      0.544764  siggen.0.triangle
```

Note that the value of parameter `siggen.0.amplitude` has changed to 5, and that the pins now have larger values.

5.4.3.5. Saving the HAL configuration

La mayor parte de lo que hasta ahora hemos hecho con `halcmd` es simplemente ver cosas con el comando *show*. Sin embargo, dos de los comandos realmente han cambiado cosas. Cuando se diseñan sistemas más complejos con HAL, utilizaremos muchos comandos para configurar las cosas tal como las queremos. HAL tienen la memoria de un elefante, y retendrá esa configuración hasta que lo apaguemos. Pero ¿que hay sobre la próxima vez?. No queremos ingresar manualmente un grupo de comandos cada vez que queremos usar el sistema.

Saving the configuration of the entire HAL with a single command.

```
halcmd: save

# components
loadrt threads name1=test-thread period1=1000000
loadrt siggen
# pin aliases
# signals
# nets
# parameter values
setp siggen.0.update.tmax 14687
# realtime thread/function links
addf siggen.0.update test-thread
```

The output of the `save` command is a sequence of HAL commands. If you start with an *empty* HAL and run all these commands, you will get the configuration that existed when the *save* command was issued. To save these commands for later use, we simply redirect the output to a file:

Save configuration to a file with halcmd

```
halcmd: save all saved.hal
```

5.4.3.6. Exiting halrun

When you're finished with your HAL session type `exit` at the "`halcmd:`" prompt. This will return you to the system prompt and close down the HAL session. Do not simply close the terminal window without shutting down the HAL session.

Exit HAL

```
halcmd: exit
```

5.4.3.7. Restoring the HAL configuration

To restore the HAL configuration stored in the file "saved.hal", we need to execute all of those HAL commands. To do that, we use "`-f _<file name>_`" which reads commands from a file, and "`-I`" (upper case i) which shows the halcmd prompt after executing the commands:

Run a Saved File

```
halrun -I -f saved.hal
```

Notice that there is not a "start" command in saved.hal. It's necessary to issue it again (or edit the file saved.hal to add it there).

5.4.3.8. Removing HAL from memory

If an unexpected shutdown of a HAL session occurs you might have to unload HAL before another session can begin. To do this type the following command in a terminal window.

Removing HAL

```
halrun -U
```

5.4.4. Halmeter

Puedes construir sistemas HAL muy complejos sin tener que utilizar una interfaz gráfica. Sin embargo, es muy satisfactorio ver gráficamente el resultado de tu trabajo. La primera y más simple herramienta GUI para HAL es halmeter. Es un programa muy simple, que es el equivalente HAL de un práctico multímetro (o medidor análogo para los amantes de lo antaño).

It allows to observe the pins, signals or parameters by displaying the current value of these entities. It is very easy to use application for graphical environments. In a console type:

```
halmeter
```

Two windows will appear. The selection window is the largest and includes three tabs:

- One lists all the pins currently defined in HAL,
- one lists all the signals,
- one lists all the parameters.

Click on a tab, then click on one of the items to select it. The small window will show the name and value of the selected item. The display is updated approximately 10 times per second. To free screen space, the selection window can be closed with the *Close* button. On the little window, hidden under the selection window at program launch, the *Select* button, re-opens the selection window and the *Exit* button stops the program and closes both windows.

It is possible to run several halmeters simultaneously, which makes it possible to visualize several items at the same time. To open a halmeter and release the console by running it in the background, run the following command:

```
halmeter &
```

It is possible to launch halmeter and make it immediately display an item. For this, add *pin|sig|par[am] name* arguments on the command line. It will display the signal, pin, or parameter *name* as soon as it will start. If the indicated item does not exist, it will start normally.

Finally, if an item is specified for display, it is possible add *-s* in front of pin|sig|param to tell halmeter to use an even smaller window. The item name will be displayed in the title bar instead of below the value and there will be no button. This is useful for displaying a lot of halmeters in a small space.

We will use the siggen component again to check out halmeter. If you just finished the previous example, then you can load siggen using the saved file. If not, we can load it just like we did before:

```
halrun
halcmd: loadrt siggen
halcmd: loadrt threads name1=test-thread period1=1000000
halcmd: addf siggen.0.update test-thread
halcmd: start
halcmd: setp siggen.0.amplitude 5
```

At this point we have the siggen component loaded and running. It’s time to start halmeter.

Starting Halmeter

```
halcmd: loadusr halmeter
```

La primera ventana que verá es la ventana *Seleccionar elemento a sondear*.

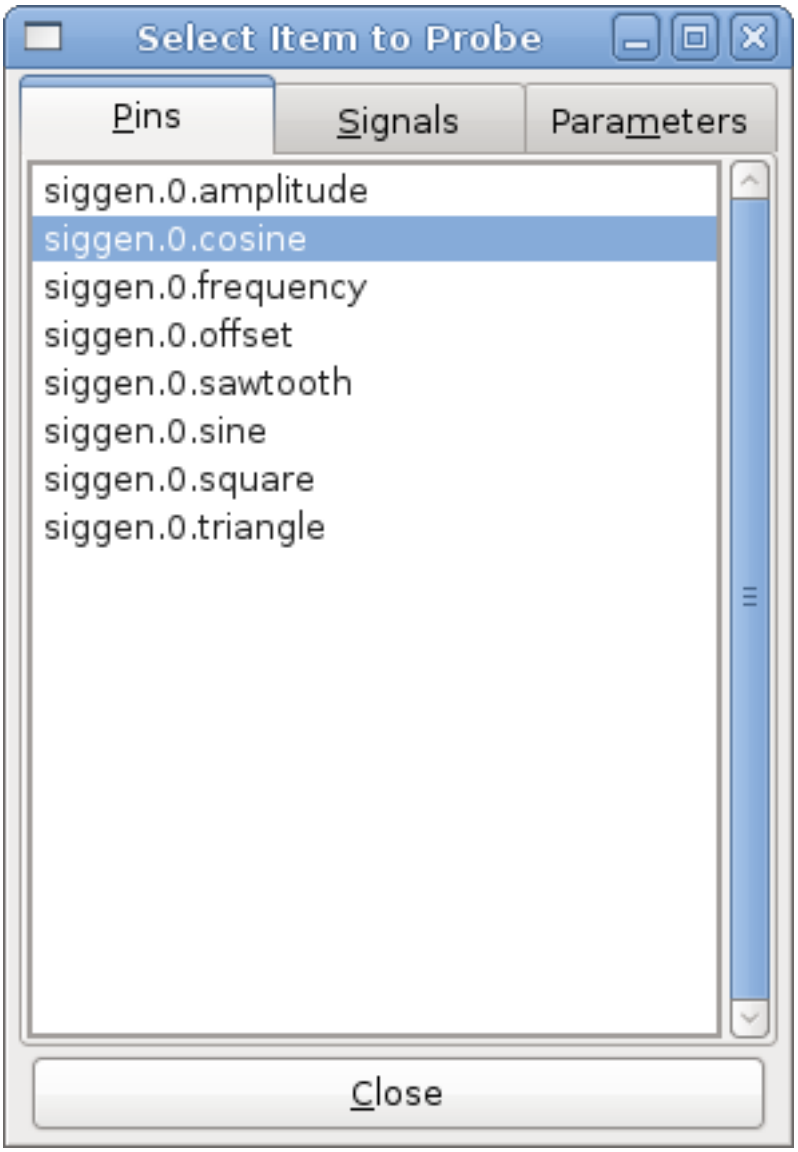


Figura 5.5: Halmeter Select Window

This dialog has three tabs. The first tab displays all of the HAL pins in the system. The second one displays all the signals, and the third displays all the parameters. We would like to look at the pin `siggen.0.cosine` first, so click on it then click the ”Close” button. The probe selection dialog will close, and the meter looks something like the following figure.



Figura 5.6: Halmeter Window

To change what the meter displays press the ”Select” button which brings back the ”Select Item to Probe” window.

You should see the value changing as siggen generates its cosine wave. Halmeter refreshes its display about 5 times per second.

To shut down halmeter, just click the exit button.

If you want to look at more than one pin, signal, or parameter at a time, you can just start more halmeters. The halmeter window was intentionally made very small so you could have a lot of them on the screen at once.

5.4.5. Stepgen Example

Up till now we have only loaded one HAL component. But the whole idea behind the HAL is to allow you to load and connect a number of simple components to make up a complex system. The next example will use two components.

Antes de que podamos comenzar a construir este nuevo ejemplo, queremos comenzar con una instancia limpia. Si se acaba de terminar uno de los ejemplos anteriores, necesitamos eliminar todos los componentes cargados y volver a cargar las bibliotecas RTAPI y HAL.

```
halcmd: exit
```

5.4.5.1. Installing the components

Now we are going to load the step pulse generator component. For a detailed description of this component refer to the stepgen section of the Integrator Manual. In this example we will use the *velocity* control type of StepGen. For now, we can skip the details, and just run the following commands.

In this example we will use the *velocity* control type from the `stepgen` component.

```
halrun
halcmd: loadrt stepgen step_type=0,0 ctrl_type=v,v
halcmd: loadrt siggen
halcmd: loadrt threads name1=fast period1=50000 name2=slow period2=1000000
```

The first command loads two step generators, both configured to generate stepping type 0. The second command loads our old friend siggen, and the third one creates two threads, a fast one with a period of 50 microseconds (µs) and a slow one with a period of 1 millisecond (ms).

---

nota
The `fp1=` parameter is deprecated and ignored. All threads now unconditionally support floating point.

As before, we can use `halcmd show` to take a look at the HAL. This time we have a lot more pins and parameters than before:

```
halcmd: show pin

Component Pins:
Owner   Type  Dir         Value  Name
     4  float IN              1  siggen.0.amplitude
     4  bit   OUT         FALSE  siggen.0.clock
     4  float OUT             0  siggen.0.cosine
     4  float IN              1  siggen.0.frequency
     4  float IN              0  siggen.0.offset
     4  float OUT             0  siggen.0.sawtooth
     4  float OUT             0  siggen.0.sine
     4  float OUT             0  siggen.0.square
     4  float OUT             0  siggen.0.triangle
     3  s32   OUT             0  stepgen.0.counts
     3  bit   OUT         FALSE  stepgen.0.dir
     3  bit   IN          FALSE  stepgen.0.enable
     3  float OUT             0  stepgen.0.position-fb
     3  bit   OUT         FALSE  stepgen.0.step
     3  float IN              0  stepgen.0.velocity-cmd
     3  s32   OUT             0  stepgen.1.counts
     3  bit   OUT         FALSE  stepgen.1.dir
     3  bit   IN          FALSE  stepgen.1.enable
     3  float OUT             0  stepgen.1.position-fb
     3  bit   OUT         FALSE  stepgen.1.step
     3  float IN              0  stepgen.1.velocity-cmd

halcmd: show param

Parameters:
Owner   Type  Dir         Value  Name
     4  s32   RO              0  siggen.0.update.time
     4  s32   RW              0  siggen.0.update.tmax
     3  u32   RW     0x00000001  stepgen.0.dirhold
     3  u32   RW     0x00000001  stepgen.0.dirsetup
     3  float RO              0  stepgen.0.frequency
     3  float RW              0  stepgen.0.maxaccel
     3  float RW              0  stepgen.0.maxvel
     3  float RW              1  stepgen.0.position-scale
     3  s32   RO              0  stepgen.0.rawcounts
     3  u32   RW     0x00000001  stepgen.0.steplen
     3  u32   RW     0x00000001  stepgen.0.stepspace
     3  u32   RW     0x00000001  stepgen.1.dirhold
     3  u32   RW     0x00000001  stepgen.1.dirsetup
     3  float RO              0  stepgen.1.frequency
     3  float RW              0  stepgen.1.maxaccel
     3  float RW              0  stepgen.1.maxvel
     3  float RW              1  stepgen.1.position-scale
     3  s32   RO              0  stepgen.1.rawcounts
     3  u32   RW     0x00000001  stepgen.1.steplen
     3  u32   RW     0x00000001  stepgen.1.stepspace
     3  s32   RO              0  stepgen.capture-position.time
     3  s32   RW              0  stepgen.capture-position.tmax
     3  s32   RO              0  stepgen.make-pulses.time
     3  s32   RW              0  stepgen.make-pulses.tmax
     3  s32   RO              0  stepgen.update-freq.time
     3  s32   RW              0  stepgen.update-freq.tmax
```

5.4.5.2. Connecting pins with signals

What we have is two step pulse generators, and a signal generator. Now it is time to create some HAL signals to connect the two components. We are going to pretend that the two step pulse generators are driving the X and Y axis of a machine. We want to move the table in circles. To do this, we will send a cosine signal to the X axis, and a sine signal to the Y axis. The siggen module creates the sine and cosine, but we need *wires* to connect the modules together. In the HAL, *wires* are called signals. We need to create two of them. We can call them anything we want, for this example they will be *X-vel* and *Y-vel*. The signal *X-vel* is intended to run from the cosine output of the signal generator to the velocity input of the first step pulse generator. The first step is to connect the signal to the signal generator output. To connect a signal to a pin we use the net command.

net command

```
halcmd: net X-vel <= siggen.0.cosine
```

To see the effect of the `net` command, we show the signals again.

```
halcmd: show sig

Signals:
Type          Value  Name     (linked to)
float             0  X-vel <== siggen.0.cosine
```

When a signal is connected to one or more pins, the show command lists the pins immediately following the signal name. The *arrow* shows the direction of data flow - in this case, data flows from pin `siggen.0.cosine` to signal `X-vel`. Now let's connect the `X-vel` to the velocity input of a step pulse generator.

```
halcmd: net X-vel => stepgen.0.velocity-cmd
```

We can also connect up the Y axis signal `Y-vel`. It is intended to run from the sine output of the signal generator to the input of the second step pulse generator. The following command accomplishes in one line what two `net` commands accomplished for `X-vel`.

```
halcmd: net Y-vel siggen.0.sine => stepgen.1.velocity-cmd
```

Now let's take a final look at the signals and the pins connected to them.

```
halcmd: show sig

Signals:
Type          Value  Name     (linked to)
float             0  X-vel <== siggen.0.cosine
                            ==> stepgen.0.velocity-cmd
float             0  Y-vel <== siggen.0.sine
                            ==> stepgen.1.velocity-cmd
```

The *show sig* command makes it clear exactly how data flows through the HAL. For example, the *X-vel* signal comes from pin `siggen.0.cosine`, and goes to pin `stepgen.0.velocity-cmd`.

5.4.5.3. Setting up realtime execution - threads and functions

Thinking about data flowing through "wires" makes pins and signals fairly easy to understand. Threads and functions are a little more difficult. Functions contain the computer instructions that actually get things done. Thread are the method used to make those instructions run when they are needed. First let's look at the functions available to us.

```
halcmd: show funct

Exported Functions:
Owner   CodeAddr  Arg       FP   Users  Name
 00004  f9992000  fc731278  YES      0   siggen.0.update
 00003  f998b20f  fc7310b8  YES      0   stepgen.capture-position
 00003  f998b000  fc7310b8  NO       0   stepgen.make-pulses
 00003  f998b307  fc7310b8  YES      0   stepgen.update-freq
```

En general, deberá consultar la documentación de cada componente para ver lo que hacen sus funciones. En este caso, la función `siggen.0.update` se usa para actualizar las salidas del generador de señales. Cada vez que se ejecuta, calcula los valores de las salidas seno, coseno, triángulo y cuadrado. Para sacar señales limpias, necesita ejecutarse a intervalos específicos.

The other three functions are related to the step pulse generators.

La primera, `stepgen.capture_position`, se usa para la realimentación de posición. Captura el valor de un contador interno que cuenta los pulsos de paso a medida que se generan. Suponiendo que no se pierden pasos, este contador indica la posición del motor.

The main function for the step pulse generator is `stepgen.make_pulses`. Every time *make_pulses* runs it decides if it is time to take a step, and if so sets the outputs accordingly. For smooth step pulses, it should run as frequently as possible. Because it needs to run so fast, *make_pulses* is highly optimized and performs only a few calculations.

The last function, `stepgen.update-freq`, is responsible for doing scaling and some other calculations that need to be performed only when the frequency command changes.

What this means for our example is that we want to run `siggen.0.update` at a moderate rate to calculate the sine and cosine values. Immediately after we run `siggen.0.update`, we want to run `stepgen.update_freq` to load the new values into the step pulse generator. Finally we need to run `stepgen.make_pulses` as fast as possible for smooth pulses. Because we don’t use position feedback, we don’t need to run `stepgen.capture_position` at all.

We run functions by adding them to threads. Each thread runs at a specific rate. Let’s see what threads we have available.

```
halcmd: show thread

Realtime Threads:
     Period  FP     Name               (     Time, Max-Time )
     996980  YES                  slow (        0,        0 )
      49849  YES                  fast (        0,        0 )
```

The two threads were created when we loaded `threads`. The first one, *slow*, runs every millisecond. We will use it for `siggen.0.update` and `stepgen.update_freq`. The second thread is *fast*, which runs every 50 microseconds (µs). We will use it for `stepgen.make_pulses`. To connect the functions to the proper thread, we use the `addf` command. We specify the function first, followed by the thread.

```
halcmd: addf siggen.0.update slow
halcmd: addf stepgen.update-freq slow
halcmd: addf stepgen.make-pulses fast
```

After we give these commands, we can run the `show thread` command again to see what happened.

```
halcmd: show thread

Realtime Threads:
     Period  FP     Name               (     Time, Max-Time )
     996980  YES                  slow (        0,        0 )
                  1 siggen.0.update
                  2 stepgen.update-freq
```

```
      49849  NO                   fast (        0,        0 )
                  1 stepgen.make-pulses
```

Now each thread is followed by the names of the functions, in the order in which the functions will run.

5.4.5.4. Setting parameters

Estamos casi listos para arrancar nuestro sistema HAL. Sin embargo, todavía tenemos que ajustar algunos parámetros. Por defecto, el componente siggen genera señales que oscilan desde +1 a -1. Para nuestro ejemplo, eso está bien, queremos que la velocidad de la mesa varíe de +1 a -1 pulgada por segundo. Sin embargo, la escala del generador de impulsos de pasos no esta del todo correcta. Por defecto, genera una frecuencia de salida de 1 paso por segundo para una entrada de 1.0. Es poco probable que un paso por segundo nos dé una pulgada por segundo de movimiento de la mesa. Supongamos que tenemos un husillo con 5 vueltas por pulgada de avance, conectado a un stepper de 200 pasos por revolucion, con microstepping 10x. Por tanto, se necesitan 2000 pasos para una revolución del tornillo, y 5 revoluciones para moverse una pulgada. Eso significa que la escala general es 10000 pasos por pulgada. Necesitamos multiplicar la entrada de velocidad al generador de impulsos de paso por 10000 para obtener la salida adecuada. Es exactamente para lo qué sirve el parámetro `stepgen.n.velocity-scale`. En este caso, tanto el eje X como el eje Y tienen la misma escala, por lo que establecemos los parámetros de escala para ambos en 10000.

```
halcmd: setp stepgen.0.position-scale 10000
halcmd: setp stepgen.1.position-scale 10000
halcmd: setp stepgen.0.enable 1
halcmd: setp stepgen.1.enable 1
```

This velocity scaling means that when the pin `stepgen.0.velocity-cmd` is 1.0, the step generator will generate 10000 pulses per second (10 kHz). With the motor and leadscrew described above, that will result in the axis moving at exactly 1.0 inches per second. This illustrates a key HAL concept - things like scaling are done at the lowest possible level, in this case in the step pulse generator. The internal signal `X-vel` is the velocity of the table in inches per second, and other components such as `siggen` don't know (or care) about the scaling at all. If we changed the leadscrew, or motor, we would change only the scaling parameter of the step pulse generator.

5.4.5.5. Run it!

We now have everything configured and are ready to start it up. Just like in the first example, we use the `start` command.

```
halcmd: start
```

Aunque parece no suceder nada, dentro de la computadora los generadores de pulsos de paso están generando impulsos, variando entre +10 kHz y -10 kHz de ida y vuelta cada segundo. Más adelante en este tutorial veremos cómo sacar esas señales internas para mover motores en la realidad, pero primero queremos ver los pulsos y comprobar lo qué está sucediendo.

5.4.6. Halscope

The previous example generates some very interesting signals. But much of what happens is far too fast to see with halmeter. To take a closer look at what is going on inside the HAL, we want an oscilloscope. Fortunately HAL has one, called halscope.

Halscope tiene dos partes - una parte en tiempo real que lee señales HAL, y una parte en tiempo no-real que proporciona la GUI y la pantalla. Sin embargo, usted no necesita preocuparse por esto, porque la porción en tiempo no-real cargará automáticame la parte en tiempo real cuando se necesite.

With LinuxCNC running in a terminal you can start halscope with the following command.

Starting Halscope

```
halcmd loadusr halscope
```

If LinuxCNC is not running or the autosave.halscope file does not match the pins available in the current running LinuxCNC the scope GUI window will open, immediately followed by a *Realtime function not linked* dialog that looks like the following figure. To change the sample rate left click on the samples box.

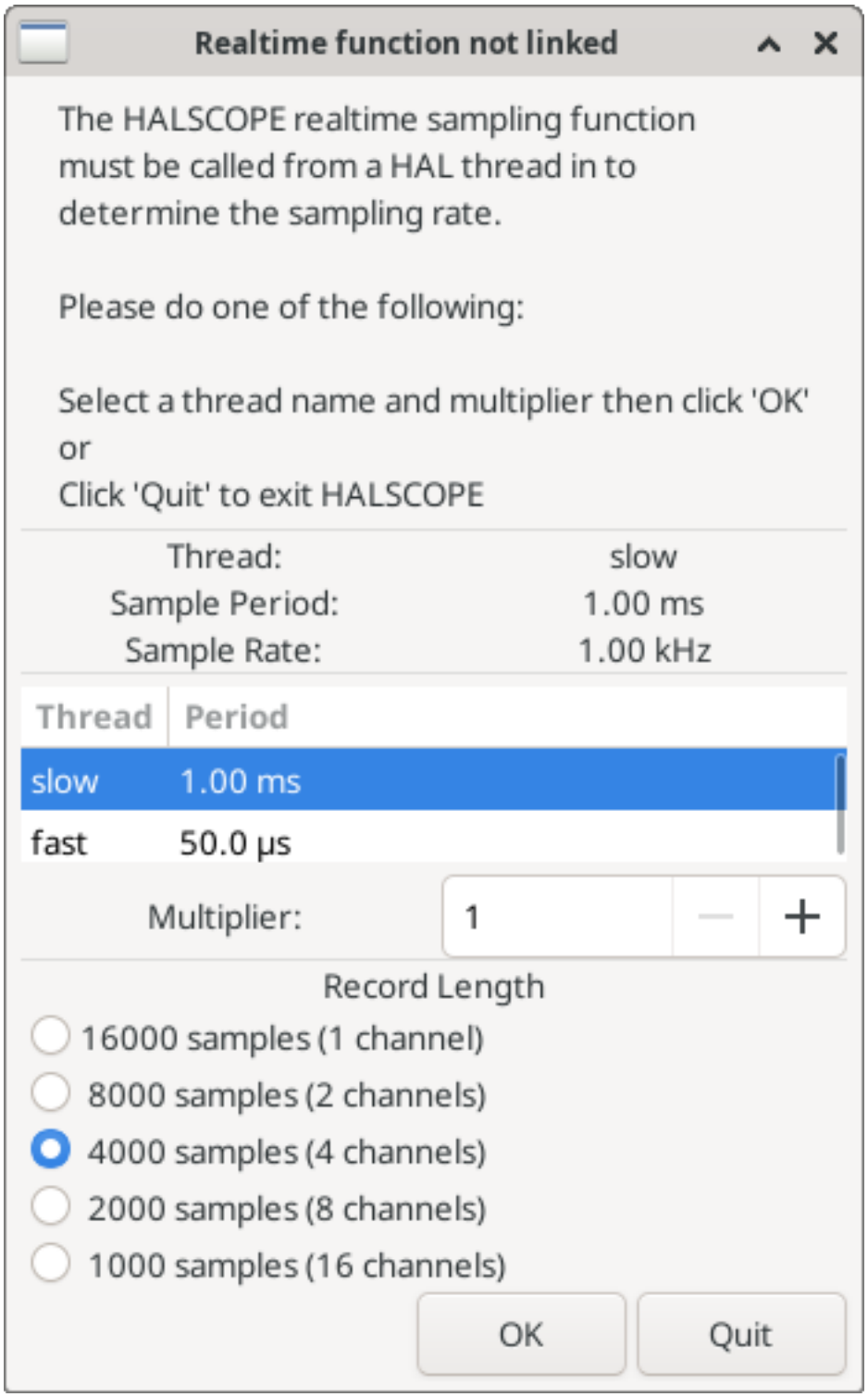


Figura 5.7: Realtime function not linked dialog

This dialog is where you set the sampling rate for the oscilloscope. For now we want to sample once per millisecond, so click on the 1.00 ms thread *slow* and leave the multiplier at 1. We will also leave the record length at 4000 samples, so that we can use up to four channels at one time. When you select a thread and then click *OK*, the dialog disappears, and the scope window looks something like the following figure.

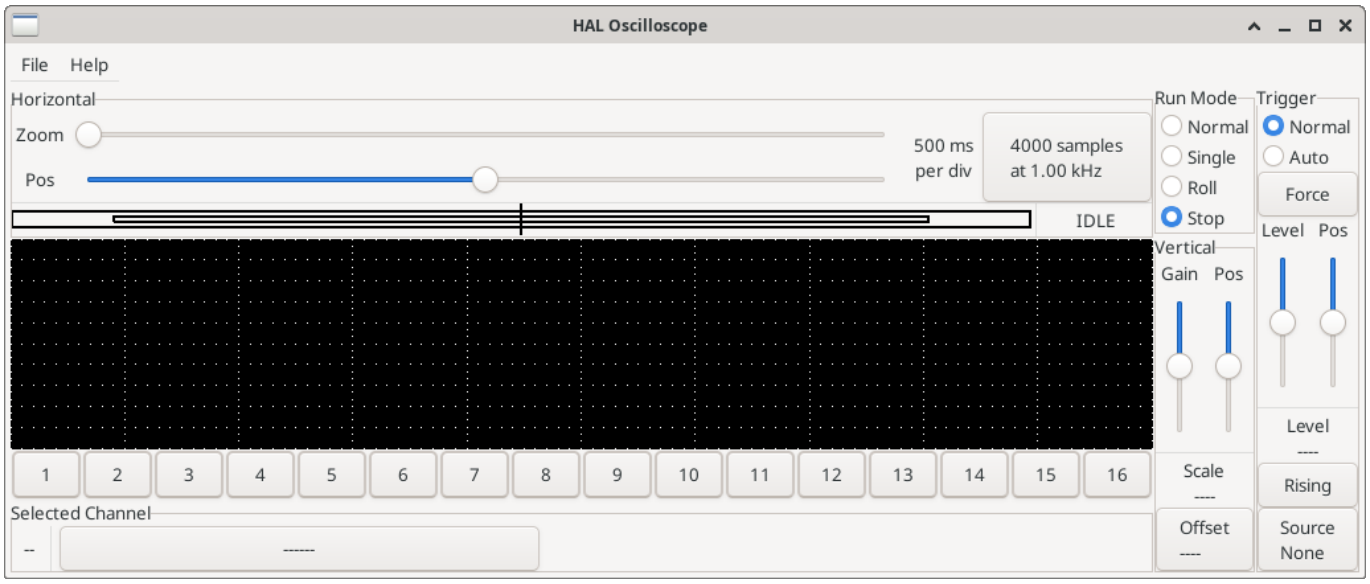


Figura 5.8: Initial scope window

5.4.6.1. Hooking up the scope probes

At this point, Halscope is ready to use. We have already selected a sample rate and record length, so the next step is to decide what to look at. This is equivalent to hooking *virtual scope probes* to the HAL. Halscope has 16 channels, but the number you can use at any one time depends on the record length - more channels means shorter records, since the memory available for the record is fixed at approximately 16,000 samples.

The channel buttons run across the bottom of the halscope screen. Click button *1*, and you will see the *Select Channel Source* dialog as shown in the following figure. This dialog is very similar to the one used by Halmeter. We would like to look at the signals we defined earlier, so we click on the *Signals* tab, and the dialog displays all of the signals in the HAL (only two for this example).

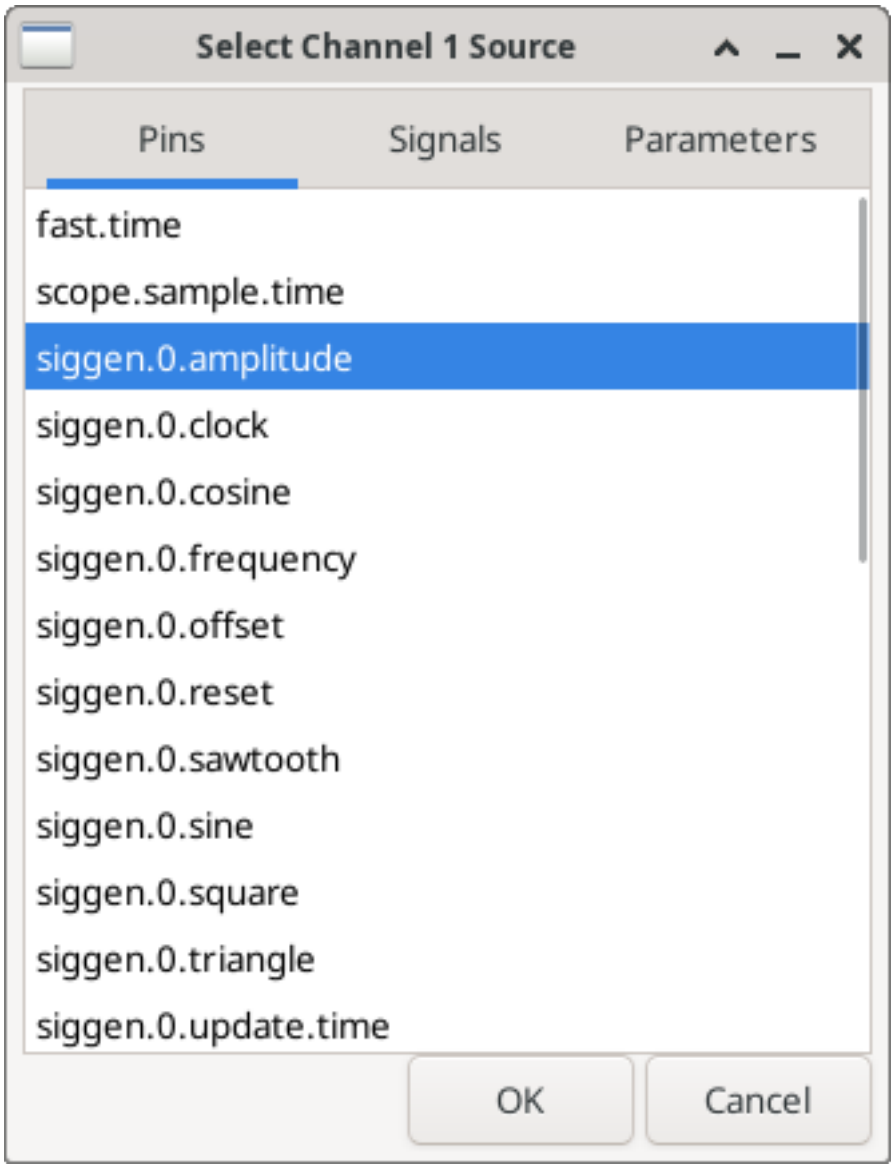


Figura 5.9: Select Channel Source

To choose a signal, just click on it. In this case, we want channel 1 to display the signal *X-vel*. Click on the Signals tab then click on *X-vel* and the dialog closes and the channel is now selected.

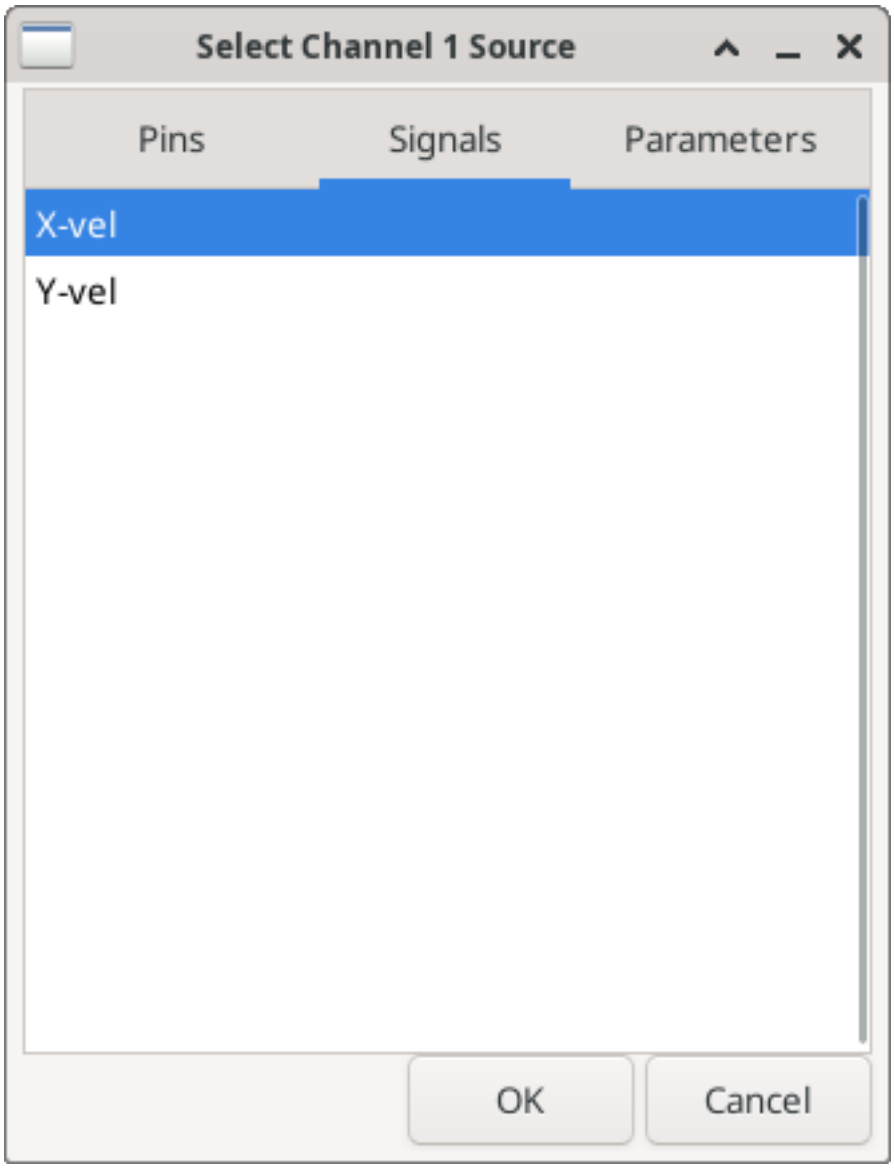


Figura 5.10: Select Signal

The channel 1 button is pressed in, and channel number 1 and the name *X-vel* appear below the row of buttons. That display always indicates the selected channel - you can have many channels on the screen, but the selected one is highlighted, and the various controls like vertical position and scale always work on the selected one.

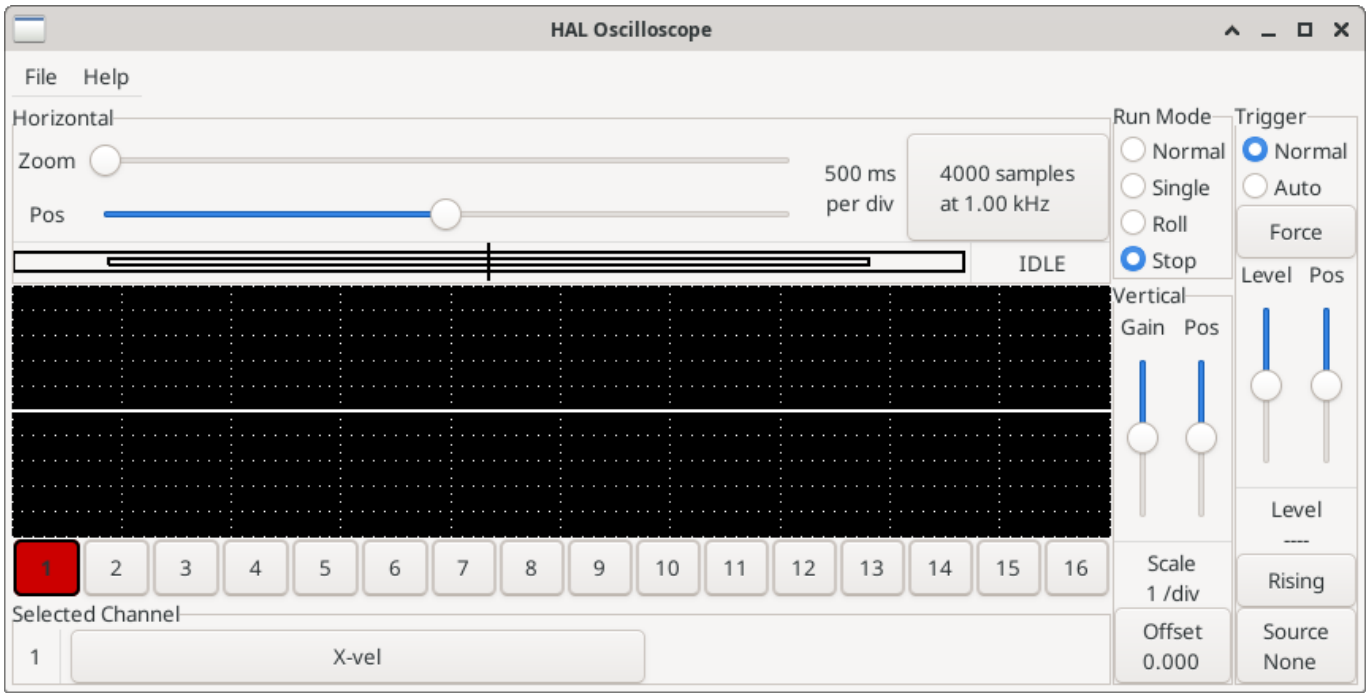


Figura 5.11: Halscope

To add a signal to channel 2, click the *2* button. When the dialog pops up, click the *Signals* tab, then click on *Y-vel*. We also want to look at the square and triangle wave outputs. There are no signals connected to those pins, so we use the *Pins* tab instead. For channel 3, select `siggen.0.triangle` and for channel 4, select `siggen.0.square`.

5.4.6.2. Capturing our first waveforms

Now that we have several probes hooked to the HAL, it's time to capture some waveforms. To start the scope, click the *Normal* button in the *Run Mode* section of the screen (upper right). Since we have a 4000 sample record length, and are acquiring 1000 samples per second, it will take halscope about 2 seconds to fill half of its buffer. During that time a progress bar just above the main screen will show the buffer filling. Once the buffer is half full, the scope waits for a trigger. Since we haven't configured one yet, it will wait forever. To manually trigger it, click the *Force* button in the *Trigger* section at the top right. You should see the remainder of the buffer fill, then the screen will display the captured waveforms. The result will look something like the following figure.

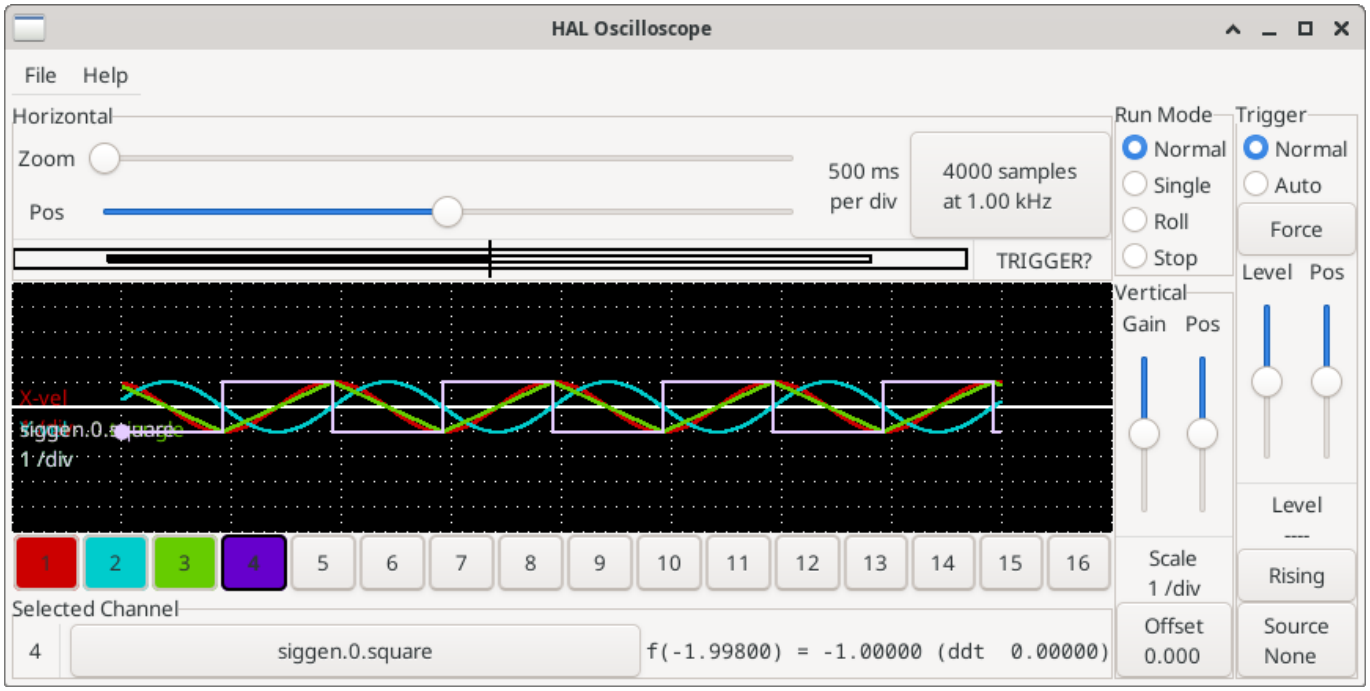


Figura 5.12: Captured Waveforms

The *Selected Channel* box at the bottom tells you that the purple trace is the currently selected one, channel 4, which is displaying the value of the pin `siggen.0.square`. Try clicking channel buttons 1 through 3 to highlight the other three traces.

5.4.6.3. Vertical Adjustments

Las ondas son bastante difíciles de distinguir ya que las cuatro están una sobre otra. Para solucionar esto, usamos los controles *Vertical* en el cuadro derecho de la pantalla. Estos controles actúan en el canal actualmente seleccionado. Al ajustar la ganancia, observe que cubre un amplio rango. A diferencia de un osciloscopio real, este puede mostrar señales que van desde muy pequeñas (pico-unidades) a muy grandes (tera-unidades). El control de posición mueve el trazo mostrado arriba y abajo sobre la altura de la pantalla solamente. Para ajustes más grandes se debe usar el botón de desplazamiento.

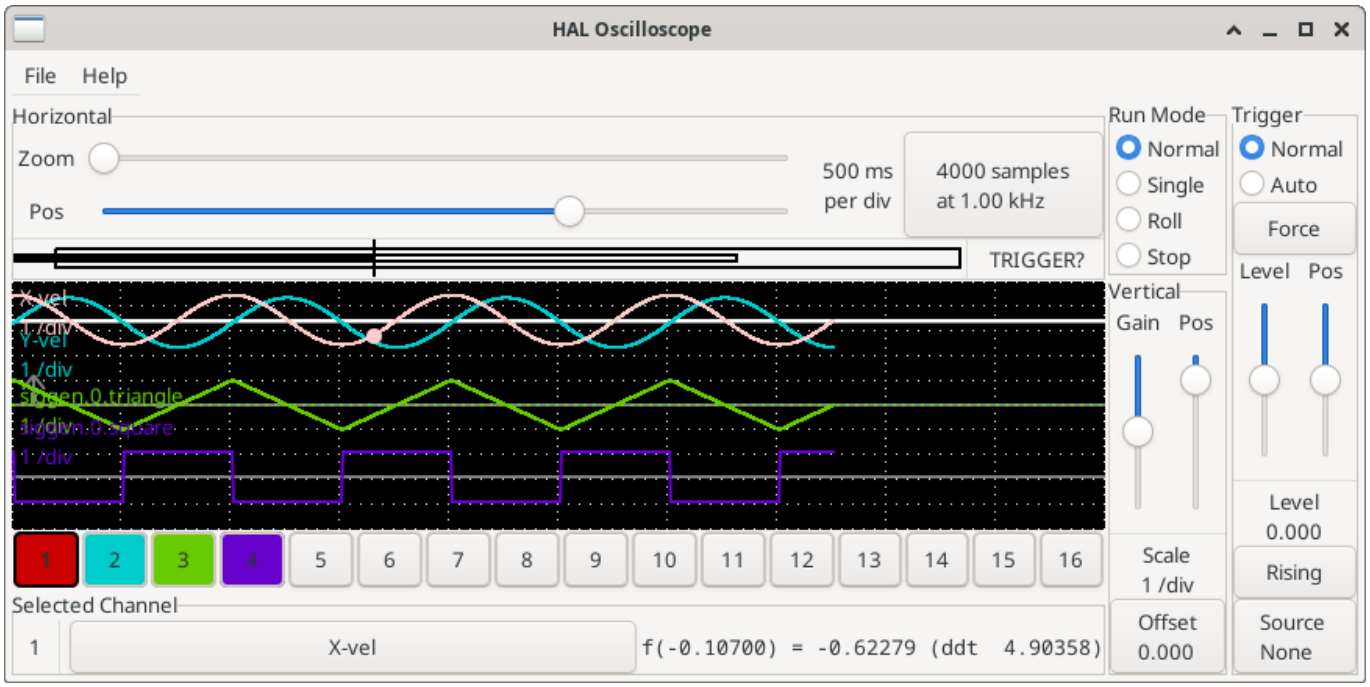


Figura 5.13: Vertical Adjustment

The large *Selected Channel* button at the bottom indicates that channel 1 is currently selected channel and that it matches the *X-vel* signal. Try clicking on the other channels to put their traces in evidence and to be able to move them with the *Pos* cursor.

5.4.6.4. Triggering

Using the *Force* button is a rather unsatisfying way to trigger the scope. To set up real triggering, click on the *Source* button at the bottom right. It will pop up the *Trigger Source* dialog, which is simply a list of all the probes that are currently connected. Select a probe to use for triggering by clicking on it. For this example we will use channel 3, the triangle wave as shown in the following figure.

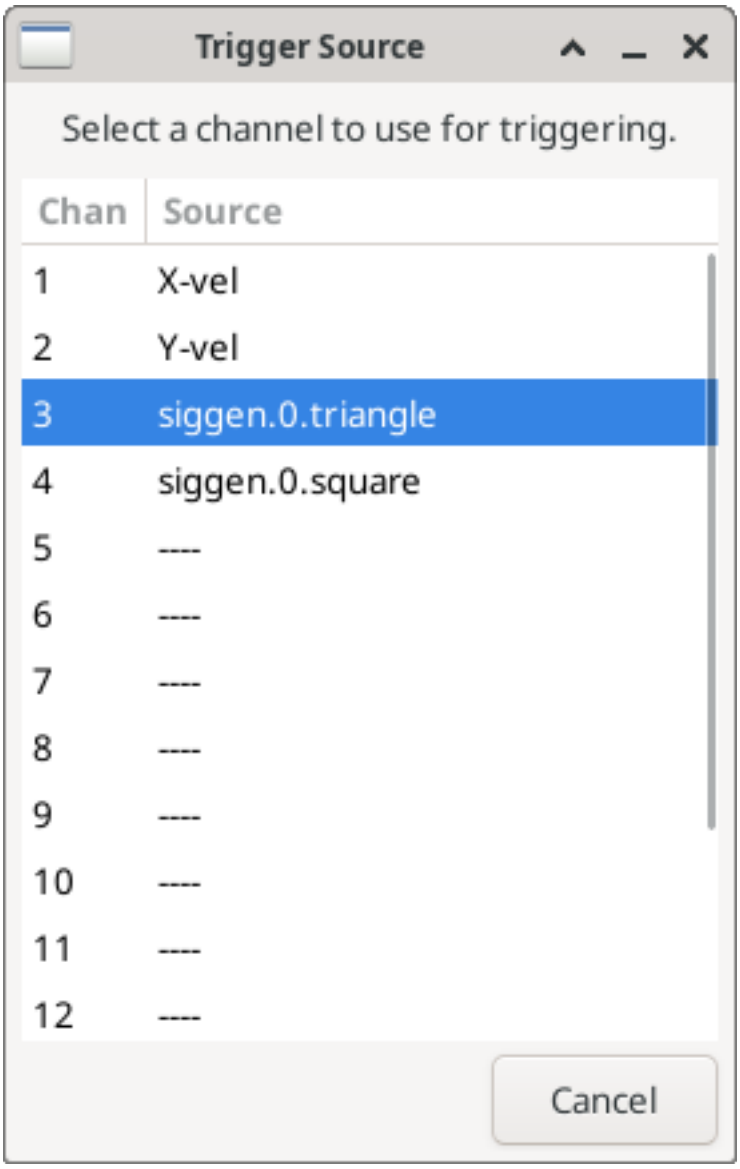


Figura 5.14: Trigger Source Dialog

After setting the trigger source, you can adjust the trigger level and trigger position using the sliders in the *Trigger* box along the right edge. The level can be adjusted from the top to the bottom of the screen, and is displayed below the sliders. The position is the location of the trigger point within the overall record. With the slider all the way down, the trigger point is at the end of the record, and halscope displays what happened before the trigger point. When the slider is all the way up, the trigger point is at the beginning of the record, displaying what happened after it was triggered. The trigger point is visible as a vertical line in the progress box above the screen. The trigger polarity can be changed by clicking the button just below the trigger level display. It will then become *descendant*. Note that changing the trigger position stops the scope once the position has been adjusted, you relaunch the scope by clicking on the *Normal* button of *Run mode* the group.

Now that we have adjusted the vertical controls and triggering, the scope display looks something like the following figure.

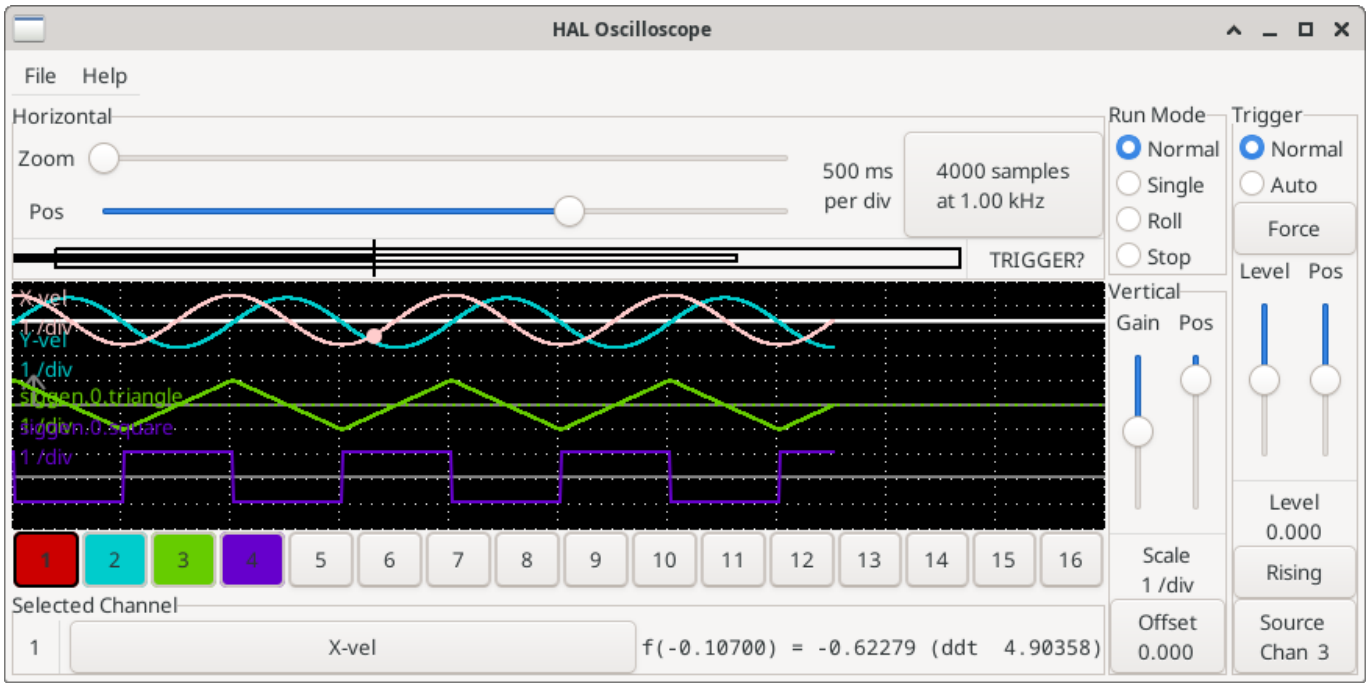


Figura 5.15: Waveforms with Triggering

5.4.6.5. Horizontal Adjustments

To look closely at part of a waveform, you can use the zoom slider at the top of the screen to expand the waveforms horizontally, and the position slider to determine which part of the zoomed waveform is visible. However, sometimes simply expanding the waveforms isn't enough and you need to increase the sampling rate. For example, we would like to look at the actual step pulses that are being generated in our example. Since the step pulses may be only 50 µs long, sampling at 1 kHz isn't fast enough. To change the sample rate, click on the button that displays the number of samples and sample rate to bring up the *Select Sample Rate* dialog figure. For this example, we will click on the 50 µs thread, *fast*, which gives us a sample rate of about 20 kHz. Now instead of displaying about 4 seconds worth of data, one record is 4000 samples at 20 kHz, or about 0.20 seconds.

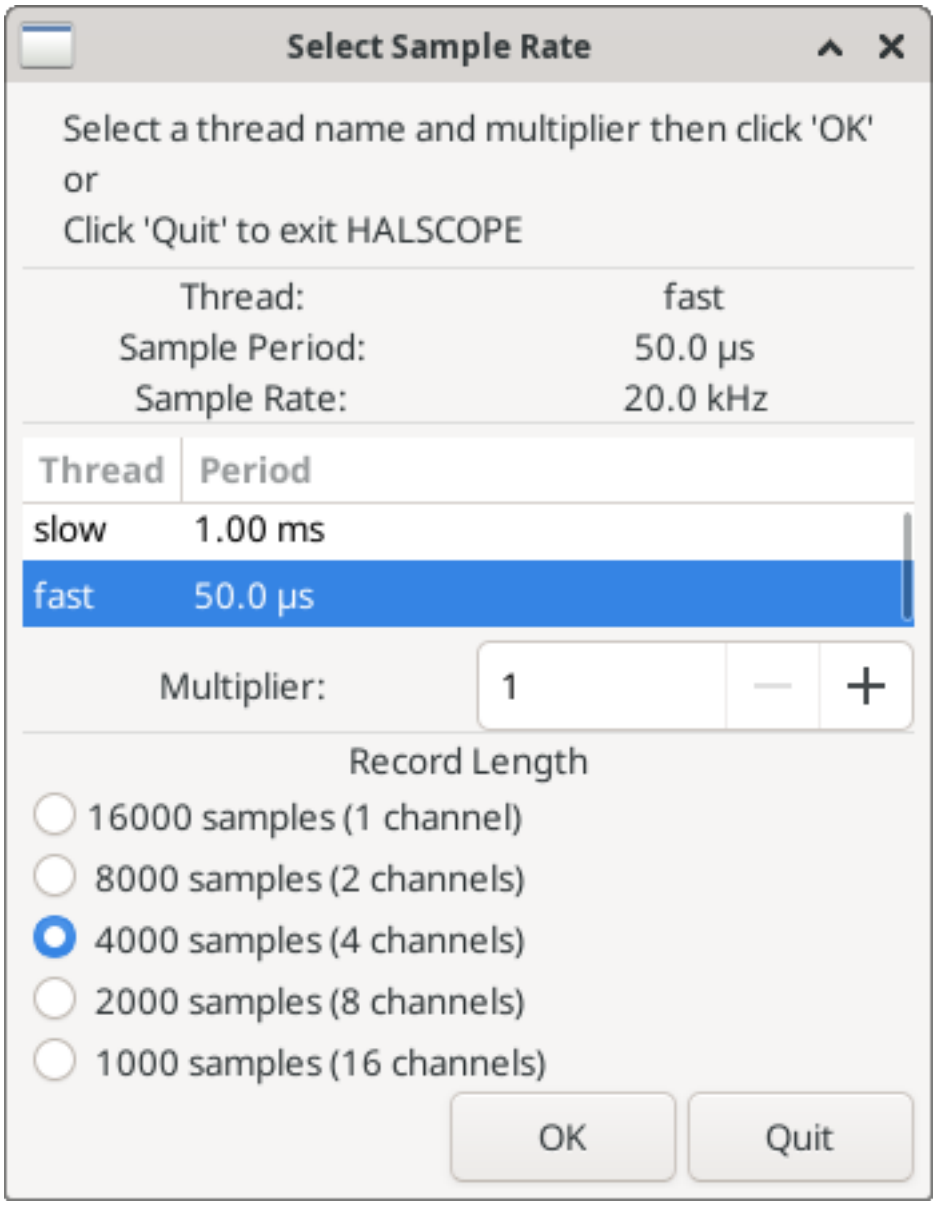


Figura 5.16: Sample Rate Dialog

5.4.6.6. More Channels

Now let's look at the step pulses. Halscope has 16 channels, but for this example we are using only 4 at a time. Before we select any more channels, we need to turn off a couple. Clicking on a selected channel button (black border) will turn the channel off. So click on the channel 2 button, then click again on this button and the channel will turn off. Then click twice on channel 3 and do the same for channel 4. Even though the channels are turned off, they still remember what they are connected to, and in fact we will continue to use channel 3 as the trigger source. To add new channels, select channel 5, and choose pin `stepgen.0.dir`, then channel 6, and select `stepgen.0.step`. Then click run mode *Normal* to start the scope, and adjust the horizontal zoom to 5 ms per division. You should see the step pulses slow down as the velocity command (channel 1) approaches zero, then the direction pin changes state and the step pulses speed up again. You might want toincrease the gain on channel 1 to about 20 milli per division to better see the change in the velocity command. The result should look like the following figure.

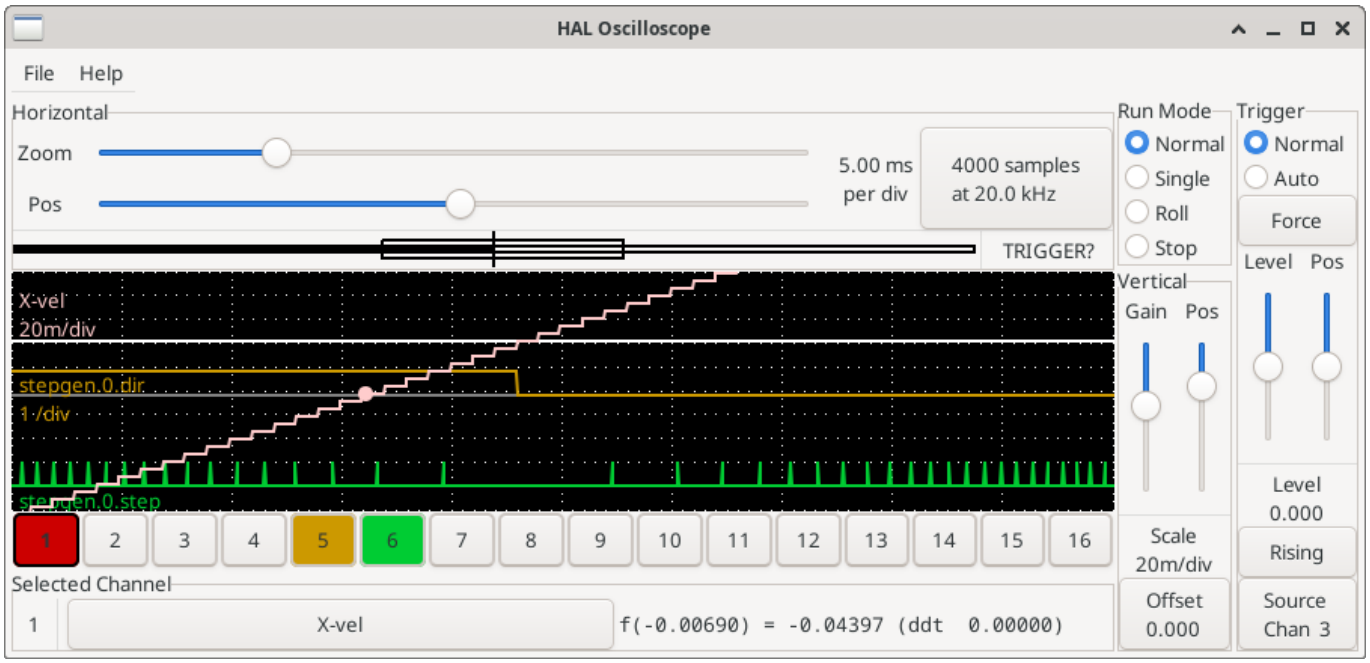


Figura 5.17: Step Pulses

5.4.6.7. More samples

Si desea grabar mas muestras a la vez, reinicie el tiempo real y cargue halscope con un argumento numérico que indique el número de muestras que quiere capturar.

```
halcmd loadusr halscope 80000
```

If the *scope_rt* component was not already loaded, halscope will load it and request 80000 total samples, so that when sampling 4 channels at a time there will be 20000 samples per channel. (If *scope_rt* was already loaded, the numeric argument to halscope will have no effect).

5.5. HAL Examples

All of these examples assume you are starting with a StepConf-based configuration and have two threads *base-thread* and *servo-thread*. The StepConf wizard will create an empty custom.hal and a custom_postgui.hal file. The custom.hal file will be loaded after the configuration HAL file and the custom_postgui.hal file is loaded after the GUI has been loaded.

5.5.1. Connecting Two Outputs

To connect two outputs to an input you can use the or2 component. The or2 works like this, if either input to or2 is on then the or2 output is on. If neither input to or2 is on the or2 output is off.

For example to have two PyVCP buttons both connected to one LED.

The .xml file to instruct PyVCP to prepare a GUI that features two buttons (named ”button-1” and ”button-2”) and an LED (named ”led-1”).

```
<pyvcp>
  <button>
    <halpin>”button-1”</halpin>
    <text>”Button 1”</text>
  </button>

  <button>
    <halpin>”button-2”</halpin>
    <text>”Button 2”</text>
  </button>

  <led>
    <halpin>”led-1”</halpin>
    <size>50</size>
    <on_color>”green”</on_color>
    <off_color>”red”</off_color>
  </led>
</pyvcp>
```

The postgui.hal file, read after the GUI is set up and ports ready to accept the logic described in HAL.

```
loadrt or2
addf or2.0 servo-thread
net button-1 or2.0.in0 <= pyvcp.button-1
net button-2 or2.0.in1 <= pyvcp.button-2
net led-1 pyvcp.led-1 <= or2.0.out
```

When you run this example in an axis simulator created with the StepConf Wizard, you can open a terminal and see the pins created with *loadrt or2* by typing in `halcmd show pin or2` in the terminal.

Running halcmd on the UNIX command line to show the pins crafted with module or2.

```
$ halcmd show pin or2
Component Pins:
Owner   Type  Dir         Value  Name
    22  bit   IN          FALSE  or2.0.in0 <== button-1
    22  bit   IN          FALSE  or2.0.in1 <== button-2
    22  bit   OUT         FALSE  or2.0.out ==> led-1
```

You can see from the HAL command `show pin or2` that the `button-1` pin is connected to the `or2.0.in0` pin. From the direction arrow you can see that the button is and output and the `or2.0.in0` is an input. The output from or2 goes to the input of the LED.

5.5.2. Manual Toolchange

En este ejemplo, se supone que está usando su propia configuración y desea agregar la ventana HAL Manual Toolchange. El cambio manual de herramientas HAL es principalmente útil si tiene herramientas preestablecidas y almacena los offsets en la tabla de herramientas. Si necesita hacer touch off para cada cambio de herramienta, entonces es mejor dividir su código G. Para usar la ventana HAL Manual Toolchange básicamente tiene que

1. load the hal_manualtoolchange component,
2. then send the iocontrol *tool change* to the hal_manualtoolchange *change* and
3. send the hal_manualtoolchange *changed* back to the iocontrol *tool changed*.

A pin is provided for an external input to indicate the tool change is complete.

This is an example of manual toolchange *with* the HAL Manual Toolchange component:

```
loadusr -W hal_manualtoolchange
net tool-change iocontrol.0.tool-change => hal_manualtoolchange.change
net tool-changed iocontrol.0.tool-changed <= hal_manualtoolchange.changed
net external-tool-changed hal_manualtoolchange.change_button <= parport.0.pin-12-in
net tool-number iocontrol.0.tool-prep-number => hal_manualtoolchange.number
net tool-prepare-loopback iocontrol.0.tool-prepare => iocontrol.0.tool-prepared
```

This is an example of manual toolchange *without* the HAL Manual Toolchange component:

```
net tool-number <= iocontrol.0.tool-prep-number
net tool-change-loopback iocontrol.0.tool-change => iocontrol.0.tool-changed
net tool-prepare-loopback iocontrol.0.tool-prepare => iocontrol.0.tool-prepared
```

5.5.3. Compute Velocity

This example uses *ddt, mult2* and *abs* to compute the velocity of a single axis. For more information on the real time components see the man pages or the HAL Components List (Sección 5.1.5).

The first thing is to check your configuration to make sure you are not using any of the real time components all ready. You can do this by opening up the HAL Configuration window and look for the components in the pin section. If you are then find the HAL file that they are being loaded in and increase the counts and adjust the instance to the correct value. Add the following to your custom.hal file.

Load the realtime components.

```
loadrt ddt count=1
loadrt mult2 count=1
loadrt abs count=1
```

Add the functions to a thread so it will get updated.

```
addf ddt.0 servo-thread
addf mult2.0 servo-thread
addf abs.0 servo-thread
```

Make the connections.

```
setp mult2.in1 60
net xpos-cmd ddt.0.in
net X-IPS mult2.0.in0 <= ddt.0.out
net X-ABS abs.0.in <= mult2.0.out
net X-IPM abs.0.out
```

In this last section we are setting the `mult2.0.in1` to 60 to convert the inch per second to inch per minute (IPM) that we get from the `ddt.0.out`.

The `xpos-cmd` sends the commanded position to the `ddt.0.in`. The ddt computes the derivative of the change of the input.

The `ddt2.0.out` is multiplied by 60 to give IPM.

The `mult2.0.out` is sent to the `abs` to get the absolute value.

The following figure shows the result when the X axis is moving at 15 IPM in the minus direction. Notice that we can get the absolute value from either the `abs.0.out` pin or the `X-IPM` signal.

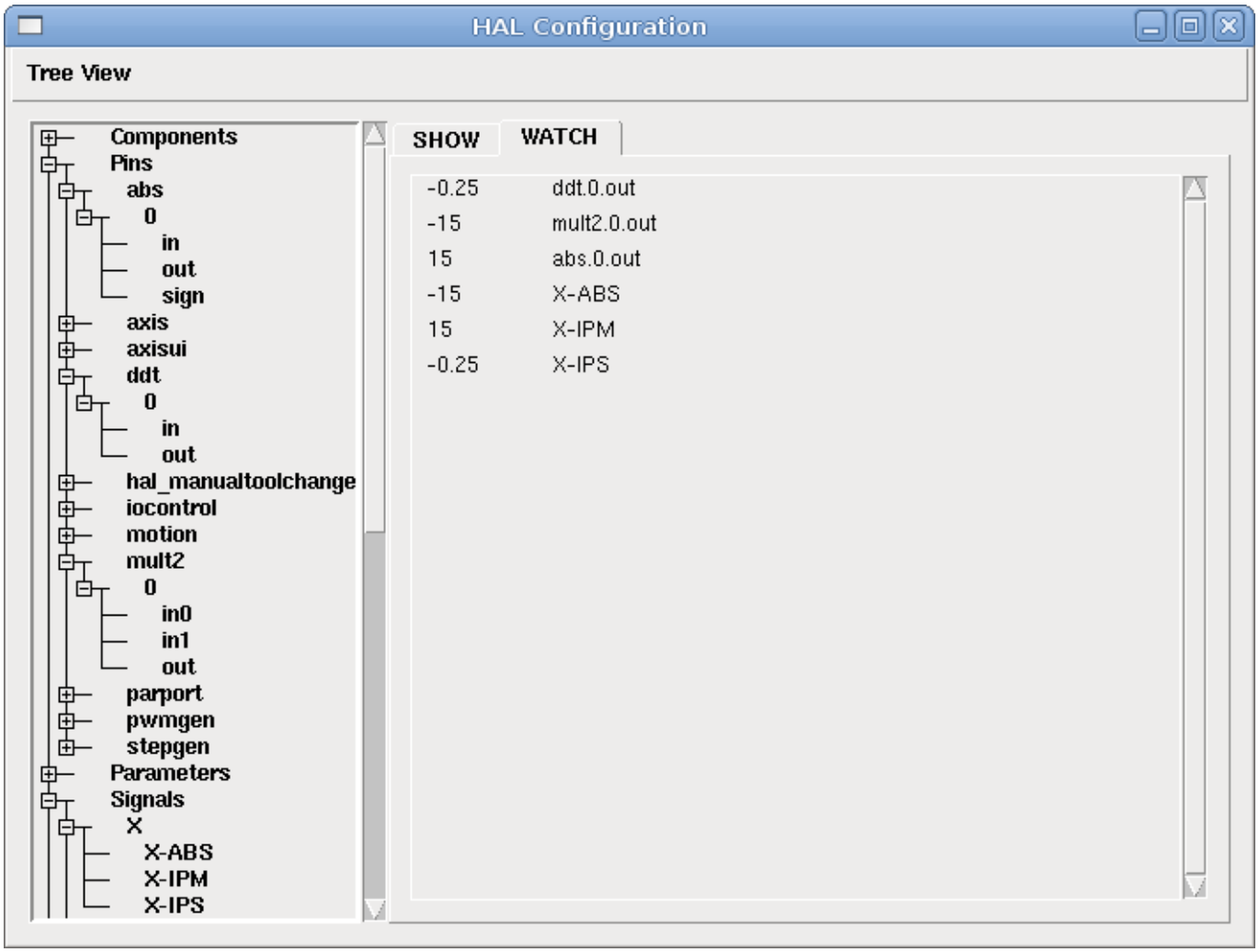


Figura 5.18: HAL: Velocity Example

5.5.4. Soft Start Details

This example shows how the HAL components *lowpass*, *limit2* or *limit3* can be used to limit how fast a signal changes.

In this example we have a servo motor driving a lathe spindle. If we just used the commanded spindle speeds on the servo it will try to go from present speed to commanded speed as fast as it can. This could cause a problem or damage the drive. To slow the rate of change we can send the `spindle.N.speed-out` through a limiter before the PID, so that the PID command value changes to new settings more slowly.

Three built-in components that limit a signal are:

- *limit2* limits the range and first derivative of a signal.
- *limit3* limits the range, first and second derivatives of a signal.
- *lowpass* uses an exponentially-weighted moving average to track an input signal.

To find more information on these HAL components check the man pages.

Place the following in a text file called softstart.hal. If you’re not familiar with Linux place the file in your home directory.

```
loadrt threads period1=1000000 name1=thread
loadrt siggen
loadrt lowpass
loadrt limit2
loadrt limit3
net square siggen.0.square => lowpass.0.in limit2.0.in limit3.0.in
net lowpass <= lowpass.0.out
net limit2 <= limit2.0.out
net limit3 <= limit3.0.out
setp siggen.0.frequency .1
setp lowpass.0.gain .01
setp limit2.0.maxv 2
setp limit3.0.maxv 2
setp limit3.0.maxa 10
addf siggen.0.update thread
addf lowpass.0 thread
addf limit2.0 thread
addf limit3.0 thread
start
loadusr halscope
```

Open a terminal window and run the file with the following command.

```
halrun -I softstart.hal
```

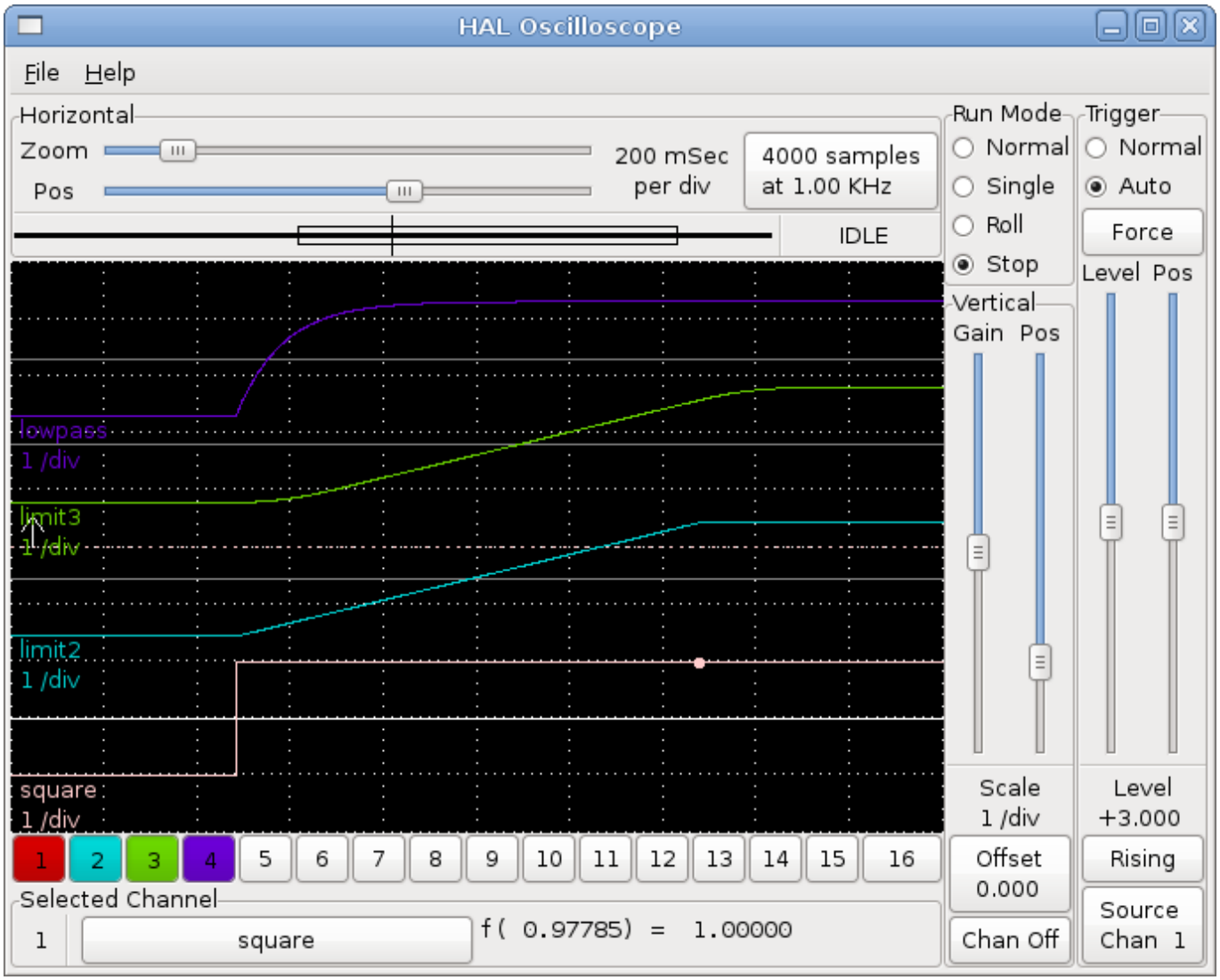
Cuando el osciloscopio HAL se inicie por primera vez haga clic en *Aceptar* para aceptar el hilo predeterminado.

Next you have to add the signals to the channels. Click on channel 1 then select *square* from the Signals tab. Repeat for channels 2-4 and add lowpass, limit2, and limit3.

Next to set up a trigger signal click on the Source None button and select square. The button will change to Source Chan 1.

Next click on Single in the Run Mode radio buttons box. This will start a run and when it finishes you will see your traces.

To separate the signals so you can see them better click on a channel then use the Pos slider in the Vertical box to set the positions.



To see the effect of changing the set point values of any of the components you can change them in the terminal window. To see what different gain settings do for lowpass just type the following in the terminal window and try different settings.

```
setp lowpass.0.gain *.01
```

After changing a setting run the oscilloscope again to see the change.

When you're finished type *exit* in the terminal window to shut down halrun and close the halscope. Don't close the terminal window with halrun running as it might leave some things in memory that could prevent LinuxCNC from loading.

For more information on Halscope see the HAL manual and the tutorial.

5.5.5. Stand Alone HAL

In some cases you might want to run a GladeVCP screen with just HAL. For example say you had a stepper driven device that all you need is to run a stepper motor. A simple *Start/Stop* interface is all you need for your application so no need to load up and configure a full blown CNC application.

In the following example we have created a simple GladeVCP panel with one stepper.

Basic Syntax

```
# load the winder.glade GUI and name it winder
loadusr -Wn winder gladevcp -c winder -u handler.py winder.glade

# load realtime components
loadrt threads name1=fast period1=50000 fp1=0 name2=slow period2=1000000
loadrt stepgen step_type=0 ctrl_type=v
loadrt hal_parport cfg="0x378 out"

# add functions to threads
addf stepgen.make-pulses fast
addf stepgen.update-freq slow
addf stepgen.capture-position slow
addf parport.0.read fast
addf parport.0.write fast

# make HAL connections
net winder-step parport.0.pin-02-out <= stepgen.0.step
net winder-dir parport.0.pin-03-out <= stepgen.0.dir
net run-stepgen stepgen.0.enable <= winder.start_button

# start the threads
start

# comment out the following lines while testing and use the interactive
# option halrun -I -f start.hal to be able to show pins etc.

# wait until the GladeVCP GUI named winder terminates
waitusr winder

# stop HAL threads
stop

# unload HAL all components before exiting
unloadrt all
```

5.6. Componentes centrales

Ver también la página de manual *motion(9)*.

5.6.1. Motion

Estos pines y parámetros son creados por el módulo en tiempo real *motmod*.

Este módulo proporciona una interfaz HAL para el planificador de movimiento de LinuxCNC.

Básicamente *motmod* toma una lista de puntos de camino y genera un flujo bien balanceado y restringido de posiciones de articulaciones para alimentar a los controladores de motor.

Opcionalmente, el número de E/S digitales se establece con *num_dio*. El número de E/S analógicas se establece con *num_aio*. El valor predeterminado es 4 en cada uno. El número de husillos se establece con *num_spindles*, el predeterminado es 1.

Los nombres de pines y parámetros, que comienzan con *axis*.L y *joint*.N son leídos y actualizados por la función motion-controller.

Motion se carga con el comando *motmod*. Antes de motion, debe ser cargado un modulo de cinematica.

```
loadrt motmod base_period_nsec=['period'] servo_period_nsec=['period']
              traj_period_nsec=['period'] num_joints=['0-9']
              num_dio=['1-64'] num_aio=['1-16'] unlock_joints_mask=['0xNN']
              num_spindles=['1-8']
```

- *base_period_nsec = 50000* - es el período de la tarea *Base* en nanosegundos. Este es el hilo más rápido en la máquina.

---

nota
En sistemas basados en servo, generalmente no hay razón para que *base_period_nsec* sea mas rapida que *servo_period_nsec*. En máquinas con generación por pasos de software, *base_period_nsec* determina la cantidad máxima de pasos por segundo. En ausencia de necesidad de pasos largos y requisitos de espacio, la tasa de pasos máxima absoluta es de un paso por *base_period_nsec*. Por lo tanto, para *base_period_nsec = 50000* (0,05 m´s) se tendria una tasa de paso máxima absoluta de (1000/0,05) 20,000 pasos por segundo. 50,000 ns (50 µs) es un valor bastante conservador. El menor valor utilizable está relacionado con el resultado de la prueba de latencia, el ttiemp de paso necesario, y la velocidad del procesador. Elegir un *base_period_nsec* que sea demasiado rapido puede conducir a un mensaje de "retraso en tiempo real inesperado", bloqueos o reinicios espontáneos.

- *servo_period_nsec = 1000000* - Este es el período de la tarea *Servo* en nanosegundos. Este valor se redondeará a un múltiplo entero de *base_period_nsec*. Este período se usa incluso en sistemas basados en motores paso a paso.

  Esta es la velocidad a la que se calculan las nuevas posiciones de los motores, se verifica el error de seguimiento, se actualizan los valores de salida PID, y así sucesivamente. La mayoría de los sistemas no necesitarán cambiar este valor. Es la tasa de actualización del planificador de movimiento de bajo nivel.

- *traj_period_nsec = 100000* - Este es el periodo de la tarea *Planificador de trayectoria* en nanosegundos. Este valor se redondeará a un número entero múltiplo de *servo_period_nsec*. Excepto para máquinas con cinemáticas inusuales (p. ej., hexápodos) no hay razón para que este valor sea más alto que *servo_period_nsec*.

5.6.1.1. Opciones

Si la cantidad de E/S digital necesaria es mayor que el valor predeterminado de 4, puede agregar hasta 64 E/S digitales usando la opción num_dio al cargar *motmod*.

Si la cantidad de E/S analógica necesaria es mayor que el valor predeterminado de 4, puede agregar hasta 16 E/S analógicas usando la opción num_aio al cargar *motmod*.

El parámetro unlock_joints_mask se usa para crear pines para una articulación utilizada como indexador con bloqueo (típicamente rotativas). Los bits de máscara seleccionan la(s) articulación(es). El bit menos significativo de la máscara selecciona la articulación 0. Ejemplo:

```
unlock_joints_mask=0x38 selecciona las articulaciones 3, 4 y 5
```

5.6.1.2. Pines

Estos pines, parámetros y funciones son creados por el módulo de tiempo real *motmod*.

- *motion.adaptive-feed* - (float, in) Cuando la alimentación adaptativa está habilitada con *M52 P1*, la velocidad ordenada se multiplica por este valor. Este efecto es multiplicativo con el valor de nivel NML de porcentaje de alimentación y *motion.feed-hold*. A partir de la versión 2.9 de LinuxCNC, es posible utilizar un valor de alimentación adaptativo negativo para ejecutar la ruta del código G en reversa.

- *motion.analog-in-00* - (float, in) Estos pines (00, 01, 02, 03, o más si están configurados) son controlados por M66.
- *motion.analog-out-00* - (float, out) Estos pines (00, 01, 02, 03, o más si están configurados) son controlados por M67 o M68.
- *motion.coord-error* - (bit, out) TRUE cuando el movimiento ha encontrado un error, como exceder un límite suave
- *motion.coord-mode* - (bit, out) TRUE cuando el movimiento está en *modo coordinado*, en oposición al *modo teleop*
- *motion.current-vel* - (float, out) La velocidad actual de la herramienta en unidades de usuario por segundo.
- *motion.digital-in-00* - (bit, in) Estos pines (00, 01, 02, 03 o más si están configurados) son controlados por M62-65.
- *motion.digital-out-00* - (bit, out) Estos pines (00, 01, 02, 03 o más si están configurados) son controlados por *M62-65*.
- *motion.distance-to-go* - (float, out) La distancia restante en el movimiento actual.
- *motion.enable* - (bit, in) Si este bit se hace FALSE, el movimiento se detiene; la máquina queda colocada en el estado *máquina apagada* y se muestra un mensaje para el operador. En movimiento normal, este bit es TRUE.
- *motion.feed-hold* - (bit, in) Cuando Feed Stop Control está habilitado con *M53 P1*, y este bit es TRUE, la velocidad de avance se establece en 0.
- *motion.feed-inhibit* - (bit, in) Cuando este bit es TRUE, la velocidad de avance se establece en 0. Esta acción se retrasará durante los movimientos de sincronización del husillo hasta el final del movimiento.
- *motion.in-position* - (bit, out) TRUE si la máquina está en posición.
- *motion.motion-enabled* - (bit, out) TRUE cuando la máquina está en estado *máquina encendida*.
- *motion.motion-type* - (s32, out) Valores definidos en src/emc/nml_intf/motion_types.h
  - 0: Idle (sin movimiento)
  - 1: Atravesar
  - 2: Avance lineal
  - 3: Avance de arco
  - 4: Cambio de herramienta
  - 5: Sondeo
  - 6: Indexación de eje rotativo
- *motion.on-soft-limit* - (bit, out) TRUE cuando la máquina está en un límite de software.
- *motion.probe-input* - (bit, in) *G38.n* usa el valor de este pin para determinar cuándo la sonda ha hecho contacto. TRUE para contacto de sonda cerrado (ha tocado), FALSO para contacto de sonda abierto.
- *motion.program-line* - (s32, out) La línea de programa actual durante la ejecución. Vale cero si no corre un programa o entre líneas mientras se avanza línea a línea.
- *motion.requested-vel* - (float, out) La velocidad solicitada actual en unidades de usuario por segundo. Este valor es la configuración de la palabra F del archivo de código G, posiblemente reducido para acomodar la velocidad y aceleración a los limites de la máquina. El valor en este pin no refleja el ajuste manual de la alimentación o cualquier otro ajuste.

- *motion.teleop-mode* - (bit, out) TRUE cuando el movimiento está en *modo teleop*, en oposición a *modo coordinado*
- *motion.tooloffset.x ... motion.tooloffset.w* - (float, out, uno por eje) muestra el offset de la herramienta en efecto; podría provenir de la tabla de herramientas (*G43* activo), o podría venir de código G (*G43.1* activo)
- *motion.on-soft-limit* - (bit, out) TRUE cuando la máquina está en un límite de software.
- *motion.probe-input* - (bit, in) *G38.n* usa el valor de este pin para determinar cuándo la sonda ha hecho contacto. TRUE para contacto de sonda cerrado (ha tocado), FALSO para contacto de sonda abierto.
- *motion.program-line* - (s32, out) La línea de programa actual durante la ejecución. Vale cero si no corre un programa o entre líneas mientras se avanza línea a línea.
- *motion.requested-vel* - (float, out) La velocidad solicitada actual en unidades de usuario por segundo. Este valor es la configuración de la palabra F del archivo de código G, posiblemente reducido para acomodar la velocidad y aceleración a los limites de la máquina. El valor en este pin no refleja el ajuste manual de la alimentación o cualquier otro ajuste.
- *motion.teleop-mode* - (bit, out) TRUE cuando el movimiento está en *modo teleop*, en oposición a *modo coordinado*
- *motion.tooloffset.x ... motion.tooloffset.w* - (float, out, uno por eje) muestra el offset de la herramienta en efecto; podría provenir de la tabla de herramientas (*G43* activo), o podría venir de código G (*G43.1* activo)

5.6.1.3. Parámetros

Muchos de estos parámetros sirven como ayudas para la depuración, y están sujetos a cambio o eliminación en cualquier momento.

- *motion-command-handler.time* - (s32, RO)
- *motion-command-handler.tmax* - (s32, RW)
- *motion-controller.time* - (s32, RO)
- *motion-controller.tmax* - (s32, RW)
- *motion.debug-bit-0* - (bit, RO) Se usa con fines de depuración.
- *motion.debug-bit-1* - (bit, RO) Se usa con fines de depuración.
- *motion.debug-float-0* - (float, RO) Se usa con fines de depuración.
- *motion.debug-float-1* - (float, RO) Se usa con fines de depuración.
- *motion.debug-float-2* - (float, RO) Se usa con fines de depuración.
- *motion.debug-float-3* - (float, RO) Se usa con fines de depuración.
- *motion.debug-s32-0* - (s32, RO) Se usa con fines de depuración.
- *motion.debug-s32-1* - (s32, RO) Se usa con fines de depuración.
- *motion.servo.last-period* - (u32, RO) El número de ciclos de CPU entre las invocaciones del hilo servo. Normalmente este número, dividido por la velocidad de la CPU, da el tiempo en segundos, y se puede usar para determinar si el movimiento en tiempo real del controlador cumple con sus restricciones de tiempo
- *motion.servo.last-period-ns* - (flotar, RO)

5.6.1.4. Funciones

En general, estas funciones se agregan al hilo servo en el orden mostrado.

- *motion-command-handler* - Recibe y procesa comandos de movimiento
- *motion-controller* - Ejecuta el controlador de movimiento LinuxCNC

5.6.2. Husillo

LinuxCNC puede controlar hasta ocho husillos. El movimiento producirá los pines siguientes: El *N* (entero entre 0 y 7) sustituye el número de husillo.

5.6.2.1. Pines

* *spindle*.N.*at-speed* - (bit, in) El movimiento se detendrá hasta que este pin sea TRUE, bajo las siguientes condiciones: **antes del primer movimiento de alimentación, después de cada arranque de husillo o cambio de velocidad;** antes del inicio de cada cadena de movimientos sincronizados con el husillo; ** si está en modo CSS, en cada transición de velocidad rapida a velocidad de alimentación. Esta entrada se puede usar para asegurar que el husillo esté a su velocidad antes de comenzar un corte, o que un husillo de torno en modo CSS se ha ralentizado después de un pase de refrentado grande a pequeño antes de comenzar el próximo pase en el diámetro grande. Muchos VFD tienen una salida *a velocidad*. De lo contrario, es fácil generar esta señal con el componente HAL *near* mediante la comparación de las velocidades solicitadas y reales del eje. * *spindle*.N.*brake* - (bit, out) TRUE cuando se debe aplicar el freno del husillo. * *spindle*.N.*forward* - (bit, out) TRUE cuando el husillo debe girar en sentido normal. * *spindle*.N.*index-enable* - (bit, I/O) Para un funcionamiento correcto de los movimientos sincronizados del eje, este pin debe estar conectado con el pin de habilitación de índice del codificador del husillo. * *spindle*.N.*inhibit* - (bit, in) Cuando este bit es TRUE, la velocidad del husillo se establece en 0. * *spindle*.N.*on* - (bit, out) TRUE cuando el husillo debe rotar. * *spindle*.N.*reverse* - (bit, out) TRUE cuando el husillo debe girar en sentido contrario * *spindle*.N.*revs* - (float, in) Para un funcionamiento correcto de los movimientos sincronizados del husillo, esta señal debe estar enganchada al pin de posición del codificador del husillo. La posición del codificador del husillo debe escalarse de manera que spindle-revs aumente en 1.0 por cada rotación del husillo en el sentido de las agujas del reloj (*M3*). * *spindle*.N.*speed-in* - (float, in) Retroalimentación de la velocidad real del husillo en rotaciones por segundo. Esto es utilizado en movimientos de avance por revolución (*G95*). Si su controlador codificador del husillo no tiene salida de velocidad, puede generar una adecuada enviando la posición del husillo a través de un componente *ddt*. Si no tiene un codificador de husillo, puede hacer bucle con *spindle*.N.*speed-out-rps*. * *spindle*.N.*speed-out* - (float, out) Velocidad ordenada del husillo en rotaciones por minuto. Positivo para giro horario (*M3*), negativo para giro antihorario (*M4*). * *spindle*.N.*speed-out-abs* - (float, out) Velocidad ordenada del husillo en rotaciones por minuto. Siempre será un número positivo. * *spindle*.N.*speed-out-rps* - (float, out) Velocidad del husillo ordenada en rotaciones por segundo. Positivo para sentido horario (*M3*), negativo para sentido antihorario (*M4*). * *spindle*.N.*speed-out-rps-abs* - (float, out) Velocidad del husillo ordenada en rotaciones por segundo. Siempre será un número positivo. * *spindle*.N.*orient-angle* - (float, out) Orientación del husillo especificada por M19. Valor del parámetro de la palabra R de M19 más el valor del parámetro ini [RS274NGC]ORIENT_OFFSET. * *spindle*.N.*orient-mode* - (s32, out) Modo de rotación de husillo para M19. Modo predeterminado = 0. * *spindle*.N.*orient* - (bit, out) Indica el inicio del ciclo de orientación del husillo. Activado por M19. Desactivado por M3, M4 o M5. Si spindle-orient-fault no es cero mientras que spindle-orient es verdadero, el comando M19 falla con un mensaje de error. * *spindle*.N.*is-oriented* - (bit, in) Pin de confirmación de spindle-orient. Completa el ciclo de orientacion. Si spindle-orient era verdadero cuando spindle-is-oriented se activa, el pin spindle-orient se borra y el pin spindle-locked se activa. Además, se activa el pin del freno del husillo. * *spindle*.N.*orient-fault* - (s32, in) Entrada del código de fallo para el ciclo de orientación. Cualquier valor distinto de cero provocará que el ciclo de orientación se aborte. * *spindle*.N.*lock* - (bit, out) Pin de orientación de husillo completada. Desactivado por M3, M4 o M5.

Uso del pin HAL de orientacion del husillo M19 Conceptualmente, el husillo está en uno de los modos siguientes:

- modo de rotación (predeterminado)
- modo de busqueda de orientación deseada
- modo de orientación completada.

Cuando se ejecuta un M19, el husillo cambia a *buscando orientación deseada*, y se activa el pin HAL `spindle.__N__.orient`. La posición objetivo deseada se especifica mediante los pines `spindle.__N__.orie` y `spindle.__N__.orient-fwd`, y se controla con los parámetros R y P de M19.

Se espera que la lógica de soporte HAL reaccione a `spindle.__N__.orient` moviendo el husillo a la posición deseada. Cuando esto se completa, se espera que la lógica HAL lo reconozca activando el pin `spindle.__N__.is-oriented`.

A continuación, motion reconoce esto desactivando el pin `spindle.__N__.orient` y activando el pin `spindle.__N__.locked` para indicar el modo *orientación completa*. También activa el pin `spindle.__N__.br` El husillo queda en modo *orientación completa*.

Si mientras que `spindle.__N__.orient` es verdadero, `spindle.__N__.is-oriented` no ha sido aun activado y el pin `spindle.__N__.orient-fault` tiene un valor diferente a cero, el comando M19 se cancela, se muestra un mensaje que incluye el código de fallo, y la cola de movimiento se vacía. El husillo vuelve a modo de rotación.

Además, cualquiera de los comandos M3, M4 o M5 cancela los modos de *búsqueda de orientación deseada* u *orientación completa*. Esto queda indicado al desactivar los pines `spindle-orient` y `spindle-locke`

El pin `spindle-orient-mode` refleja la palabra P M19 y debe ser interpretado de la siguiente manera:

- 0: girar en sentido horario o antihorario, el que obtenga el movimiento angular más pequeño
- 1: girar siempre en el sentido horario
- 2: girar siempre en sentido antihorario

Se puede usar con el componente HAL `orient` que proporciona un PID basado en la posición del codificador del husillo, `spindle-orient-angle` y `spindle-orient-mode`.

5.6.3. Pines y parámetros de ejes y articulaciones

Estos pines y parámetros son creados por el modulo *motmod* en tiempo real. [En las máquinas de *cinemática trivial*, hay una correspondencia uno-a-uno entre articulaciones y ejes.] Son leídos y actualizados por la función *motion-controller*.

Consulte la página del manual *motion(9)* para obtener detalles sobre los pines y parámetros.

5.6.4. iocontrol

iocontrol - acepta comandos de E/S en tiempo no-real mediante NML, interactúa con HAL.

Los pines HAL de iocontrol se activan y desactivan en un contexto de tiempo no-real. Si tiene requisitos de tiempo estrictos o simplemente necesita más E/S, considere usar en su lugar la E/S sincronizada de tiempo real proporcionada por motion.

5.6.4.1. Pines

* *iocontrol.0.coolant-flood* - (bit, out) TRUE cuando se solicita refrigerante de inundación. * *iocontrol.0.coolant-mist* - (bit, out) TRUE cuando se solicita refrigerante de niebla. * *iocontrol.0.emc-enable-in* - (bit, in) Debe ser FALSE cuando exista una condición externa de E-Stop. * *iocontrol.0.tool-change* - (bit, out) TRUE cuando se solicita un cambio de herramienta. * *iocontrol.0.tool-changed* - (bit, in) Debe ser TRUE cuando se completa un cambio de herramienta. * *iocontrol.0.tool-number* - (s32, out) El número de herramienta actual. * *iocontrol.0.tool-prep-number* - (s32, out) El número de la siguiente herramienta, de la palabra T RS274NGC. * *iocontrol.0.tool-prepare* - (bit, out) TRUE cuando se solicita una preparación de herramienta. * *iocontrol.0.tool-prepared* - (bit, in) Debe ser TRUE cuando se completa una preparación de herramienta. * *iocontrol.0.user-enable-out* - (bit, out) FALSE cuando existe una condición de parada de emergencia interna. * *iocontrol.0.user-request-enable* - (bit, out) TRUE cuando el usuario ha solicitado que se borre el E-Stop.

5.6.5. Configuraciones de INI

Muchas configuraciones INI están disponibles como pines HAL de entrada.

5.6.5.1. Pines

N se refiere a un número de articulación, *L* se refiere a una letra de eje.

- *ini*.N.*ferror* - (float, in) [JOINT_N]FERROR
- *ini*.N.*min_ferror* - (float, in) [JOINT_N]MIN_FERROR
- *ini*.N.*backlash* - (float, in) [JOINT_N]BACKLASH
- *ini*.N.*min_limit* - (float, in) [JOINT_N]MIN_LIMIT
- *ini*.N.*max_limit* - (float, in) [JOINT_N]MAX_LIMIT
- *ini*.N.*max_velocity* - (float, in) [JOINT_N]MAX_VELOCITY
- *ini*.N.*max_acceleration* - (float, in) [JOINT_N]MAX_ACCELERATION
- *ini*.N.*home* - (float, in) [JOINT_N]HOME
- *ini*.N.*home_offset* - (float, in) [JOINT_N]HOME_OFFSET
- *ini*.N.*home_offset* - (s32, in) [JOINT_N]HOME_SEQUENCE
- *ini*.L.*min_limit* - (float, in) [AXIS_L]MIN_LIMIT
- *ini*.L.*max_limit* - (float, in) [AXIS_L]MAX_LIMIT
- *ini*.L.*max_velocity* - (float, in) [AXIS_L]MAX_VELOCITY
- *ini*.L.*max_acceleration* - (float, in) [AXIS_L]MAX_ACCELERATION

---

nota
Los pines min_limit y max_limit por eje se respetan continuamente después de homing. Los pines ferror y min_ferror por eje se respetan cuando la máquina está encendida y no en posición. Los pines max_velocity y max_acceleration por eje se muestrean cuando la máquina está encendida y motion_state es libre (homing o trote) pero no son muestreados cuando se está ejecutando un programa (modo automático) o en modo mdi. Por consiguiente, cambiar los valores de pin cuando un programa se está ejecutando no tendrá efecto hasta el programa se detiene y motion_state vuelve a estar libre.

- *ini.traj_arc_blend_enable* - (bit, in) [TRAJ]ARC_BLEND_ENABLE
- *ini.traj_arc_blend_fallback_enable* - (bit, in) [TRAJ]ARC_BLEND_FALLBACK_ENABLE
- *ini.traj_arc_blend_gap_cycles* - (float, in) [TRAJ]ARC_BLEND_GAP_CYCLES
- *ini.traj_arc_blend_optimization_depth* - (float, in) [TRAJ]ARC_BLEND_OPTIMIZATION_DEPTH
- *ini.traj_arc_blend_ramp_freq* - (float, in) [TRAJ]ARC_BLEND_RAMP_FREQ

---

nota
Los pines traj_arc_blend se muestrean continuamente, pero cambiar los valores de pin mientras se ejecuta un programa puede no tener efecto inmediato debido a la cola de comandos.

- *ini.traj_default_acceleration* - (float, in) [TRAJ]DEFAULT_ACCELERATION
- *ini.traj_default_velocity* - (float, in) [TRAJ]DEFAULT_VELOCITY
- *ini.traj_max_acceleration* - (float, in) [TRAJ]MAX_ACCELERATION

Pines de planificación de trayectoria de curva S (muestreados continuamente, pueden cambiar en tiempo de ejecución):

- *ini.traj_planner_type* - (s32, in) [TRAJ]PLANNER_TYPE
- *ini.traj_max_jerk* - (float, in) [TRAJ]MAX_LINEAR_JERK

Pines de límite de sacudida por eje (donde *L* es x, y, z, a, b, c, u, v o w):

- *ini*.L.*max_jerk* - (float, in) [AXIS__L_]MAX_JERK

Pines de límite de sacudida por articulación (donde *N* es el número de articulación 0-8):

- *ini*.N.*max_jerk* - (float, in) [JOINT__N_]MAX_JERK

5.7. HAL Component List

5.7.1. Componentes

Most of the commands in the following list have their own dedicated man pages. Some will have expanded descriptions, some will have limited descriptions. From this list you know what components exist, and you can use `man` *name* on your UNIX command line to get additional information. To view the information in the man page, in a terminal window type:

```
man axis
```

The one or other setup of a UNIX system may require to explicitly specify the section of the man page. If you do not find the man page or the name of the man page is already taken by another UNIX tool with the LinuxCNC man page residing in another section, then try to explicitly specify the section, as in `man _section-no_ axis`, with *section-no* = 1 for non-realtime and 9 for realtime components.

---

nota
See also the *Man Pages* section of the docs main page or the directory listing. To search in the man pages, use the UNIX tool `apropos`.

5.7.1.1. User Interfaces (non-realtime)

axis	AXIS LinuxCNC (The Enhanced Machine Controller) GUI
axis-remote	Interfaz remota AXIS
gmoccapy	Touchy LinuxCNC Graphical User Interface
gscreen	Touchy LinuxCNC Graphical User Interface
halui	Observar los pines HAL y dar ordenes a LinuxCNC a través de NML
mdro	manual only Digital Read Out (DRO)
ngcgui	Framework for conversational G-code generation on the controller
panelui	
pyngcgui	Implementación Python de NGCGUI
touchy	AXIS - TOUCHY LinuxCNC Graphical User Interface

gladevcp	Panel de control virtual para LinuxCNC basado en widgets Glade, Gtk y HAL
gladevcp_demo	GladeVCP - used by sample configs to demonstrate Glade Virtual_demo
gremlin_view	G-code graphical preview
moveoff_gui	Interfaz gráfica de usuario para el componente moveoff
pyui	Utility for panelui
pyvcp	Panel de control virtual para LinuxCNC
pyvcp_demo	Python Virtual Control Panel demonstration component
qtvcp	Qt based virtual control panel

5axisgui	Vismach Virtual Machine GUI
hbmgui	Vismach Virtual Machine GUI
hexagui	Vismach Virtual Machine GUI
lineardelta	Vismach Virtual Machine GUI
maho600gui	hexagui - Vismach Virtual Machine GUI
max5gui	hexagui - Vismach Virtual Machine GUI
melfagui	Vismach Virtual Machine GUI
puma560gui	puma560agui - Vismach Virtual Machine GUI
pumagui	Vismach Virtual Machine GUI
rotarydelta	Vismach Virtual Machine GUI
scaragui	Vismach Virtual Machine GUI
xyzac-trt-gui	Vismach Virtual Machine GUI
xyzbc-trt-gui	Vismach Virtual Machine GUI
xyzab-tdr-gui	Vismach Virtual Machine GUI

5.7.1.2. Motion (non-realtime)

io	iocontrol - interactúa con HAL o código G en tiempo no-real
iocontrol	Interactúa con HAL o código G en tiempo no-real
mdi	Send G-code commands from the terminal to the running LinuxCNC instance
milltask	Controlador de tareas en tiempo no-real para LinuxCNC

5.7.1.3. Controladores de Hardware

elbpcom	Communicate with Mesa Ethernet cards
gs2_vfd	Componente HAL en tiempo no-real para VFDs de Automation Direct GS2
hy_gt_vfd	Componente HAL en tiempo no-real para VFDs de Huanyang serie GT
hy_vfd	Componente HAL en tiempo no-real para VFDs de Huanyang
mb2hal	MB2HAL is a generic non-realtime HAL component to communicate with one or more Modbus devices. Modbus RTU and Modbus TCP are supported.
mitsub_vfd	HAL non-realtime component for Mitsubishi A500 F500 E500 A500 D700 E700 F700-series VFDs (others may work)
monitor-xhc-hb04	Monitors the XHC-HB04 pendant and warns of disconnection
pi500_vfd	Powtran PI500 modbus driver
pmx485	Modbus communications with a Powermax Plasma Cutter
pmx485-test	Modbus communications testing with a Powermax Plasma Cutter
shuttle	controla los pines HAL con los dispositivos ShuttleXpress, ShuttlePRO y ShuttlePRO2 fabricados por Contour Design
svd-ps_vfd	Componente HAL de tiempo diferido para VDFs de SVD-P(S)
vfdb_vfd	Componente HAL de tiempo diferido para VFDs Delta VFD-B
vfs11_vfd	Componente HAL de tiempo diferido para VFDs Toshiba-Schneider VF-S11
wj200_vfd	Hitachi wj200 Modbus driver
xhc-hb04	Componente HAL en tiempo diferido para xhc-hb04 pendiente
xhc-hb04-accels	Obsolete script for jogging wheel
xhc-whb04b-6	Non-realtime jog dial HAL component for the wireless XHC WHB04B-6 USB device

5.7.1.4. Mesa and other I/O Cards (Realtime)

hal_ppmc	<< cha:pico-drivers,Controlador >> Pico Systems para servo analógico, PWM y motor a pasos
hal_bb_gpio	Driver for BeagleBone GPIO pins
hal_parport	Realtime HAL component to communicate with one or more PC parallel ports
hm2_7i43	Controlador Mesa Electronics para las tarjetas de E/S 7I43 EPP Anything con HostMot2. (Ver la página del manual para mas información)
hm2_7i90	Controlador HAL LinuxCNC para tarjetas de E/S Mesa Electronics 7I90 EPP Anything con firmware HostMot2
hm2_eth	Controlador HAL LinuxCNC para tarjetas Ethernet Anything IO de Mesa Electronics con firmware HostMot2
hm2_pci	Controlador Mesa Electronics para las tarjetas Anything I/O 5I20, 5I22, 5I23, 4I65 y 4I68 con firmware HostMot2. (Ver la página del manual para mas información)
hm2_rpspi	Controlador HAL LinuxCNC para las tarjetas de Mesa Electronics SPI Anything IO con firmware HostMot2
hm2_spi	Controlador HAL LinuxCNC para las tarjetas de Mesa Electronics SPI Anything IO con firmware HostMot2
hostmot2	Mesa Electronics driver for the HostMot2 firmware.
max31855	Support for the MAX31855 Thermocouple-to-Digital converter using bitbanged SPI
mesa_7i65	Controlador Mesa Electronics para la tarjeta servo de ocho ejes 7I65. (Ver la página del manual para mas información)
mesa_pktgyro_test	Pktuart simple test with Microstrain 3DM-GX3-15 gyro
mesa_uart	An example component demonstrating how to access the Hostmot2 UART
opto_ac5	Realtime driver for opto22 PCI-AC5 cards
pluto_servo	Controlador Pluto-P y firmware para el puerto paralelo FPGA, para servos

pluto_step	Controlador Pluto-P para el puerto paralelo FPGA, para steppers
serport	Controlador de hardware para los bits de E/S digitales del puerto serie 8250 y 16550
setsserial	An utility for setting Smart Serial NVRAM parameters
sserial	hostmot2 - Smart Serial LinuxCNC HAL driver for the Mesa Electronics HostMot2 Smart-Serial remote cards

5.7.1.5. Utilities (non-realtime)

hal-histogram	Plots the value of a HAL pin as a histogram
halcompile	Construye, compila e instala componentes HAL de LinuxCNC
halmeter	Observar pines, señales y parámetros de HAL
halcmd	Manipula el HAL de LinuxCNC desde la línea de comandos
halcmd_twopass	Utility script used when parsing HAL files. It allows to have multiple load-commands for multiple instances of the same component.
halreport	Crea un reporte sobre el estado del HAL
halrmt	Interfaz de control remoto para LinuxCNC
halrun	Manipula el HAL de LinuxCNC desde la línea de comandos
halsampler	Muestreo de datos de HAL en tiempo real
halscope	Software oscilloscope for viewing real time waveforms of HAL pins and signals
halshow	Show HAL parameters, pins and signals
halstreamer	Transmitir archivos de datos a HAL en tiempo real
haltcl	Manipula el HAL de LinuxCNC desde la línea de comandos usando Tcl
image-to-gcode	Converts bitmap images to G-code
inivar	Query an INI file
latency-histogram	Plots histogram of machine latency
latency-plot	Another way to view latency numbers
latency-test	Tests the realtime system latency
linuxcncmkdesktop	Crea un icono de escritorio para LinuxCNC
modcompile	Utility for compiling Modbus drivers
motion-logger	Log motion commands sent from LinuxCNC
pncconf	Configuration wizard for Mesa cards
sim_pin	GUI for displaying and setting one or more HAL inputs
stepconf	Configuration wizard for parallel-port based machines
update_ini	Convierte archivos INI formato 2.7 a formato 2.8
debuglevel	Sets the debug level for the non-realtime part of LinuxCNC
emccalib	Adjust ini tuning variables on the fly with save option
hal_input	Controlar los pines HAL con cualquier dispositivo de entrada de Linux, incluidos los dispositivos USB HID
linuxcnc_info	Collects information about the LinuxCNC version and the host
linuxcnc_module_helper	Controlled root access for system hardware
linuxcnc_var	Retrieves LinuxCNC variables
linuxcnc	LinuxCNC (El controlador de máquina mejorado)
linuxcnclcd	LinuxCNC Graphical User Interface for LCD character display
linuxcncrsh	Text-mode interface for commanding LinuxCNC over the network
linuxcncsvr	Allows network access to LinuxCNC internals via NML
linuxcnctop	Live LinuxCNC status description
rs274	Standalone G-code interpreter
schedrmt	Programador de tareas basado en Telnet para LinuxCNC
setup_designer	A script to configure the system for use of Qt Designer

teach-in	Jog the machine to a position, and record the state
tool_mmap_read	A component of the tool database system (an alternative to the classic tooltable)
tool_watch	A component of the tool database system (an alternative to the classic tooltable)
tooledit	Tooltable editor

5.7.1.6. Signal processing (Realtime)

and2	Compuerta AND de dos entradas. Para devolver TRUE, ambas entradas deben ser verdaderas. (and2)
bitwise	Computes various bitwise operations on the two input values
dbounce	Filtra entradas digitales ruidosas Detalles
debounce	Filter noisy digital inputs Details description
demux	Select one of several output pins by integer and/or or individual bits
demux_generic	Routes a single input signal to one of multiple outputs.
edge	Detector de flanco
estop_latch	E-stop latch
flipflop	Flip-flop tipo D
logic	Componente de función lógica general
lut5	Función lógica de 5 entradas basada en una tabla de consulta (look-up table) descripción
match8	Detector de coincidencia binaria de 8 bits
multiclick	Single-, double-, triple-, and quadruple-click detector
multiswitch	Toggles between a specified number of output bits
not	Inverter
oneshot	Generador de disparos de un pulso
or2	Compuerta OR de dos entradas
reset	Restablece una señal de E/S
select8	8-bit binary match detector.
tof	IEC TOF timer - delay falling edge on a signal
toggle	Push-on, push-off de pulsadores momentáneos
toggle2nist	Botón alternante para logica nist
ton	IEC TON timer - delay rising edge on a signal
timedelay	Equivalent of a time-delay relay.
tp	IEC TP timer - generate a high pulse of defined duration on rising edge
tristate_bit	Coloca una señal en un pin de E/S solo cuando esté habilitado, similar a un buffer triestado en electrónica
tristate_float	Coloca una señal en un pin de E/S solo cuando esté habilitado, similar a un buffer triestado en electrónica
xor2	Compuerta XOR (O exclusivo) de dos entradas

abs_s32	Calcula el valor absoluto y el signo de la señal de entrada
abs_s64	Calcula el valor absoluto y el signo de una señal de entrada entera de 64 bits
abs	Calcula el valor absoluto y el signo de una señal de entrada de punto flotante
biquad	Biquad IIR filter
blend	Realiza interpolación lineal entre dos valores
comp	Comparador de dos entradas con histéresis
counter	Counts input pulses (deprecated). Use the encoder component.
ddt	Computes the derivative of the input function.
deadzone	Returns the center if within the threshold.
div2	Quotient of two floating point inputs.

hypot	Three-input hypotenuse (Euclidean distance) calculator.
ilowpass	Low-pass filter with integer inputs and outputs
integ	Integrator
invert	Computes the inverse of the input signal.
filter_kalman	Unidimensional Kalman filter, also known as linear quadratic estimation (LQE)
knob2float	Converts counts (probably from an encoder) to a float value.
led_dim	HAL component for dimming LEDs
lowpass	Low-pass filter
limit1	Limita la señal de salida para que caiga entre mín y máx. [3]
limit2	Limits the output signal to fall between min and max. Limit its slew rate to less than maxv per second. [4]
limit3	Limita la señal de salida para que caiga entre min y max. Limita su velocidad de giro a menos de maxv por segundo. Limita su segunda derivada a menos de MaxA por segundo al cuadrado. [5].
lincurve	One-dimensional lookup table
maj3	Calcula la mayor de 3 entradas
minmax	Tracks the minimum and maximum values of the input to the outputs.
mult2	Product of two inputs.
mux16	Select from one of 16 input values (multiplexer).
mux2	Select from one of two input values (multiplexer).
mux4	Select from one of four input values (multiplexer).
mux8	Select from one of eight input values (multiplexer).
mux_generic	Select one from several input values (multiplexer).
near	Determine whether two values are roughly equal.
offset	Adds an offset to an input, and subtracts it from the feedback value.
safety_latch	latch for error signals
sample_hold	Sample and Hold.
scaled_s32_sums	Suma de cuatro entradas (cada una con escala)
scale	Applies a scale and offset to its input.
sum2	Sum of two inputs (each with a gain) and an offset.
time	Accumulated run-time timer counts HH:MM:SS of *active* input.
timedelta	Component that measures thread scheduling timing behavior.
updown	Counts up or down, with optional limits and wraparound behavior.
wcomp	Window comparator.
watchdog	Monitor one to thirty-two inputs for a *heartbeat*.
weighted_sum	Convert a group of bits to an integer.
xhc_hb04_util	xhc-hb04 convenience utility

5.7.1.7. Signal generation (Realtime)

charge_pump	Creates a square-wave for the *charge pump* input of some controller boards.
pwmgen	Software PWM/PDM generation, see description.
siggen	Signal generator, see description.
sim_encoder	Simulated quadrature encoder, see description.
stepgen	Software step pulse generation, see description.

bin2gray	Converts a number to the gray-code representation
bitmerge	Converts individual input bits into an unsigned-32

[3] Cuando la entrada es una posición, esto significa que la *posición* está limitada.
[4] When the input is a position, this means that *position* and *velocity* are limited.
[5] Cuando la entrada es una posición, esto significa que la *posición*, *velocidad*, y *aceleración* están limitadas.

bitslice	Converts an unsigned-32 input into individual bits
conv_bit_float	Convierte de bit a float
conv_bit_s32	Convierte de bit a s32
conv_bit_u32	Convierte de bit a u32
conv_float_s32	Convierte de float a s32
conv_float_u32	Convierte de float a u32
conv_s32_bit	Convierte un valor s32 a bit
conv_s32_float	Convierte un valor s32 a float
conv_s32_u32	Convierte un valor s32 a u32
conv_u32_bit	Convierte un valor u32 a bit
conv_u32_float	Convierte un valor u32 a float
conv_u32_s32	Convierte un valor u32 a s32
conv_bit_s64	Convierte un valor bit a s64
conv_bit_u64	Convierte un valor bit a u64
conv_float_s64	Convierte un valor float a s64
conv_float_u64	Convierte un valor float a u64
conv_s32_s64	Convierte un valor s32 a s64
conv_s32_u64	Convierte un valor s32 a u64
conv_s64_bit	Convierte un valor s64 a bit
conv_s64_float	Convierte un valor s64 a float
conv_s64_s32	Convierte un valor s64 a s32
conv_s64_u32	Convierte un valor s64 a u32
conv_s64_u64	Convierte un valor s64 a u64
conv_u32_s64	Convierte un valor u32 a s64
conv_u32_u64	Convierte un valor u32 a u64
conv_u64_bit	Convierte un valor u64 a bit
conv_u64_float	Convierte un valor u64 a float
conv_u64_s32	Convierte un valor u64 a s32
conv_u64_s64	Convierte un valor u64 a s64
conv_u64_u32	Convierte un valor u64 a u32
gray2bin	Converts gray-code input to binary

5.7.1.8. Kinematics (Realtime)

corexy_by_hal	CoreXY kinematics
differential	Cinemática para una transmisión diferencial
gantry	LinuxCNC HAL component for driving multiple joints from a single axis
gantrykins	Kinematics module that maps one axis to multiple joints.
genhexkins	Gives six degrees of freedom in position and orientation (XYZABC). The location of the motors is defined at compile time.
genserkins	Kinematics that can model a general serial-link manipulator with up to 6 angular joints.
gentrivkins	Correspondencia 1:1 entre articulaciones y ejes. La mayoría de las fresadoras y tornos estándar utilizan el módulo de cinemática trivial.
kins	Kinematics definitions for LinuxCNC.
lineardeltakins	Cinemática para un robot delta lineal
matrixkins	Calibrated kinematics for 3-axis machines
maxkins	Kinematics for a tabletop 5 axis mill named *max* with tilting head (B axis) and horizontal rotary mounted to the table (C axis). Provides UVW motion in the rotated coordinate system.
millturn	Switchable kinematics for a mill-turn machine
pentakins	
pumakins	Kinematics for PUMA-style robots.
rosekins	Kinematics for a rose engine
rotatekins	The X and Y axes are rotated 45 degrees compared to the joints 0 and 1.

scarakins	Kinematics for SCARA-type robots.
tripodkins	The joints represent the distance of the controlled point from three predefined locations (the motors), giving three degrees of freedom in position (XYZ).
userkins	Template for user-built kinematics
xyzab_tdr_kins	Switchable kinematics for 5 axis machine with rotary table A and B
xyzacb_trsrn	Switchable kinematics for 6 axis machine with a rotary table C, rotary spindle B and nutating spindle A
xyzbca_trsrn	Switchable kinematics for 6 axis machine with a rotary table B, rotary spindle C and nutating spindle A

5.7.1.9. Motion control (Realtime)

feedcomp	Multiply the input by the ratio of current velocity to the feed rate.
homecomp	Homing module template
limit_axis	Dynamic range based axis limits
motion	Acepta comandos de movimiento NML e interactúa con HAL en tiempo real
simple_tp	This component is a single axis simple trajectory planner, same as used for jogging in LinuxCNC.
tpcomp	Trajectory Planning (tp) module skeleton

5.7.1.10. Motor control (Realtime)

at_pid	Proportional/integral/derivative controller with auto tuning.
bldc	BLDC and AC-servo control component
clarke2	Versión de dos entradas de la transformada de Clarke
clarke3	Transformada Clarke (3 fases a cartesiana)
clarkeinv	Transformada Clarke inversa
encoder	Software counting of quadrature encoder signals, see description.
pid	Proportional/integral/derivative controller, description.
pwmgen	Software PWM/PDM generation, see description.
stepgen	Software step pulse generation, see description.

5.7.1.11. Simulation/Testing

axistest	Used to allow testing of an axis. Used In PnCConf.
rtapi_app	creates a simulated real time environment
sim-torch	A simulated plasma torch
sim_axis_hardware	A component to simulate home and limit switches
sim_home_switch	Home switch simulator
sim_matrix_kb	convert HAL pin inputs to key codes
sim_parport	A component to simulate the pins of the hal_parport component
sim_spindle	Simulated spindle with index pulse
simulate_probe	simulate a probe input

5.7.1.12. Other (Realtime)

anglejog	Jog two axes (or joints) at an angle
classicladder	Realtime software PLC based on ladder logic. See ClassicLadder chapter for more information.

charge_pump	Creates a square-wave for the *charge pump* input of some controller boards.
encoder_ratio	Electronic gear to synchronize two axes.
enum	Enumerate integer values into bits
eoffset_per_angle	Compute External Offset Per Angle
gladevcp (Realtime)	muestra paneles de control virtuales construidos con GTK/GLADE
histobins	Histogram bins utility for scripts/hal-histogram
joyhandle	Sets nonlinear joypad movements, deadbands and scales.
latencybins	Comp utility for scripts/latency-histogram
message	Display a message
moveoff	Component for HAL-only offsets
raster	Outputs laser power based upon pre programmed rastering data
sampler	Sample data from HAL in real time.
siggen	Signal generator, see description.
sphereprobe	Probe a pretend hemisphere.
threads	Creates hard realtime HAL threads.
threadtest	Component for testing thread behavior.
steptest	Used by StepConf to allow testing of acceleration and velocity values for an axis.
streamer	Stream file data into HAL in real time.
supply	Set output pins with values from parameters (deprecated).
laserpower	Scales laser power output based upon velocity input power and distance to go
lcd	Transmitir datos HAL a una pantalla LCD
matrix_kb	Convert integers to HAL pins. Optionally scan a matrix of I/O ports to create those integers.
gearchange	Select from one of two speed ranges.
orient	Provide a PID command input for orientation mode based on current spindle position, target angle and orient mode
spindle	Control a spindle with different acceleration and deceleration and optional gear change scaling
spindle_monitor	Spindle at-speed and underspeed detection
carousel	Orient a toolchanger carousel using various encoding schemes
hal_manualtoolchange	HAL non-realtime component to enable manual tool changes&.
thc	Control de la altura de antorcha utilizando una tarjeta Mesa THC o cualquier análogo a entrada de velocidad
thcud	Torch Height Control Up/Down Input
ohmic	Componente HAL de LinuxCNC que usa una tarjeta A/D Mesa THCAD para detección óhmica
plasmac	A plasma cutter controller

5.7.2. HAL API calls

```
hal_add_funct_to_thread.3
hal_bit_t.3
hal_create_thread.3
hal_del_funct_from_thread.3
```

hal_exit.3
hal_export_funct.3
hal_export_functf.3
hal_float_t.3
hal_get_lock.3
hal_init.3
hal_link.3
hal_malloc.3
hal_param_bit_new.3
hal_param_bit_newf.3
hal_param_float_new.3
hal_param_float_newf.3
hal_param_new.3
hal_param_s32_new.3
hal_param_s32_newf.3
hal_param_u32_new.3
hal_param_u32_newf.3
hal_parport.3
hal_pin_bit_new.3
hal_pin_bit_newf.3
hal_pin_float_new.3
hal_pin_float_newf.3
hal_pin_new.3
hal_pin_s32_new.3
hal_pin_s32_newf.3
hal_pin_u32_new.3
hal_pin_u32_newf.3
hal_ready.3
hal_s32_t.3
hal_set_constructor.3
hal_set_lock.3
hal_signal_delete.3
hal_signal_new.3
hal_start_threads.3
hal_type_t.3
hal_u32_t.3
hal_unlink.3
hal.3

5.7.3. RTAPI calls

EXPORT_FUNCTION.3
MODULE_AUTHOR.3
MODULE_DESCRIPTION.3
MODULE_LICENSE.3
RTAPI_MP_ARRAY_INT.3
RTAPI_MP_ARRAY_LONG.3
RTAPI_MP_ARRAY_STRING.3
RTAPI_MP_INT.3
RTAPI_MP_LONG.3
RTAPI_MP_STRING.3
rtapi.3
rtapi_app_exit.3
rtapi_app_main.3
rtapi_clock_set_period.3

```
rtapi_delay.3
rtapi_delay_max.3
rtapi_exit.3
rtapi_get_clocks.3
rtapi_get_msg_level.3
rtapi_get_time.3
rtapi_inb.3
rtapi_init.3
rtapi_module_param.3
RTAPI_MP_ARRAY_INT.3
RTAPI_MP_ARRAY_LONG.3
RTAPI_MP_ARRAY_STRING.3
RTAPI_MP_INT.3
RTAPI_MP_LONG.3
RTAPI_MP_STRING.3
rtapi_mutex.3
rtapi_outb.3
rtapi_print.3
rtapi_prio.3
rtapi_prio_highest.3
rtapi_prio_lowest.3
rtapi_prio_next_higher.3
rtapi_prio_next_lower.3
rtapi_region.3
rtapi_release_region.3
rtapi_request_region.3
rtapi_set_msg_level.3
rtapi_shmem.3
rtapi_shmem_delete.3
rtapi_shmem_getptr.3
rtapi_shmem_new.3
rtapi_snprintf.3
rtapi_task_delete.3
rtapi_task_new.3
rtapi_task_pause.3
rtapi_task_resume.3
rtapi_task_start.3
rtapi_task_wait.3
```

5.8. HAL Component Descriptions

This chapter provides details on core functionalities of LinuxCNC that demand exact timing for

- the generation of signals that is interpreted by hardware (like motors) or
- for the interpretation of signals sent by the hardware (like encoders).

5.8.1. StepGen

This component provides software based generation of step pulses in response to position or velocity commands. In position mode, it has a built in pre-tuned position loop, so PID tuning is not required. In velocity mode, it drives a motor at the commanded speed, while obeying velocity and acceleration limits. It is a realtime component only, and depending on CPU speed, etc., is capable of maximum step

rates of 10 kHz to perhaps 50 kHz. The step pulse generator block diagram shows three block diagrams, each is a single step pulse generator. The first diagram is for step type *0*, (step and direction). The second is for step type *1* (up/down, or pseudo-PWM), and the third is for step types 2 through 14 (various stepping patterns). The first two diagrams show position mode control, and the third one shows velocity mode. Control mode and step type are set independently, and any combination can be selected.

Step Pulse Generator Block Diagram position mode

:images/stepgen-block-diag.png

Loading stepgen component

```
halcmd: loadrt stepgen step_type=<type-array> [ctrl_type=<ctrl_array>]
```

<type-array>
: is a series of comma separated decimal integers. Each number causes a single step pulse generator to be loaded, the value of the number determines the stepping type.

<ctrl_array>
: is a comma separated series of *p* or *v* characters, to specify position or velocity mode.

ctrl_type
: is optional, if omitted, all of the step generators will be position mode.

For example:

```
halcmd: loadrt stepgen step_type=0,0,2 ctrl_type=p,p,v
```

Will install three step generators. The first two use step type *0* (step and direction) and run in position mode. The last one uses step type *2* (quadrature) and runs in velocity mode. The default value for *<config-array>* is *0,0,0* which will install three type *0* (step/dir) generators. The maximum number of step generators is 8 (as defined by MAX_CHAN in stepgen.c). Each generator is independent, but all are updated by the same function(s) at the same time. In the following descriptions, *<chan>* is the number of a specific generator. The first generator is number 0.

Unloading stepgen component

```
halcmd: unloadrt stepgen
```

5.8.1.1. Pines

En el tipo de paso y el tipo de control seleccionado.

- (float) `stepgen. `__<chan>__.position-cmd` - Desired motor position, in position units (position mode only).
- (float) `stepgen. `__<chan>__.velocity-cmd` - Desired motor velocity, in position units per second (velocity mode only).
- s32) `stepgen. `__<chan>__.counts` - Feedback position in counts, updated by *capture_position()*.
- (float) `stepgen. `__<chan>__.position-fb` - Feedback position in position units, updated by *capture_position()*.
- (bit) `stepgen. `__canal__.enable` - Habilita la salida de pasos. Cuando es falso, no se generan pasos.
- (bit) ‘stepgen.`__<chan>__`.step’ - Salida de pulso de pasos (sólo paso tipo 0).

- (bit) `stepgen.`__<chan>__`.dir` - Direction output (step type 0 only).
- (bit) `stepgen.`__<chan>__`.up` - UP pseudo-PWM output (step type 1 only).
- (bit) `stepgen.`__<chan>__`.down` - DOWN pseudo-PWM output (step type 1 only).
- (bit) `stepgen.`__<chan>__`.phase-A` - Phase A output (step types 2-14 only).
- (bit) `stepgen.`__<chan>__`.phase-B` - Phase B output (step types 2-14 only).
- (bit) `stepgen.`__<chan>__`.phase-C` - Phase C output (step types 3-14 only).
- (bit) `stepgen.`__<chan>__`.phase-D` - Phase D output (step types 5-14 only).
- (bit) `stepgen.`__<chan>__`.phase-E` - Phase E output (step types 11-14 only).

5.8.1.2. Parámetros

* (float) `stepgen.`__<canal>__`.position-scale` - Pasos por unidad de posición. Este parámetro se usa tanto por la salida como por la retroalimentación. * (float) `stepgen.`__<chan>__`.maxvel` - Maximum velocity, in position units per second. If 0.0, has no effect. * (float) `stepgen.`__<chan>__`.maxaccel` - Maximum accel/decel rate, in positions units per second squared. If 0.0, has no effect. * (float) `stepgen.`__<chan>__`.frequency` - The current step rate, in steps per second. * (float) `stepgen.`__<chan>__ - Length of a step pulse (step type 0 and 1) or minimum time in a given state (step types 2-14), in nano-seconds. * (float) `stepgen.`__<chan>__`.stepspace` - Minimum spacing between two step pulses (step types 0 and 1 only), in nano-seconds. Set to 0 to enable the stepgen *doublefreq* function. To use *doublefreq* the parport reset function must be enabled. * (float) `stepgen.`__<chan>__`.dirsetup` - Minimum time from a direction change to the beginning of the next step pulse (step type 0 only), in nanoseconds. * (float) `stepgen.`__<chan>__`.dirhold` - Minimum time from the end of a step pulse to a direction change (step type 0 only), in nanoseconds. * (float) `stepgen.`__<chan>__`.dirdelay` - Minimum time any step to a step in the opposite direction (step types 1-14 only), in nano-seconds. * (s32) `stepgen.`__<chan>__`.rawcounts` - The raw feedback count, updated by *make_pulses()*.

In position mode, the values of maxvel and maxaccel are used by the internal position loop to avoid generating step pulse trains that the motor cannot follow. When set to values that are appropriate for the motor, even a large instantaneous change in commanded position will result in a smooth trapezoidal move to the new location. The algorithm works by measuring both position error and velocity error, and calculating an acceleration that attempts to reduce both to zero at the same time. For more details, including the contents of the *control equation* box, consult the code.

En el modo velocidad, *maxvel* es un límite simple que se aplica a la velocidad ordenada, y *maxaccel* se usa para aumentar la frecuencia real si la velocidad ordenada cambia bruscamente. Como en el modo de posición, unos valores apropiados para estos parámetros aseguran que el motor pueda seguir el tren de pulsos generado.

5.8.1.3. Step Types

Step generator supports 15 different *step sequences*:

Step Type 0 El tipo de pasos 0 es el tipo *paso y dirección* estándar. Cuando se configura para pasos tipo 0, hay cuatro parámetros adicionales que determinan el tiempo exacto de las señales de paso y dirección. En la siguiente figura se muestra el significado de estos parámetros. Los parámetros están en nanosegundos, pero se redondearán a un número entero múltiplo del período del hilo que llama a la función de stepgen *make_pulses()*. Por ejemplo, si se llama a *make_pulses()* cada 16 µs, y la longitud steplen es 20000, entonces los pulsos de paso son 2 x 16 = 32 µs de largo. El valor predeterminado para los cuatro parámetros es 1 ns, pero el redondeo automático tiene efecto la primera vez que el código se ejecuta. Como un paso requiere *steplen* ns en alto y *stepspace* ns en bajo, la frecuencia máxima es 1,000,000,000 dividido por *(*steplen + stepspace*)*. Si *maxfreq* se establece más alto que

ese límite, se bajará automáticamente, y si es cero, permanecerá en cero, pero la frecuencia de salida seguirá limitada.

When using the parallel port driver the step frequency can be doubled using the parport reset function together with StepGen's *doublefreq* setting.

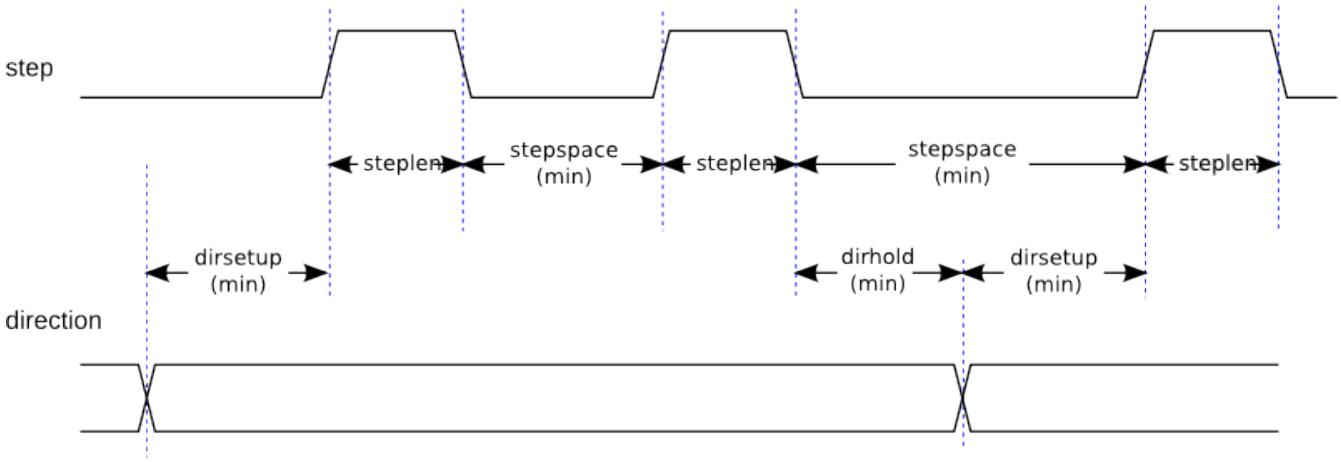


Figura 5.19: Step and Direction Timing

Step Type 1 Step type 1 has two outputs, up and down. Pulses appear on one or the other, depending on the direction of travel. Each pulse is *steplen* ns long, and the pulses are separated by at least *stepspace* ns. The maximum frequency is the same as for step type 0. If *maxfreq* is set higher than the limit it will be lowered. If *maxfreq* is zero, it will remain zero but the output frequency will still be limited.

aviso
Do not use the parport reset function with step types 2 - 14. Unexpected results can happen.

Step Type 2 - 14 Los tipos de pasos 2 a 14 están basados en estados y tienen de dos a cinco salidas. En cada paso, un contador de estado se incrementa o disminuye. Se muestran patrones de salida de dos-tres fases, cuatro fases y cinco fases como una función del contador de estado. La frecuencia máxima es 1,000,000,000 divididos por *steplen*, y como en los otros modos, *maxfreq* se reducirá si está por encima del límite.

Two-and-Three-Phase Step Types Step Types: Two-and-Three-Phase

Four-Phase Step Types Step Types: Four-Phase

Five-Phase Step Types Step Types: Five-Phase

5.8.1.4. Funciones

The component exports three functions. Each function acts on all of the step pulse generators - running different generators in different threads is not supported.

- (funct) `stepgen.make-pulses` - High speed function to generate and count pulses.
- (funct) `stepgen.update-freq` - Low speed function does position to velocity conversion, scaling and limiting.

- (funct) `stepgen.capture-position` - Low speed function for feedback, updates latches and scales position.

La función de alta velocidad *stepgen.make-pulses* debe ejecutarse en un hilo muy rápido, de 10 a 50 µs dependiendo de las capacidades de la computadora. El período de ese hilo determina la frecuencia de paso máxima, ya que *steplen*, *stepspace*, *dirsetup*, *dirhold* y *dirdelay* se redondean a un múltiplo entero del periodo del hilo en nanosegundos. Las otras dos funciones se pueden llamar a una tasa mucho más baja.

5.8.2. PWMgen

This component provides software based generation of PWM (Pulse Width Modulation) and PDM (Pulse Density Modulation) waveforms. It is a realtime component only, and depending on CPU speed, etc., is capable of PWM frequencies from a few hundred Hertz at pretty good resolution, to perhaps 10 kHz with limited resolution.

Loading PWMgen

```
loadrt pwmgen output_type=<config-array>
```

The <*config-array*> is a series of comma separated decimal integers. Each number causes a single PWM generator to be loaded, the value of the number determines the output type. The following example will install three PWM generators. There is no default value, if <*config-array*> is not specified, no PWM generators will be installed. The maximum number of frequency generators is 8 (as defined by MAX_CHAN in pwmgen.c). Each generator is independent, but all are updated by the same function(s) at the same time. In the following descriptions, <chan> is the number of a specific generator. The first generator is number 0.

Loading PWMgen Example

```
loadrt pwmgen output_type=0,1,2
```

Will install three PWM generators. The first will use an output of type *0* (PWM only), the next one will use a type 1 output (PWM and direction) and the third will use a type 2 output (UP and DOWN). There is no default value, if <*config-array*> is not not specified, no PWM generator will be installed. The maximum number of frequency generators is 8 (as defined by MAX_CHAN in pwmgen.c). Each generator is independent, but all are updated by the same function(s), at the same time. In the descriptions that follow, <*chan*> is the number of specific generators. The numbering of PWM generators starts at 0.

Unloading PWMgen

```
unloadrt pwmgen
```

5.8.2.1. Output Types

The PWM generator supports three different *output types*.

- *Output type 0* - PWM output pin only. Only positive commands are accepted, negative values are treated as zero (and will be affected by the parameter *min-dc* if it is non-zero).
- *Output type 1* - PWM/PDM and direction pins. Positive and negative inputs will be output as positive and negative PWM. The direction pin is false for positive commands, and true for negative commands. If your control needs positive PWM for both CW and CCW use the abs component to convert your PWM signal to positive value, when a negative input is input.
- *Output type 2* - UP and DOWN pins. For positive commands, the PWM signal appears on the up output, and the down output remains false. For negative commands, the PWM signal appears on the down output, and the up output remains false. Output type 2 is suitable for driving most H-bridges.

5.8.2.2. Pines

Each PWM generator will have the following pins:

- (float) pwmgen.__<chan>__.value - Command value, in arbitrary units. Will be scaled by the *scale* parameter (see below).
- (bit) pwmgen.__<chan>__.enable - Enables or disables the PWM generator outputs.

Each PWM generator will also have some of these pins, depending on the output type selected:

- (bit) pwmgen.__<chan>__.pwm - PWM (or PDM) output, (output types 0 and 1 only).
- (bit) pwmgen.__<chan>__.dir - Direction output (output type 1 only).
- (bit) pwmgen.__<chan>__.up - PWM/PDM output for positive input value (output type 2 only).
- (bit) pwmgen.__<chan>__.down - PWM/PDM output for negative input value (output type 2 only).

5.8.2.3. Parámetros

- (float) pwmgen.__<canal>__.scale - Factor de escala para convertir `value` desde unidades arbitrarias hasta valores de ciclo de trabajo. Por ejemplo, si la escala está configurada en 4000 y el valor de entrada pasado a pwmgen.__<canal>__.value es 4000, se tomará el 100 % del ciclo de trabajo (siempre activado). Si el valor es 2000, entonces será un 50 % 25 Hz onda cuadrada.
- (float) pwmgen.__<chan>__.pwm-freq - Desired PWM frequency, in Hz. If 0.0, generates PDM instead of PWM. If set higher than internal limits, next call of *update_freq()* will set it to the internal limit. If non-zero, and *dither* is false, next call of *update_freq()* will set it to the nearest integer multiple of the *make_pulses()* function period.
- (bit) pwmgen.__<chan>__.dither-pwm - If true, enables dithering to achieve average PWM frequencies or duty cycles that are unobtainable with pure PWM. If false, both the PWM frequency and the duty cycle will be rounded to values that can be achieved exactly.
- (float) pwmgen.__<chan>__.min-dc - Minimum duty cycle, between 0.0 and 1.0 (duty cycle will go to zero when disabled, regardless of this setting).
- (float) pwmgen.__<chan>__.max-dc - Maximum duty cycle, between 0.0 and 1.0.
- (float) pwmgen.__<chan>__.curr-dc - Current duty cycle - after all limiting and rounding (read only).

5.8.2.4. Funciones

El componente exporta dos funciones. Cada función actúa en todos los generadores PWM. No esta soportado ejecutar diferentes generadores en diferentes hilos.

- (funct) `pwmgen.make-pulses` - High speed function to generate PWM waveforms. The high speed function `pwmgen.make-pulses` should be run in the base (fastest) thread, from 10 to 50 µs depending on the capabilities of the computer. That thread's period determines the maximum PWM carrier frequency, as well as the resolution of the PWM or PDM signals. If the base thread is 50,000 ns then every 50 µs the module decides if it is time to change the state of the output. At 50 % duty cycle and 25 Hz PWM frequency this means that the output changes state every (1/25) s / 50 µs * 50 % = 400 iterations. This also means that you have a 800 possible duty cycle values (without dithering).
- (funct) `pwmgen.update` - Low speed function to scale and limit value and handle other parameters. This is the function of the module that does the more complicated mathematics to work out how many base-periods the output should be high for, and how many it should be low for.

5.8.3. Codificador

This component provides software based counting of signals from quadrature (or single-pulse) encoders. It is a realtime component only, and depending on CPU speed, latency, etc., is capable of maximum count rates of 10 kHz to perhaps up to 50 kHz.

El hilo base debe ser 1/2 velocidad de conteo para permitir ruido y variaciones de timing. Por ejemplo, si tiene un codificador de 100 pulsos por revolución en el husillo y sus RPM máximas son 3000, el hilo base máximo debe ser 25 µs. Un codificador de 100 pulsos por revolución tendrá 400 conteos. La velocidad del husillo de 3000 RPM = 50 RPS (revoluciones por segundo). 400 * 50 = 20,000 cuentas por segundo o 50 µs entre cuentas.

The Encoder Counter Block Diagram is a block diagram of one channel of an encoder counter.

Encoder Counter Block Diagram

:images/encoder-block-diag.png

Loading Encoder

```
halcmd: loadrt encoder [num_chan=<counters>]
```

<*contadores*> es la cantidad de contadores de codificador que desea instalar. Si no se especifica *num_chan*, serán instalados tres contadores. El número máximo de contadores es 8 (definido en MAX_CHAN en encoder.c). Cada contador es independiente, pero todos son actualizados por la(s) misma(s) función(es) al mismo tiempo. En las siguientes descripciones, <*canal*> es el número de un contador específico. El primer contador es el número 0.

Unloading Encoder

```
halcmd: unloadrt encoder
```

5.8.3.1. Pines

- `encoder._<chan>_.counter-mode` (bit, I/O) (default: FALSE) - Enables counter mode. When true, the counter counts each rising edge of the phase-A input, ignoring the value on phase-B. This is useful for counting the output of a single channel (non-quadrature) sensor. When false, it counts in quadrature mode.
- `encoder._<chan>_.missing-teeth` (s32, In) (default: 0) - Enables the use of missing-tooth index. This allows a single IO pin to provide both position and index information. If the encoder wheel has 58 teeth with two missing, spaced as if there were 60(common for automotive crank sensors) then the position-scale should be set to 60 and missing-teeth to 2. To use this mode counter-mode should be set true. This mode will work for lathe threading but not for rigid tapping.
- `encoder._<chan>_.counts` (s32, Out) - Position in encoder counts.
- `encoder._<chan>_.counts-latched` (s32, Out) - Not used at this time.
- `encoder._<canal>.index-enable` (bit, I/O) - Cuando es VERDADERO, `counts` y `position` se restablecen a cero en el siguiente flanco ascendente de la Fase Z.
  Al mismo tiempo, `index-enable` se restablece a cero para indicar que el flanco ascendente ha aparecido. El pin es bidireccional. Si `index-enable` es falso, el canal de la fase Z del codificador será ignorado, y el contador contará normalmente. El controlador del codificador nunca pondra `index-enable` en verdadero, pero otros componentes pueden hacerlo.
- `encoder._<chan>_.latch-falling` (bit, In) (default: TRUE) - Not used at this time.
- `encoder._<chan>_.latch-input` (bit, In) (default: TRUE) - Not used at this time.
- `encoder._<chan>_.latch-rising` (bit, In) - Not used at this time.

- encoder._<chan>_.min-speed-estimate (float, in) - Determine the minimum true velocity magnitude, at which velocity will be estimated as nonzero and position-interpolated will be interpolated. The units of min-speed-estimate are the same as the units of velocity. Scale factor, in counts per length unit. Setting this parameter too low will cause it to take a long time for velocity to go to 0 after encoder pulses have stopped arriving.
- encoder._<chan>_.phase-A (bit, In) - Phase A of the quadrature encoder signal.
- encoder._<chan>_.phase-B (bit, In) - Phase B of the quadrature encoder signal.
- encoder._<chan>_.phase-Z (bit, In) - Phase Z (index pulse) of the quadrature encoder signal.
- encoder._<chan>_.position (float, Out) - Position in scaled units (see position-scale).
- encoder._<chan>_.position-interpolated (float, salida) - Posición en unidades escaladas, interpoladas entre cuenta del codificador.
  La posición interpolada intenta interpolar entre cuentas del codificador, con base en la velocidad medida más reciente. Solo válido cuando la velocidad es aproximadamente constante y superior a min-speed-estimate. No lo use para control de posición, ya que su valor es incorrecto a bajas velocidades, durante las inversiones de dirección y durante los cambios de velocidad.
  Sin embargo, permite usar un codificador de pocos impulsos (incluido un impulso por revolución) para roscado en el torno, y puede tener otros usos también.
- encoder._<chan>_.position-latched (float, Out) - Not used at this time.
- encoder._<chan>_.position-scale (float, I/O) - Scale factor, in counts per length unit. For example, if position-scale is 500, then 1000 counts of the encoder will be reported as a position of 2.0 units.
- encoder._<chan>_.rawcounts (s32, In) - The raw count, as determined by update-counters. This value is updated more frequently than counts and position. It is also unaffected by reset or the index pulse.
- encoder._<chan>_.reset (bit, In) - When True, force *counts* and *position* to zero immediately.
- encoder._<chan>_.velocity (float, Out) - Velocity in scaled units per second. encoder uses an algorithm that greatly reduces quantization noise as compared to simply differentiating the *position* output. When the magnitude of the true velocity is below min-speed-estimate, the velocity output is 0.
- encoder._<chan>_.x4-mode (bit, I/O) (default: TRUE) - Enables times-4 mode. When true, the counter counts each edge of the quadrature waveform (four counts per full cycle). When false, it only counts once per full cycle. In counter-mode, this parameter is ignored. The 1x mode is useful for some jogwheels.

5.8.3.2. Parámetros

- encoder._<chan>_.capture-position.time (s32, RO)
- encoder._<chan>_.capture-position.tmax (s32, RW)
- encoder._<chan>_.update-counters.time (s32, RO)
- encoder._<chan>_.update-counter.tmax (s32, RW)

5.8.3.3. Funciones

The component exports two functions. Each function acts on all of the encoder counters - running different counters in different threads is not supported.

- (funct) encoder.update-counters - High speed function to count pulses.
- (funct) encoder.capture-position - Low speed function to update latches and scale position.

5.8.4. PID

This component provides Proportional/Integral/Derivative control loops. It is a realtime component only. For simplicity, this discussion assumes that we are talking about position loops, however this component can be used to implement other feedback loops such as speed, torch height, temperature, etc. The PID Loop Block Diagram is a block diagram of a single PID loop.

PID Loop Block Diagram

:images/pid-block-diag.png

Loading PID

```
halcmd: loadrt pid [num_chan=<loops>] [debug=1]
```

<loops> is the number of PID loops that you want to install. If *num_chan* is not specified, one loop will be installed. The maximum number of loops is 16 (as defined by MAX_CHAN in pid.c). Each loop is completely independent. In the following descriptions, *<loopnum>* is the loop number of a specific loop. The first loop is number 0.

If `debug=1` is specified, the component will export a few extra pins that may be useful during debugging and tuning. By default, the extra pins are not exported, to save shared memory space and avoid cluttering the pin list.

Unloading PID

```
halcmd: unloadrt pid
```

5.8.4.1. Pines

The three most important pins are

- (float) `pid. `__<loopnum>__.command` - The desired position, as commanded by another system component.
- (float) `pid. `__<loopnum>__.feedback` - The present position, as measured by a feedback device such as an encoder.
- (float) `pid. `__<loopnum>__.output` - A velocity command that attempts to move from the present position to the desired position.

For a position loop, `.command` and `.feedback` are in position units. For a linear axis, this could be inches, mm, meters, or whatever is relevant. Likewise, for an angular axis, it could be degrees, radians, etc. The units of the `.output` pin represent the change needed to make the feedback match the command. As such, for a position loop `.output` is a velocity, in inches/s, mm/s, degrees/s, etc. Time units are always seconds, and the velocity units match the position units. If command and feedback are in meters, then output is in meters per second.

Each loop has two pins which are used to monitor or control the general operation of the component.

- *(float) pid.*<loopnum>*.error* - Equals *.command* minus *.feedback*.
- *(bit) pid.*<loopnum>*.enable* - A bit that enables the loop. If *.enable* is false, all integrators are reset, and the output is forced to zero. If *.enable* is true, the loop operates normally.

Pines usados para informar la saturación. La saturación ocurre cuando la salida de el bloque PID está en su límite máximo o mínimo.

- *(bit) pid.*<loopnum>*.saturated* - True when output is saturated.

- *(float) pid.*<loopnum>*.saturated_s* - The time the output has been saturated.
- *(s32) pid.*<loopnum>*.saturated_count* - The time the output has been saturated.

The PID gains, limits, and other *tunable* features of the loop are available as pins so that they can be adjusted dynamically for more advanced tuning possibilities.

- *(float) pid.*<loopnum>*.Pgain* - Proportional gain
- *(float) pid.*<loopnum>*.Igain* - Integral gain
- *(float) pid.*<loopnum>*.Dgain* - Derivative gain
- *(float) pid.*<loopnum>*.bias* - Constant offset on output
- *(float) pid.*<loopnum>*.FF0* - Zeroth order feedforward - output proportional to command (position).
- *(float) pid.*<loopnum>*.FF1* - First order feedforward - output proportional to derivative of command (velocity).
- *(float) pid.*<loopnum>*.FF2* - Second order feedforward - output proportional to 2nd derivative of command (acceleration).
- *(float) pid.*<loopnum>*.deadband* - Amount of error that will be ignored
- *(float) pid.*<loopnum>*.maxerror* - Limit on error
- *(float) pid.*<loopnum>*.maxerrorI* - Limit on error integrator
- *(float) pid.*<loopnum>*.maxerrorD* - Limit on error derivative
- *(float) pid.*<loopnum>*.maxcmdD* - Limit on command derivative
- *(float) pid.*<loopnum>*.maxcmdDD* - Limit on command 2nd derivative
- *(float) pid.*<loopnum>*.maxoutput* - Limit on output value

All *max** limits are implemented so that if the value of this parameter is zero, there is no limit.

If *debug=1* was specified when the component was installed, four additional pins will be exported:

- *(float) pid.*<loopnum>*.errorI* - Integral of error.
- *(float) pid.*<loopnum>*.errorD* - Derivative of error.
- *(float) pid.*<loopnum>*.commandD* - Derivative of the command.
- *(float) pid.*<loopnum>*.commandDD* - 2nd derivative of the command.

5.8.4.2. Funciones

The component exports one function for each PID loop. This function performs all the calculations needed for the loop. Since each loop has its own function, individual loops can be included in different threads and execute at different rates.

- *(funct) pid.*<loopnum>*.do_pid_calcs* - Performs all calculations for a single PID loop.

If you want to understand the exact algorithm used to compute the output of the PID loop, refer to

- figure PID Loop Block Diagram,
- the comments at the beginning of *emc2/src/hal/components/pid.c*, and of course to
- the code itself.

The loop calculations are in the C function *calc_pid()*.

5.8.5. Simulated Encoder

The simulated encoder is exactly that. It produces quadrature pulses with an index pulse, at a speed controlled by a HAL pin. Mostly useful for testing.

Loading sim-encoder

```
halcmd: loadrt sim-encoder num_chan=<number>
```

<number> is the number of encoders that you want to simulate If not specified, one encoder will be installed. The maximum number is 8 (as defined by MAX_CHAN in sim_encoder.c).

Unloading sim-encoder

```
halcmd: unloadrt sim-encoder
```

5.8.5.1. Pines

- (float) `sim-encoder. `__<chan-num>__.speed` - The speed command for the simulated shaft.
- (bit) `sim-encoder. `__<chan-num>__.phase-A` - Quadrature output.
- (bit) `sim-encoder. `__<chan-num>__.phase-B` - Quadrature output.
- (bit) `sim-encoder. `__<chan-num>__.phase-Z` - Index pulse output.

When `.speed` is positive, `.phase-A` leads `.phase-B`.

5.8.5.2. Parámetros

- (u32) `sim-encoder. `__<chan-num>__.ppr` - Pulses Per Revolution.
- (float) `sim-encoder. `__<chan-num>__.scale` - Scale Factor for `.speed`. The default is 1.0, which means that `.speed` is in revolutions per second. Change to 60 for RPM, to 360 for degrees per second, 6.283185 (= 2*π) for radians per second, etc.

Note that pulses per revolution is not the same as counts per revolution. A pulse is a complete quadrature cycle. Most encoder counters will count four times during one complete cycle.

5.8.5.3. Funciones

El componente exporta dos funciones. Cada función afecta a todos los codificadores simulados.

- (funct) `sim-encoder.make-pulses` - High speed function to generate quadrature pulses.
- (funct) `sim-encoder.update-speed` - Low speed function to read `.speed`, do scaling, and set up `.make-pulses`.

5.8.6. Debounce

Debounce is a realtime component that can filter the glitches created by mechanical switch contacts. It may also be useful in other applications where short pulses are to be rejected.

Loading debounce

```
halcmd: loadrt debounce cfg=<config-string>
```

<config-string>
Es una serie de enteros decimales separados por comas. Cada número instala un grupo de filtros antirrebote idénticos, el número determina cuántos filtros están en el grupo.

Loading debounce Example

```
halcmd: loadrt debounce cfg=1,4,2
```

will install three groups of filters. Group 0 contains one filter, group 1 contains four, and group 2 contains two filters. The default value for *<config-string>* is "1" which will install a single group containing a single filter. The maximum number of groups 8 (as defined by MAX_GROUPS in debounce.c). The maximum number of filters in a group is limited only by shared memory space. Each group is completely independent. All filters in a single group are identical, and they are all updated by the same function at the same time. In the following descriptions, *<G>* is the group number and *<F>* is the filter number within the group. The first filter is group 0, filter 0.

Unloading debounce

```
halcmd: unloadrt debounce
```

5.8.6.1. Pines

Each individual filter has two pins.

- (bit) `debounce.`__<G>__.__<F>__.in` - Input of filter *<F>* in group *<G>*.
- (bit) `debounce.`__<G>__.__<F>__.out` - Output of filter *<F>* in group *<G>*.

5.8.6.2. Parámetros

Each group of filters has one parameter[6].

- (s32) `debounce.`__<G>__.delay` - Filter delay for all filters in group *<G>*.

The filter delay is in units of thread periods. The minimum delay is zero. The output of a zero delay filter exactly follows its input - it doesn't filter anything. As `.delay` increases, longer and longer glitches are rejected. If `.delay` is 4, all glitches less than or equal to four thread periods will be rejected.

5.8.6.3. Funciones

Each group of filters has one function, which updates all the filters in that group *simultaneously*. Different groups of filters can be updated from different threads at different periods.

- (funct) `debounce.`*<G>* - Updates all filters in group *<G>*.

[6] Each individual filter also has an internal state variable. There is a compile time switch that can export that variable as a parameter. This is intended for testing, and simply wastes shared memory under normal circumstances.

5.8.7. SigGen

SigGen is a realtime component that generates square, triangle, and sine waves. It is primarily used for testing.

Loading siggen

```
halcmd: loadrt siggen [num_chan=<chans>]
```

<chans>
is the number of signal generators that you want to install. If *numchan* is not specified, one signal generator will be installed. The maximum number of generators is 16 (as defined by MAX_CHAN in siggen.c). Each generator is completely independent. In the following descriptions is

<chan>
the number of a specific signal generator (the numbers start at 0).

Unloading siggen

```
halcmd: unloadrt siggen
```

5.8.7.1. Pines

Each generator has five output pins.

- (float) `siggen. `__<chan>__.sine` - Sine wave output.
- (float) `siggen. `__<chan>__.cosine` - Cosine output.
- (float) `siggen. `__<chan>__.sawtooth` - Sawtooth output.
- (float) `siggen. `__<chan>__.triangle` - Triangle wave output.
- (float) `siggen. `__<chan>__.square` - Square wave output.

All five outputs have the same frequency, amplitude, and offset.

In addition to the output pins, there are three control pins:

- (float) `siggen. `__<chan>__.frequency` - Sets the frequency in Hertz, default value is 1 Hz.
- (float) `siggen. `__<chan>__.amplitude` - Sets the peak amplitude of the output waveforms, default is 1.
- (float) `siggen. `__<chan>__.offset` - Sets DC offset of the output waveforms, default is 0.

For example, if `siggen.0.amplitude` is 1.0 and `siggen.0.offset` is 0.0, the outputs will swing from -1.0 to +1.0. If `siggen.0.amplitude` is 2.5 and `siggen.0.offset` is 10.0, then the outputs will swing from 7.5 to 12.5.

5.8.7.2. Parámetros

None. [7]

[7] Prior to version 2.1, frequency, amplitude, and offset were parameters. They were changed to pins to allow control by other components.

5.8.7.3. Funciones

- (funct) `siggen. `__<chan>__.update` - Calculates new values for all five outputs.

5.8.8. lut5

The `lut5` component is a 5 input logic component based on a look up table.

- `lut5` does not use floating point math.

Loading lut5

```
loadrt lut5 [count=N|names=name1[,name2...]]
addf lut5.N servo-thread | base-thread
setp lut5.N.function 0xN
```

lut5 Computing Function To compute the hexadecimal number for the function starting from the top put a 1 or 0 to indicate if that row would be true or false. Next write down every number in the output column starting from the top and writing them from right to left. This will be the binary number. Using a calculator with a program view like the one in Ubuntu enter the binary number and then convert it to hexadecimal and that will be the value for function.

Cuadro 5.28: `lut5` Look Up Table

Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Output
0	0	0	0	0	
0	0	0	0	1	
0	0	0	1	0	
0	0	0	1	1	
0	0	1	0	0	
0	0	1	0	1	
0	0	1	1	0	
0	0	1	1	1	
0	1	0	0	0	
0	1	0	0	1	
0	1	0	1	0	
0	1	0	1	1	
0	1	1	0	0	
0	1	1	0	1	
0	1	1	1	0	
0	1	1	1	1	
1	0	0	0	0	
1	0	0	0	1	
1	0	0	1	0	
1	0	0	1	1	
1	0	1	0	0	
1	0	1	0	1	
1	0	1	1	0	
1	0	1	1	1	
1	1	0	0	0	
1	1	0	0	1	
1	1	0	1	0	
1	1	0	1	1	
1	1	1	0	0	
1	1	1	0	1	

Cuadro 5.28: (continued)

Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Output
1	1	1	1	0	
1	1	1	1	1	

lut5 Two Inputs Example En la siguiente tabla, hemos seleccionado el estado de salida para cada línea que deseamos que sea verdadera.

Cuadro 5.29: `lut5` Two Inputs Example Look Up Table

Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Output
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	1	1

Looking at the output column of our example we want the output to be on when Bit 0 or Bit 0 and Bit1 is on and nothing else. The binary number is *b1010* (rotate the output 90 degrees CW). Enter this number into the calculator then change the display to hexadecimal and the number needed for function is *0xa*. The hexadecimal prefix is *0x*.

5.9. HAL Component Generator

5.9.1. Introducción

This section introduces to the compilation HAL components, i.e. the addition of some machinists' knowledge on how to deal with the machine. It should be noted that such components do not necessarily deal with the hardware directly. They often do, but not necessarily, e.g. there could be a component to convert between imperial and metric scales, so this section does not require to dive into the interaction with hardware.

Writing a HAL component can be a tedious process, most of it in setup calls to *rtapi_* and *hal_* functions and associated error checking. *halcompile* will write all this code for you, automatically. Compiling a HAL component is also much easier when using *halcompile*, whether the component is part of the LinuxCNC source tree, or outside it.

For instance, when coded in C, a simple component such as "ddt" is around 80 lines of code. The equivalent component is very short when written using the *halcompile* preprocessor:

Simple Component Example

```
component ddt "Calcular la derivada de la función de entrada";
pin in float in;
pin out float out;
variable double old;
option period no;
function _;
license "GPL"; // indica GPL v2 o posterior
;;
float tmp = in;
out = (tmp - old) / fperiod;
old = tmp;
```

5.9.2. Instalación

Para compilar un componente, si está trabajando con una versión instalada de LinuxCNC, necesitará instalar los paquetes de desarrollo ya sea usando Synaptic desde el menú principal *System -> Administration -> Synaptic package manager* o ejecutando en una ventana de terminal uno de los comandos siguientes:

Installation of Development packages for LinuxCNC

```
sudo apt install linuxcnc-dev
# or
sudo apt install linuxcnc-uspace-dev
```

Another method is using the Synaptic package manager, from the Applications menu, to install the `linuxcnc-dev` or `linuxcnc-uspace-dev` packages.

5.9.3. Compiling

5.9.3.1. Inside the source tree

Place the `.comp` file in the source directory `linuxcnc/src/hal/components` and re-run `make`. *Comp* files are automatically detected by the build system.

If a `.comp` file is a driver for hardware, it may be placed in `linuxcnc/src/hal/drivers` and will be built unless LinuxCNC is configured as a non-realtime simulator.

5.9.3.2. Realtime components outside the source tree

halcompile can process, compile, and install a realtime component in a single step, placing `rtexample.ko` in the LinuxCNC realtime module directory:

```
[sudo] halcompile --install rtexample.comp
```

---

nota
sudo (for root permission) is needed when using LinuxCNC from a deb package install. When using a Run-In-Place (RIP) build, root privileges should not be needed.

Or, it can process and compile in one step, leaving `example.ko` (or `example.so` for the simulator) in the current directory:

```
halcompile --compile rtexample.comp
```

Or it can simply process, leaving `example.c` in the current directory:

```
halcompile rtexample.comp
```

halcompile can also compile and install a component written in C, using the `--install` and `--compile` options shown above:

```
[sudo] halcompile --install rtexample2.c
```

man-format documentation can also be created from the information in the declaration section:

```
halcompile --document -o example.9 rtexample.comp
```

The resulting manpage, *example.9* can be viewed with

```
man ./example.9
```

or copied to a standard location for manual pages.

5.9.3.3. Non-realtime components outside the source tree

halcompile can process, compile, install, and document non-realtime components:

```
halcompile non-rt-example.comp
halcompile --compile non-rt-example.comp
[sudo] halcompile --install non-rt-example.comp
halcompile --document non-rt-example.comp
```

For some libraries (for example modbus) it might be necessary to add extra compiler and linker arguments to enable the compiler to find and link the libraries. In the case of .comp files this can be done via ”option” statements in the `.comp` file. For `.c` files this is not possible so the `--extra-compile-args` and `--extra-link-args` parameters can be used instead. As an example, this command line can be used to compile the `vfdb_vfd.c` component out-of-tree.

```
halcompile --userspace --install --extra-compile-args=”-I/usr/include/modbus” --extra-link- ↩
    args=”-lm -lmodbus -llinuxcncini” vfdb_vfd.c
```

nota
The effect of using both command-line and in-file extra-args is undefined.

5.9.4. Using a Component

Components need to be loaded and added to a thread before it can be employed. The provided functionality can then be invoked directly and repeatedly by one of the threads or it is called by other components that have their own respective triggers.

Example HAL script for installing a component (ddt) and executing it every millisecond.

```
loadrt threads name1=servo-thread period1=1000000
loadrt ddt
addf ddt.0 servo-thread
```

More information on `loadrt` and `addf` can be found in the HAL Basics.

To test your component you can follow the examples in the HAL Tutorial.

5.9.5. Definitions

- **component** - A component is a single real-time module, which is loaded with `Halcmd loadrt`. One `.comp` file specifies one component. The component name and file name must match.
- **instance** - A component can have zero or more instances. Each instance of a component is created equal (they all have the same pins, parameters, functions, and data) but behave independently when their pins, parameters, and data have different values.
- **singleton** - Es posible que un componente sea un ”singleton”, en cuyo caso se crea una instancia única. Rara vez tiene sentido escribir un componente *singleton* a menos que, literalmente, solo pueda haber un único objeto de ese tipo en el sistema (por ejemplo, un componente cuyo propósito es proporcionar un pin con la hora actual de UNIX, o un controlador de hardware para el altavoz interno de PC).

5.9.6. Instance creation

For a singleton, the one instance is created when the component is loaded.

For a non-singleton, the *count* module parameter determines how many numbered instances are created. If *count* is not specified, the *names* module parameter determines how many named instances are created. If neither *count* nor *names* is specified, a single numbered instance is created.

5.9.7. Implicit Parameters

Functions are implicitly passed the *period* parameter which is the time in nanoseconds of the last period to execute the component. Functions which use floating-point can also refer to *fperiod* which is the floating-point time in seconds, or (period*1e-9). This can be useful in components that need the timing information. See also *option period* below.

5.9.8. Sintaxis

A `.comp` file consists of a number of declarations, followed by `;;` on a line of its own, followed by C code implementing the module's functions.

Declarations include:

- *component HALNAME (DOC);*
- *pin PINDIRECTION TYPE HALNAME ([SIZE]|[MAXSIZE: CONDSIZE]) (if CONDITION) (= STARTVALUE) (DOC) ;*
- *param PARAMDIRECTION TYPE HALNAME ([SIZE]|[MAXSIZE: CONDSIZE]) (if CONDITION) (= STARTVALUE) (DOC) ;*
- *function HALNAME (fp | nofp) (DOC);*
- *option OPT (VALUE);*
- *variable CTYPE STARREDNAME ([SIZE]);*
- *description DOC;*
- *examples DOC;*
- *notes DOC;*
- *see_also DOC;*
- *license LICENSE;*
- *author AUTHOR;*
- *include HEADERFILE;*

Parentheses indicate optional items. A vertical bar indicates alternatives. Words in *CAPITALS* indicate variable text, as follows:

- *NAME* - A standard C identifier
- *STARREDNAME* - A C identifier with zero or more * before it. This syntax can be used to declare instance variables that are pointers. Note that because of the grammar, there may not be whitespace between the * and the variable name.

- *HALNAME* - An extended identifier. When used to create a HAL identifier, any underscores are replaced with dashes, and any trailing dash or period is removed, so that ”this_name_” will be turned into ”this-name”, and if the name is ”_”, then a trailing period is removed as well, so that ”function _” gives a HAL function name like ”component.<num>” instead of ”component.<num>.”

  If present, the prefix *hal_* is removed from the beginning of the component name when creating pins, parameters and functions.

In the HAL identifier for a pin or parameter, # denotes an array item, and must be used in conjunction with a *[SIZE]* declaration. The hash marks are replaced with a 0-padded number with the same length as the number of # characters.

When used to create a C identifier, the following changes are applied to the HALNAME:

1. Any ”#” characters, and any ”.”, ”_” or ”-” characters immediately before them, are removed.
2. Any remaining ”.” and ”-” characters are replaced with ”_”.
3. Repeated ”_” characters are changed to a single ”_” character.

A trailing ”_” is retained, so that HAL identifiers which would otherwise collide with reserved names or keywords (e.g., *min*) can be used.

HALNAME	C Identifier	HAL Identifier
x_y_z	x_y_z	x-y-z
x-y.z	x_y_z	x-y.z
x_y_z_	x_y_z_	x-y-z
x.##.y	x_y(MM)	x.MM.z
x.##	x(MM)	x.MM

- *if CONDITION* - Una expresión que involucra la variable *personality* , que no es cero cuando se debe crear el pin o parámetro.
- *SIZE* - A number that gives the size of an array. The array items are numbered from 0 to *SIZE*-1.
- *MAXSIZE : CONDSIZE* - A number that gives the maximum size of the array, followed by an expression involving the variable *personality* and which always evaluates to less than *MAXSIZE*. When the array is created its size will be *CONDSIZE*.
- *DOC* - A string that documents the item. String can be a C-style ”double quoted” string, like:

```
”Selects the desired edge: TRUE means falling, FALSE means rising”
```

  or a Python-style ”triple quoted” string, which may include embedded newlines and quote characters, such as:

```
”””The effect of this parameter, also known as ”the orb of zot”,
will require at least two paragraphs to explain.

Hopefully these paragraphs have allowed you to understand ”zot”
better.”””
```

  Or a string may be preceded by the literal character *r*, in which case the string is interpreted like a Python raw-string.

  The documentation string is in ”groff -man” format. For more information on this markup format, see *groff_man(7)*. Remember that *halcompile* interprets backslash escapes in strings, so for instance to set the italic font for the word *example*, write:

```
”\\fIexample\\fB”
```

In this case, r-strings are particularly useful, because the backslashes in an r-string need not be doubled:

```
r"\fIexample\fB"
```

- *TYPE* - One of the HAL types: *bit*, *s32*, *u32*, *s64*, *u64* or *float*. The names *signed* and *unsigned* may also be used for *s32* and *u32* but *s32* and *u32* are preferred.
- *PINDIRECTION* - One of the following: *in*, *out*, or *io*. A component sets a value for an *out* pin, it reads a value from an *in* pin, and it may read or set the value of an *io* pin.
- *PARAMDIRECTION* - One of the following: *r* or *rw*. A component sets a value for a *r* parameter, and it may read or set the value of a *rw* parameter.
- *STARTVALUE* - Specifies the initial value of a pin or parameter. If it is not specified, then the default is *0* or *FALSE*, depending on the type of the item.
- *HEADERFILE* - The name of a header file, either in double-quotes (`include "myfile.h";`) or in angle brackets (`include <systemfile.h>;`). The header file will be included (using C's #include) at the top of the file, before pin and parameter declarations.

5.9.8.1. HAL functions

- *fp* - Indicates that the function performs floating-point calculations.
- *nofp* - Indicates that it only performs integer calculations. If neither is specified, *fp* is assumed. Neither *halcompile* nor gcc can detect the use of floating-point calculations in functions that are tagged *nofp*, but the use of such operations results in undefined behavior.

5.9.8.2. Opciones

The currently defined options are:

- *option singleton yes* - (valor predeterminado: no)
  No crear el parámetro de módulo *count*, y siempre crear una sola instancia. Con *singleton*, los elementos se denominan *nombre-componente.nombre-elemento* y sin *singleton*, los elementos, para las instancias numeradas, se nombran *nombre-componente.<número>.nombre-del-elemento*.
- *option default_count number* - (valor predeterminado: 1)
  Normalmente, el parámetro de módulo *count* se establece de manera predeterminada en 1. Si se especifica, *count* cambiará a este valor por defecto.
- *option count_function yes* - (predeterminado: no)
  Normalmente, el número de instancias a crear se especifica en el parámetro del módulo *count*; si se especifica *count_function*, se usa en su lugar el valor devuelto por la función *int get_count(void)* , y el parámetro del módulo *count* no se define.
- *option rtapi_app no* - (predeterminado: yes)
  Normalmente, las funciones `rtapi_app_main()` y `rtapi_app_exit()` son definidas automáticamente. Con *option rtapi_app no*, no lo son, y debe ser provistas en código C. Use los siguientes prototipos:

```
‘int rtapi_app_main(void);‘

‘void rtapi_app_exit(void);‘
```

  When implementing your own `rtapi_app_main()`, call the function `int export(char *prefix, long extra_arg)` to register the pins, parameters, and functions for `prefix`.

- *option data TYPE* - (predeterminado: ninguno) **obsoleto**
  Si se especifica, cada instancia del componente tendrá asociado un bloque de datos de tipo *TYPE* (que puede ser un tipo simple como *float* o el nombre de un tipo creado con *typedef*). En los componentes nuevos, se debe usar *variable* en su lugar.

- *option extra_setup yes* - (predeterminado: no)
  Si se especifica, llama a la función definida por *EXTRA_SETUP* en cada instancia. Si usa *rtapi_app_main* definido automáticamente, *extra_arg* es el número de esta instancia.

- *option extra_cleanup yes* - (predeterminado: no)
  Si se especifica, llama a la función definida por *EXTRA_CLEANUP* desde *rtapi_app_exit* definido automáticamente, o si se detecta un error en *rtapi_app_main* definido automáticamente.

- *option userspace yes* - (predeterminado: no)
  Si se especifica, este archivo describe un componente de espacio de usuario (es decir, no en tiempo real), en lugar de uno regular (es decir, en tiempo real). Un componente de espacio de usuario puede no tener funciones definidas por la directiva *function*. En cambio, después de que todas las instancias se construyan, se llama a la función C `void user_mainloop(void);`. Cuando esta función retorna, el componente sale. Normalmente, *user_mainloop()* usará *FOR_ALL_INSTS()* para realizar la acción de actualización para cada instancia, luego se detiene un tiempo corto. Otra acción común en *user_mainloop()* puede ser llamar al bucle del controlador de eventos de un toolkit de GUI.

- *option userinit yes* - (predeterminado: no)
  Esta opción se ignora si la opción *userspace* (ver arriba) está configurada en *no*. Si se especifica *userinit*, la función *userinit(argc, argv)* se llama antes que *rtapi_app_main()* (y por lo tanto antes de la llamada a *hal_init()*). Esta función puede procesar los argumentos de la línea de comando o tomar otras acciones. Su tipo de retorno es *void*; puede llamar a *exit()* si desea terminar en lugar de crear un componente HAL (p. ej., porque los argumentos de línea de comando no eran válidos).

- *option extra_link_args "..."* - (default: "")
  This option is ignored if the option *userspace* (see above) is set to *no*. When linking a non-realtime component, the arguments given are inserted in the link line. Note that because compilation takes place in a temporary directory, "-L." refers to the temporary directory and not the directory where the .comp source file resides. This option can be set in the halcompile command-line with -extra-link-args="-L.....". This alternative provides a way to set extra flags in cases where the input file is a .c file rather than a .comp file.

- *option extra_compile_args "..."* - (default: "")
  This option is ignored if the option *userspace* (see above) is set to *no*. When compiling a non-realtime component, the arguments given are inserted in the compiler command line. If the input file is a .c file this option can be set in the halcompile command-line with --extra-compile-args="-I.....". This alternative provides a way to set extra flags in cases where the input file is a .c file rather than a .comp file.

- *option homemod yes* - (predeterminado: no)
  Es un módulo personalizado de homing que se carga usando `[EMCMOT]HOMEMOD=`*nombre-de-módulo*.

- *option tpmod yes* - (predeterminado: no)
  Es un módulo de planeación de trayectoria personalizado cargado con `[TRAJ]TPMOD=`*nombre-del-módulo* .

- *option period no* - (default: yes)
  Control the implicit *period* parameter of the function(s) defined in the component. A standard function has an implicit parameter *period*. Many components do no use the *period* parameter and would cause a "unused parameter" compiler warning. Setting *option period no* creates a function declaration omitting the *period* parameter preventing the warning. Setting this option will also prevent *fperiod* from being defined, as it depends on *period*.

Si el VALOR de una opción no está especificado, entonces es equivalente a especificar *option ... yes*.
El resultado de asignar un valor inapropiado a una opción no está definido.

El resultado de usar cualquier otra opción no está definido.

5.9.8.3. License and Authorship

- `LICENSE` - Especifica la licencia del módulo para la documentación y para la declaración del módulo MODULE_LICENSE(). Por ejemplo, para especificar que la licencia del módulo es GPL v2 o posterior:

  `‘license ”GPL”; // indicates GPL v2 or later‘`

  Para obtener información adicional sobre el significado de MODULE_LICENSE() e identificadores de licencia adicionales, consulte *<linux/module.h>* o la página de manual *rtapi_module_param(3)*.

  This declaration is **required**.
- `AUTHOR` - Specify the author of the module for the documentation.

5.9.8.4. Per-instance data storage

- `variable CTYPE STARREDNAME; + variable CTYPE STARREDNAME[SIZE]; + variable CTYPE STARRED = DEFAULT; + variable CTYPE STARREDNAME[SIZE] = DEFAULT;`

  Declare a per-instance variable *STARREDNAME* of type *CTYPE*, optionally as an array of *SIZE* items, and optionally with a default value *DEFAULT*. Items with no *DEFAULT* are initialized to all-bits-zero. *CTYPE* is a simple one-word C type, such as `float`, `u32`, `s32`, `int`, etc. Access to array variables uses square brackets.

If a variable is to be of a pointer type, there may not be any space between the ”*” and the variable name. Therefore, the following is acceptable:

```
variable int *example;
```

But the following are not:

```
variable int* badexample;
variable int * badexample;
```

5.9.8.5. Comentarios

C++-style one-line comments (`//...`) and C-style multi-line comments (`/* ... */`) are both supported in the declaration section.

5.9.9. Restrictions

Though HAL permits a pin, a parameter, and a function to have the same name, *halcompile* does not.

Variable and function names that can not be used or are likely to cause problems include:

- Anything beginning with _comp.
- *comp_id*
- *fperiod*
- *rtapi_app_main*
- *rtapi_app_exit*
- *extra_setup*
- *extra_cleanup*

5.9.10. Convenience Macros

Based on the items in the declaration section, *halcompile* creates a C structure called `struct __comp_state`. However, instead of referring to the members of this structure (e.g., `*(inst->name)`), they will generally be referred to using the macros below. The details of `struct __comp_state` and these macros may change from one version of *halcompile* to the next.

- `FUNCTION( `__name__)` - Use this macro to begin the definition of a realtime function, which was previously declared with *function NAME*. The function includes a parameter *period* which is the integer number of nanoseconds between calls to the function. See also *option period* above.
- `EXTRA_SETUP()` - Use this macro to begin the definition of the function called to perform extra setup of this instance. Return a negative UNIX *errno* value to indicate failure (e.g., *return -EBUSY* on failure to reserve an I/O port), or 0 to indicate success.
- `EXTRA_CLEANUP()` - Use this macro to begin the definition of the function called to perform extra cleanup of the component. Note that this function must clean up all instances of the component, not just one. The ”pin_name”, ”parameter_name”, and ”data” macros may not be used here.
- *pin_name* or *parameter_name* - For each pin *pin_name* or param *parameter_name* there is a macro which allows the name to be used on its own to refer to the pin or parameter. When *pin_name* or *parameter_name* is an array, the macro is of the form *pin_name(idx)* or *param_name(idx)*, where *idx* is the index into the pin array. When the array is a variable-sized array, it is only legal to refer to items up to its *condsize*.

  When the item is a conditional item, it is only legal to refer to it when its *condition* evaluated to a nonzero value.
- *variable_name* - Para cada variable *variable_name* hay una macro que permite usar el nombre que se utiliza para referirse a la variable. Cuando *variable_name* es una matriz, se usa el subíndice estilo C normal: *variable_name[idx]*.
- *data* - If ”option data” is specified, this macro allows access to the instance data.
- *fperiod* - The floating-point number of seconds between calls to this realtime function. See also *option period* above.
- `FOR_ALL_INSTS() {...}` - For non-realtime components. This macro iterates over all the defined instances. Inside the body of the loop, the *pin_name*, *parameter_name*, and *data* macros work as they do in realtime functions.

5.9.11. Components with one function

Si un componente tiene solo una función y la cadena `”FUNCTION”` no aparece después de `;;`, el resto sera tomado como el cuerpo de una función única del componente. Ver Simple Comp como ejemplo.

5.9.12. Component Personality

If a component has any pins or parameters with an ”if condition” or ”[maxsize : condsize]”, it is called a component with *personality*. The *personality* of each instance is specified when the module is loaded. *Personality* can be used to create pins only when needed. For instance, personality is used in the *logic* component, to allow for a variable number of input pins to each logic gate and to allow for a selection of any of the basic boolean logic functions *and*, *or*, and *xor*.

The default number of allowed *personality* items is a compile-time setting (64). The default applies to numerous components included in the distribution that are built using halcompile.

To alter the allowed number of personality items for user-built components, use the *--personalities* option with halcompile. For example, to allow up to 128 personality times:

```
  [sudo] halcompile --personalities=128 --install ...
```

When using components with personality, normal usage is to specify a personality item for **each** specified component instance. Example for 3 instances of the logic component:

```
loadrt logic names=and4,or3,nand5, personality=0x104,0x203,0x805
```

nota
If a loadrt line specifies more instances than personalities, the instances with unspecified personalities are assigned a personality of 0. If the requested number of instances exceeds the number of allowed personalities, personalities are assigned by indexing modulo the number of allowed personalities. A message is printed denoting such assignments.

5.9.13. Ejemplos

5.9.13.1. constant

Note that the declaration ”function _” creates functions named ”constant.0”, etc. The file name must match the component name.

```
component constant;
pin out float out;
param r float value = 1.0;
option period no;
function _;
license ”GPL”; // indica GPL v2 o posterior
;;
FUNCTION(_) { out = value; }
```

5.9.13.2. sincos

This component computes the sine and cosine of an input angle in radians. It has different capabilities than the ”sine” and ”cosine” outputs of siggen, because the input is an angle, rather than running freely based on a ”frequency” parameter.

The pins are declared with the names *sin_* and *cos_* in the source code so that they do not interfere with the functions *sin()* and *cos()*. The HAL pins are still called *sincos.<num>.sin*.

```
component sincos;
pin out float sin_;
pin out float cos_;
pin in float theta;
option period no;
function _;
license ”GPL”; // indica GPL v2 o posterior
;;
#include <rtapi_math.h>
FUNCTION(_) { sin_ = sin(theta); cos_ = cos(theta); }
```

5.9.13.3. out8

Este componente es un controlador para una tarjeta *ficticia* llamada "out8", que tiene 8 pines de salida digital que son tratado como un solo valor de 8 bits. Puede haber un número variable de tales tarjetas en el sistema, y pueden estar en varias direcciones. El pin es llamado *out_* porque *out* es un identificador utilizado en *<asm/io.h>*. Eso ilustra el uso de *EXTRA_SETUP* y *EXTRA_CLEANUP* para solicitar una región de E/S y luego liberarla en caso de error o cuando el módulo sea descargado.

```
component out8;
pin out unsigned out_ "Output value; only low 8 bits are used";
param r unsigned ioaddr;

function _;

option period no;
option count_function;
option extra_setup;
option extra_cleanup;
option constructable no;

license "GPL"; // indicates GPL v2 or later
;;
#include <asm/io.h>

#define MAX 8
int io[MAX] = {0,};
RTAPI_MP_ARRAY_INT(io, MAX, "I/O addresses of out8 boards");

int get_count(void) {
    int i = 0;
    for(i=0; i<MAX && io[i]; i++) { /* Nothing */ }
    return i;
}

EXTRA_SETUP() {
    if(!rtapi_request_region(io[extra_arg], 1, "out8")) {
        // set this I/O port to 0 so that EXTRA_CLEANUP does not release the IO
        // ports that were never requested.
        io[extra_arg] = 0;
        return -EBUSY;
    }
    ioaddr = io[extra_arg];
    return 0;
}

EXTRA_CLEANUP() {
    int i;
    for(i=0; i < MAX && io[i]; i++) {
        rtapi_release_region(io[i], 1);
    }
}

FUNCTION(_) { outb(out_, ioaddr); }
```

5.9.13.4. hal_loop

```
component hal_loop;
pin out float example;
```

Este fragmento de componente ilustra el uso del prefijo *hal_* en el nombre de un componente.

`loop` is a common name, and the `hal_` prefix avoids potential name collisions with other unrelated software. For example, on RTAI realtime systems realtime code runs in the kernel, so if the component were named just `loop` it could easily conflict with the standard `loop` kernel module.

When loaded, *halcmd show comp* will show a component called *hal_loop*. However, the pin shown by *halcmd show pin* will be *loop.0.example*, not *hal-loop.0.example*.

5.9.13.5. arraydemo

This realtime component illustrates use of fixed-size arrays:

```
component arraydemo "4-bit Shift register";
pin in bit in;
pin out bit out-# [4];
option period no;
function _ nofp;
license "GPL"; // indica GPL v2 o posterior
;;
int i;
for(i=3; i>0; i--) out(i) = out(i-1);
out(0) = in;
```

5.9.13.6. rand

This non-realtime component changes the value on its output pin to a new random value in the range (0,1) about once every 1 ms.

```
component rand;
option userspace;

pin out float out;
license "GPL"; // indicates GPL v2 or later
;;
#include <unistd.h>

void user_mainloop(void) {
    while(1) {
        usleep(1000);
        FOR_ALL_INSTS() out = drand48();
    }
}
```

5.9.13.7. logic (using personality)

This realtime component shows how to use "personality" to create variable-size arrays and optional pins.

```
component logic "componente LinuxCNC HAL que proporciona funciones lógicas experimentales";
pin in bit in-##[16 : personality & 0xff];
pin out bit and if personality & 0x100;
pin out bit or if personality & 0x200;
pin out bit xor if personality & 0x400;
option period no;
function _ nofp;
description """
```

```
Componente ’función lógica’ general experimental.  Puede realizar ’and’, ’or’
y ’xor’ de hasta 16 entradas.  Determina el valor apropiado para ’personalidad’
añadiendo:
.IP \\(bu 4
El número de pines de entrada, generalmente de 2 a 16
.IP \\(bu
256 (0x100) si se desea la salida ’and’
.IP \\(bu
512 (0x200) si se desea la salida ’or’
.IP \\(bu
1024 (0x400) si se desea la salida ’xor’ (or exclusivo)”””;
licencia ”GPL”; // indica GPL v2 o posterior
;;
FUNCTION(_) {
    int i, a=1, o=0, x=0;
    for(i=0; i < (personality & 0xff); i++) {
        if(in(i)) { o = 1; x = !x; }
        else { a = 0; }
    }
    if(personality & 0x100) and = a;
    if(personality & 0x200) or = o;
    if(personality & 0x400) xor = x;
}
```

A typical load line for this component might be

```
loadrt logic count=3 personality=0x102,0x305,0x503
```

which creates the following pins:

- A 2-input AND gate: `logic.0.and`, `logic.0.in-00`, `logic.0.in-01`
- 5-input AND and OR gates: `logic.1.and`, `logic.1.or`, `logic.1.in-00`, `logic.1.in-01`, `logic.1.in-02`, `logic.1.in-03`, `logic.1.in-04`,
- 3-input AND and XOR gates: `logic.2.and`, `logic.2.xor`, `logic.2.in-00`, `logic.2.in-01`, `logic.2.in-02`

5.9.13.8. General Functions

Este ejemplo muestra cómo llamar a funciones desde la función principal. También muestra cómo pasar la referencia de los pines HAL a esas funciones.

```
component example;
pin in s32 in;
pin out bit out1;
pin out bit out2;

option period no;
function _;
license ”GPL”;
;;

// general pin set true function
void set(hal_bit_t *p){
    *p = 1;
}

// general pin set false function
void unset(hal_bit_t *p){
    *p = 0;
```

```
}

//main function
FUNCTION(_) {
    if (in < 0){
        set(&out1);
        unset(&out2);
    }else if (in >0){
        unset(&out2);
        set(&out2);
    }else{
        unset(&out1);
        unset(&out2);
    }
}
```

Este componente utiliza dos funciones generales para manipular un pin bit HAL al que se hace referencia.

5.9.14. Command Line Usage

The halcompile man page gives details for invoking `halcompile`.

```
$ man halcompile
```

A brief summary of halcompile usage is given by:

```
$ halcompile --help
```

5.10. HALTCL Files

halcmd excels in specifying components and connections but these scripts offer no computational capabilities. As a result, INI files are limited in the clarity and brevity that is possible with higher level languages.

The haltcl facility provides a means to use Tcl scripting and its features for computation, looping, branching, procedures, etc. in INI files. To use this functionality, you use the Tcl language and the extension .tcl for HAL files.

The .tcl extension is understood by the main script (`linuxcnc`) that processes INI files. Haltcl files are identified in the the HAL section of INI files (just like HAL files).

Ejemplo

```
[HAL]
HALFILE = conventional_file.hal
HALFILE = tcl_based_file.tcl
```

With appropriate care, HAL and Tcl files can be intermixed.

5.10.1. Compatibility

The halcmd language used in HAL files has a simple syntax that is actually a subset of the more powerful general-purpose Tcl scripting language.

5.10.2. Haltcl Commands

Haltcl files use the Tcl scripting language augmented with the specific commands of the LinuxCNC hardware abstraction layer (HAL). The HAL-specific commands are:

```
addf, alias,
delf, delsig,
getp, gets
ptype,
stype,
help,
linkpp, linkps, linksp, list, loadrt, loadusr, lock,
net, newsig,
save, setp, sets, show, source, start, status, stop,
unalias, unlinkp, unload, unloadrt, unloadusr, unlock,
waitusr
```

Hay dos casos especiales para los comandos *gets* y *list* debido a conflictos con los comandos internos de Tcl. Para haltcl, estos comandos deben ir precedidos de la palabra clave *hal*:

```
halcmd   haltcl
------   ------
gets     hal gets
list     hal list
```

5.10.3. Haltcl INI-file variables

Las variables INI son accesibles tanto por `halcmd` como por `haltcl` pero con sintaxis diferente. Los archivos INI de LinuxCNC usan los especificadores SECTION y ITEM para identificar elementos de configuración:

```
[SECTION_A]
ITEM1 = value_1
ITEM2 = value_2
...
[SECTION_B]
...
```

Los valores de los archivos INI son accesibles por sustituciones de texto en archivos HAL utilizando la forma:

```
[SECTION]ITEM
```

Los mismos valores de archivo INI son accesibles en archivos .tcl usando la forma de una variable global de matriz Tcl:

```
$::SECTION(ITEM)
```

For example, an INI file item like:

```
[JOINT_0]
MAX_VELOCITY = 4
```

se expresa como `[JOINT_0]MAX_VELOCITY` en archivos HAL para halcmd
y como ` $::JOINT_0(MAX_VELOCITY)` en archivos Tcl para haltcl.

Because INI files can repeat the same ITEM in the same SECTION multiple times, `$::SECTION(ITEM)` is actually a Tcl list of each individual value.

When there is just one value and it is a simple value (all values that are just letters and numbers without whitespace are in this group), then it is possible to treat `$::SECTION(ITEM)` as though it is not a list.

When the value could contain special characters (quote characters, curly-brace characters, embedded whitespace, and other characters that have special meaning in Tcl) then it is necessary to distinguish between the list of values and the initial (and possibly only) value in the list.

In Tcl, this is written `[lindex $::SECTION(ITEM) 0]`.

For example: given the following INI values

```
[HOSTMOT2]
DRIVER=hm2_eth
IPADDR=”10.10.10.10”
BOARD=7i92
CONFIG=”num_encoders=0 num_pwmgens=0 num_stepgens=6”
```

And this loadrt command:

```
loadrt $::HOSTMOT2(DRIVER) board_ip=$::HOSTMOT2(IPADDR) config=$::HOSTMOT2(CONFIG)
```

Here is the actual command that is run:

```
loadrt hm2_eth board_ip={”10.10.10.10”} config={”num_encoders=0 num_pwmgens=0 num_stepgens ↩
    =6”}
```

This fails because loadrt does not recognize the braces.

So to get the values just as entered in the INI file, re-write the loadrt line like this:

```
loadrt $::HOSTMOT2(DRIVER) board_ip=[lindex $::HOSTMOT2(IPADDR) 0] config=[lindex  ↩
    $::HOSTMOT2(CONFIG) 0]
```

5.10.4. Converting HAL files to Tcl files

Los archivos .hal existentes se pueden convertir en archivos .tcl mediante la edición manual para adaptar las diferencias mencionadas anteriormente. El proceso puede ser automatizado con scripts que los convierten utilizando estas sustituciones.

```
[SECTION]ITEM ---> $::SECTION(ITEM)
gets          ---> hal gets
list          ---> hal list
```

5.10.5. Haltcl Notes

In haltcl, the value argument for the *sets* and *setp* commands is implicitly treated as an expression in the Tcl language.

Ejemplo

```
# set gain to convert deg/sec to units/min for JOINT_0 radius
setp scale.0.gain 6.28/360.0*$::JOINT_0(radius)*60.0
```

Whitespace in the bare expression is not allowed, use quotes for that:

```
setp scale.0.gain ”6.28 / 360.0 * $::JOINT_0(radius) * 60.0”
```

In other contexts, such as *loadrt*, you must explicitly use the Tcl expr command ([expr {}]) for computational expressions.

Ejemplo

```
loadrt motion base_period=[expr {500000000/$::TRAJ(MAX_PULSE_RATE)}]
```

5.10.6. Haltcl Examples

Consider the topic of *stepgen headroom*. Software `stepgen` runs best with an acceleration constraint that is ”a bit higher” than the one used by the motion planner. So, when using `halcmd` files, we force INI files to have a manually calculated value.

```
[JOINT_0]
MAXACCEL = 10.0
STEPGEN_MAXACCEL = 10.5
```

Con `haltcl`, puede usar los comandos Tcl para hacer el cálculo y eliminar el elemento `STEPGEN_MAXACCEL` del archivo INI por completo:

```
setp stepgen.0.maxaccel $::JOINT_0(MAXACCEL)*1.05
```

Another `haltcl` feature is looping and testing. For example, many simulator configurations use ”core_sim.hal” or ”core_sim9.hal” HAL files. These differ because of the requirement to connect more or fewer axes. The following haltcl code would work for any combination of axes in a trivkins machine.

```
# Create position, velocity and acceleration signals for each axis
set ddt 0
for {set jnum 0} {$jnum < $::KINS(JOINTS)} {incr jnum} {
  # 'list pin' returns an empty list if the pin doesn't exist
  if {[hal list pin joint.${jnum}.motor-pos-cmd] == {}} {
    continue
  }
  net ${jnum}pos joint.${jnum}.motor-pos-cmd => joint.$axno.motor-pos-fb \
                                            => ddt.$ddt.in
  net ${axis}vel <= ddt.$ddt.out
  incr ddt
  net ${axis}vel => ddt.$ddt.in
  net ${axis}acc <= ddt.$ddt.out
  incr ddt
}
puts [show sig *vel]
puts [show sig *acc]
```

5.10.7. Haltcl Interactive

The `halrun` command recognizes haltcl files. With the -T option, haltcl can be run interaactively as a Tcl interpreter. This capability is useful for testing and for standalone HAL applications.

Ejemplo

```
$ halrun -T haltclfile.tcl
```

5.10.8. Haltcl Distribution Examples (sim)

The configs/sim/axis/simtcl directory includes an INI file that uses a .tcl file to demonstrate a haltcl configuration in conjunction with the usage of twopass processing. The example shows the use of Tcl procedures, looping, the use of comments and output to the terminal.

5.11. HAL User Interface

5.11.1. Introducción

Halui is a HAL based user interface for LinuxCNC, it connects HAL pins to NML commands. Most of the functionality (buttons, indicators etc.) that is provided by a traditional GUI (AXIS, GMOCCAPY, QtDragon, etc.), is provided by HAL pins in Halui.

La forma más fácil para agregar halui es agregando lo siguiente a la sección [HAL] del archivo INI:

```
[HAL]
HALUI = halui
```

An alternate way to invoke it (specially if you generate the configuration with StepConf) is to include the following in your `custom.hal` file.
Make sure you use the correct path to your INI file.

```
loadusr halui -ini /path/to/inifile.ini
```

5.11.2. MDI

Sometimes the user wants to add more complicated tasks to be performed by the activation of a HAL pin. This is possible by adding MDI commands to the INI file in the [HALUI] section. Example:

```
[HALUI]
MDI_COMMAND = G0 X0
MDI_COMMAND = G0 G53 Z0
MDI_COMMAND = G28
MDI_COMMAND = o<mysub>call
...
```

When halui starts it will read the `MDI_COMMAND` fields in the INI and export pins from 00 to the number of `MDI_COMMAND` 's found in the INI, up to a maximum of 64 commands. These pins can be connected like any HAL pins. A common method is to use buttons provided by virtual control panels like shown in the example Example for MDI_COMMAND connections.

Ejemplo 5.1 Example for MDI_COMMAND connections

HAL file

```
net quill-up      halui.mdi-command-00 <= pyvcp.quillup
net reference-pos halui.mdi-command-01 <= pyvcp.referencepos
net call-mysub    halui.mdi-command-02 <= pyvcp.callmysub
```

Nets connecting the halui.mdi-command-NN pins provided by halui.

```
$ halcmd show pin halui.mdi
Component Pins:
Owner  Type  Dir Value  Name
   10  bit   IN  FALSE  halui.mdi-command-00 <== quill-up
   10  bit   IN  FALSE  halui.mdi-command-01 <== reference-pos
   10  bit   IN  FALSE  halui.mdi-command-02 <== call-mysub
    ...
```

When a halui MDI pin is set (pulsed) true, halui will send the MDI command defined in the INI. This will not always succeed depending on the current operating mode (e.g., while in AUTO halui can't successfully send MDI commands).

5.11.3. Example Configuration

An example sim config (`configs/sim/axis/halui_pyvcp/halui.ini`) is included in the distribution.

5.11.4. Halui Pin Reference

All halui pins are also documented in the halui man page:

```
$ man halui
```

Or see http://linuxcnc.org/docs/devel/html/man/man1/halui.1.html

5.11.4.1. Abortar

- *halui.abort* (bit, in) - pin to send an abort message (clears out most errors)

5.11.4.2. E-Stop

- *halui.estop.activate* (bit, in) - pin for requesting E-Stop
- *halui.estop.is-activated* (bit, out) - indicates E-stop reset
- *halui.estop.reset* (bit, in) - pin for requesting E-Stop reset

5.11.4.3. Porcentaje de alimentación

- *halui.feed-override.count-enable* (bit, in) - must be true for *counts* or *direct-value* to work.
- *halui.feed-override.counts* (s32, in) - counts * scale = FO percentage. Can be used with an encoder or *direct-value*.
- *halui.feed-override.decrease* (bit, in) - pin for decreasing the FO (-=scale)
- *halui.feed-override.increase* (bit, in) - pin for increasing the FO (+=scale)
- *halui.feed-override.reset* (bit, in) - pin for resetting the FO (scale=1.0)
- *halui.feed-override.direct-value* (bit, in) - false when using encoder to change counts, true when setting counts directly.
- *halui.feed-override.scale* (float, in) - pin for setting the scale for increase and decrease of *feed-override*.
- *halui.feed-override.value* (float, out) - current FO value

5.11.4.4. Mist

- *halui.mist.is-on* (bit, out) - indicates mist is on
- *halui.mist.off* (bit, in) - pin for requesting mist off
- *halui.mist.on* (bit, in) - pin for requesting mist on

5.11.4.5. Flood

- *halui.flood.is-on* (bit, out) - indicates flood is on
- *halui.flood.off* (bit, in) - pin for requesting flood off
- *halui.flood.on* (bit, in) - pin for requesting flood on

5.11.4.6. Homing

- *halui.home-all* (bit, in) - pin for requesting all axis to home. This pin will only be there if HOME_SEQUENCE is set in the INI file.

5.11.4.7. Machine

- *halui.machine.units-per-mm* (float out) - pin for machine units-per-mm (inch:1/25.4, mm:1) according to inifile setting: [TRAJ]LINEAR_UNITS
- *halui.machine.is-on* (bit, out) - indicates machine on
- *halui.machine.off* (bit, in) - pin for requesting machine off
- *halui.machine.on* (bit, in) - pin for requesting machine on

5.11.4.8. Velocidad máxima

The maximum linear velocity can be adjusted from 0 to the MAX_VELOCITY that is set in the [TRAJ] section of the INI file.

- *halui.max-velocity.count-enable* (bit, in) - must be true for *counts* or *direct-value* to work.
- *halui.max-velocity.counts* (s32, in) - counts * scale = MV percentage. Can be used with an encoder or *direct-value*.
- *halui.max-velocity.direct-value* (bit, in) - false when using encoder to change counts, true when setting counts directly.
- *halui.max-velocity.decrease* (bit, in) - pin for decreasing max velocity
- *halui.max-velocity.increase* (bit, in) - pin for increasing max velocity
- *halui.max-velocity.scale* (float, in) - the amount applied to the current maximum velocity with each transition from off to on of the increase or decrease pin in machine units per second.
- *halui.max-velocity.value* (float, out) - is the maximum linear velocity in machine units per second.

5.11.4.9. MDI

- *halui.mdi-command-*<nn> (bit, in) - halui will try to send the MDI command defined in the INI. <*nn*> is a two digit number starting at 00.
  If the command succeeds then it will place LinuxCNC in the MDI mode and then back to Manual mode.
  If no [HALUI]MDI_COMMAND variables are set in the ini file, no halui.mdi-command-<nn> pins will be exported by halui.
- *halui.halui-mdi-is-running* (bit, out) - execution status of MDI commands sent by halui. The status is active even during mode switching. If no [HALUI]MDI_COMMAND variables are set in the ini file, this pins will not be exported by halui.

5.11.4.10. Joint

N = joint number (0 … num_joints-1)
Example:

- *halui.joint.*N.*select* (bit in) - pin para seleccionar articulación *N*
- *halui.joint.*N.*is-selected* (bit out) - pin de estado que la articulación *N* esta seleccionada
- *halui.joint.*N.*has-fault* (bit out) - pin de estado indicando que la articulación *N* tiene un fallo
- *halui.joint.*N.*home* (bit in) - pin for homing joint *N*
- *halui.joint.*N.*is-homed* (bit out) - pin de estado indicando que la articulación *N* esta en casa
- *halui.joint.*N.*on-hard-max-limit* (bit out) - pin de estatus indicando que la articulación *N* esta en el límite de hardware positivo
- *halui.joint.*N.*on-hard-min-limit* (bit out) - pin de estado indicando que la articulación *N* esta en el límite de hardware negativo
- *halui.joint.*N.*on-soft-max-limit* (bit out) - pin de estado indicando que la articulación *N* esta en el límite de software positivo
- *halui.joint.*N.*on-soft-min-limit* (bit out) - pin de estado indicando que la articulación *N* esta en el límite de software negativo
- *halui.joint.*N.*override-limits* (bit out) - pin de estado indicando que los límites de la articulación *N* están anulados temporalmente
- *halui.joint.*N.*unhome* (bit in) - pin for unhoming joint *N*
- *halui.joint.selected* (u32 out) - selected joint number (0 … num_joints-1)
- *halui.joint.selected.has-fault* (bit out) - status pin selected joint is faulted
- *halui.joint.selected.home* (bit in) - pin for homing the selected joint
- *halui.joint.selected.is-homed* (bit out) - pin de estado indicando que la articulación seleccionada esta en casa
- *halui.joint.selected.on-hard-max-limit* (bit out) - status pin telling that the selected joint is on the positive hardware limit
- *halui.joint.selected.on-hard-min-limit* (bit out) - status pin telling that the selected joint is on the negative hardware limit
- *halui.joint.selected.on-soft-max-limit* (bit out) - status pin telling that the selected joint is on the positive software limit
- *halui.joint.selected.on-soft-min-limit* (bit out) - status pin telling that the selected joint is on the negative software limit
- *halui.joint.selected.override-limits* (bit out) - status pin telling that the selected joint’s limits are temporarily overridden
- *halui.joint.selected.unhome* (bit in) - pin for unhoming the selected joint

5.11.4.11. Joint Jogging

N = joint number (0 ... num_joints-1)

- *halui.joint.jog-deadband* (float in) - pin for setting jog analog deadband (jog analog inputs smaller/slower than this - in absolute value - are ignored)
- *halui.joint.jog-speed* (float in) - pin for setting jog speed for plus/minus jogging.
- *halui.joint.*N.*analog* (float in) - pin for jogging the joint *N* using a float value (e.g. joy-stick). The value, typically set between 0.0 and ±1.0, is used as a jog-speed multiplier.
- *halui.joint.*N.*increment* (float in) - pin para asignar el incremento de trote para la articulación *N* cuando se usa +/-
- *halui.joint.*N.*increment-minus* (bit in) - un flanco ascendente hará que la articulación *N* trote en la dirección negativa por la cantidad incrementada
- *halui.joint.*N.*increment-plus* (bit in) - un flanco ascendente hará que la articulación *N* trote en la dirección positiva por la cantidad incrementada
- *halui.joint.*N.*minus* (bit in) - pin para trotar la articulación *N* en dirección negativa a la velocidad halui.joint.jog-speed
- *halui.joint.*N.*plus* (bit in) - pin para trotar la articulación *N* en dirección positiva a la velocidad halui.joint.jog-speed
- *halui.joint.selected.increment* (float in) - pin para asignar el incremento de trote para la articulación seleccionada cuando se usa +/-
- *halui.joint.selected.increment-minus* (bit in) - a rising edge will will make the selected joint jog in the negative direction by the increment amount
- *halui.joint.selected.increment-plus* (bit in) - a rising edge will will make the selected joint jog in the positive direction by the increment amount
- *halui.joint.selected.minus* (bit in) - pin for jogging the selected joint in negative direction at the halui.joint.jog-speed velocity
- *halui.joint.selected.plus* (bit in) - pin for jogging the selected joint in positive direction at the halui.joint.jog-speed velocity

5.11.4.12. Eje

L = axis letter (xyzabcuvw)

- *halui.axis.*L.*select* (bit) - pin for selecting axis by letter
- *halui.axis.*L.*is-selected* (bit out) - pin de estado que indica que el eje *L* esta seleccionado
- *halui.axis.*L.*pos-commanded* (float out) - Posición de eje comandada en coordenadas de máquina
- *halui.axis.*L.*pos-feedback* float out) - Posición del eje de retroalimentación en coordenadas de máquina
- *halui.axis.*L.*pos-relative* (float out) - Posición del eje de retroalimentación en coordenadas relativas

5.11.4.13. Axis Jogging

L = axis letter (xyzabcuvw)

- *halui.axis.jog-deadband* (float in) - pin for setting jog analog deadband (jog analog inputs smaller/slower than this (in absolute value) are ignored)
- *halui.axis.jog-speed* (float in) - pin para asignar la velocidad de trote para +/-.
- *halui.axis.*L.*analog* (float in) - pin for jogging the axis *L* using an float value (e.g. joystick). The value, typically set between 0.0 and ±1.0, is used as a jog-speed multiplier.
- *halui.axis.*L.*increment* (float in) - pin para asignar el incremento de trote para el eje *L* cuando se usa +/-
- *halui.axis.*L.*increment-minus* (bit in) - in flanco ascendente hará que el eje *L* trote en la dirección negativa por la cantidad incrementada
- *halui.axis.*L.*increment-plus* (bit in) - un flanco ascendente hará trotar al eje *L* in la dirección positiva por la cantidad incrementada
- *halui.axis.*L.*minus* (bit in) - pin para trotar el eje *L* en dirección negativa a la velocidad halui.axis.jog-speed
- *halui.axis.*L.*plus* (bit in) - pin para trotar el eje *L* en dirección positiva a la velocidad halui.axis.jog-speed
- *halui.axis.selected* (u32 out) - selected axis (by index: 0:x 1:y 2:z 3:a 4:b 5:cr 6:u 7:v 8:w)
- *halui.axis.selected.increment* (float in) - pin para asignar el incremento de trote para el eje seleccionado cuando se usa +/-
- *halui.axis.selected.increment-minus* (bit in) - un flanco ascendente hará que el eje seleccionado trote en la dirección negativa por la cantidad incrementada
- *halui.axis.selected.increment-plus* (bit in) - un flanco ascendente hará que el eje seleccionado trote en la dirección positiva por la cantidad incrementada
- *halui.axis.selected.minus* (bit in) - pin para trotar el eje seleccionado en dirección negativa a la velocidad de halui.axis.jog-speed
- *halui.axis.selected.plus* (pin in) - para trotar el eje seleccionado en dirección positiva a la velocidad de halui.axis.jog-speed

5.11.4.14. Modo

- *halui.mode.auto* (bit, in) - pin for requesting auto mode
- *halui.mode.is-auto* (bit, out) - indicates auto mode is on
- *halui.mode.is-joint* (bit, out) - indicates joint by joint jog mode is on
- *halui.mode.is-manual* (bit, out) - indicates manual mode is on
- *halui.mode.is-mdi* (bit, out) - indicates MDI mode is on
- *halui.mode.is-teleop* (bit, out) - indicates coordinated jog mode is on
- *halui.mode.joint* (bit, in) - pin for requesting joint by joint jog mode
- *halui.mode.manual* (bit, in) - pin for requesting manual mode
- *halui.mode.mdi* (bit, in) - pin for requesting MDI mode
- *halui.mode.teleop* (bit, in) - pin for requesting coordinated jog mode

5.11.4.15. Program

- *halui.program.block-delete.is-on* (bit, out) - status pin telling that block delete is on
- *halui.program.block-delete.off* (bit, in) - pin for requesting that block delete is off
- *halui.program.block-delete.on* (bit, in) - pin for requesting that block delete is on
- *halui.program.is-idle* (bit, out) - status pin telling that no program is running
- *halui.program.is-paused* (bit, out) - status pin telling that a program is paused
- *halui.program.is-running* (bit, out) - status pin telling that a program is running
- *halui.program.optional-stop.is-on* (bit, out) - status pin telling that the optional stop is on
- *halui.program.optional-stop.off* (bit, in) - pin requesting that the optional stop is off
- *halui.program.optional-stop.on* (bit, in) - pin requesting that the optional stop is on
- *halui.program.pause* (bit, in) - pin for pausing a program
- *halui.program.resume* (bit, in) - pin for resuming a paused program
- *halui.program.run* (bit, in) - pin for running a program
- *halui.program.step* (bit, in) - pin for stepping in a program
- *halui.program.stop* (bit, in) - pin for stopping a program

5.11.4.16. Rapid Override

- *halui.rapid-override.count-enable* (bit in (default: TRUE)) - When TRUE, modify Rapid Override when counts changes.
- *halui.rapid-override.counts* (s32 in) - counts X scale = Rapid Override percentage. Can be used with an encoder or *direct-value*.
- *halui.rapid-override.decrease* (bit in) - pin for decreasing the Rapid Override (-=scale)
- *halui.rapid-override.direct-value* (bit in) - pin to enable direct value Rapid Override input
- *halui.rapid-override.increase* (bit in) - pin for increasing the Rapid Override (+=scale)
- *halui.rapid-override.scale* (float in) - pin for setting the scale on changing the Rapid Override
- *halui.rapid-override.value* (float out) - current Rapid Override value
- *halui.rapid-override.reset* (bit, in) - pin for resetting the Rapid Override value (scale=1.0)

5.11.4.17. Spindle Override

- *halui.spindle.*N.*override.count-enable* (bit, in) - must be true for *counts* or *direct-value* to work.
- *halui.spindle.*N.*override.counts* (s32, in) - counts * scale = SO percentage. Can be used with an encoder or *direct-value*.
- *halui.spindle.*N.*override.decrease* (bit, in) - pin for decreasing the SO (-=scale)
- *halui.spindle.*N.*override.direct-value* (bit, in) - false when using encoder to change counts, true when setting counts directly.
- *halui.spindle.*N.*override.increase* (bit, in) - pin for increasing the SO (+=scale)
- *halui.spindle.*N.*override.scale* (float, in) - pin for setting the scale on changing the SO
- *halui.spindle.*N.*override.value* (float, out) - current SO value
- *halui.spindle.*N.*override.reset* (bit, in) - pin for resetting the SO value (scale=1.0)

5.11.4.18. Husillo

- *halui.spindle.*N.*brake-is-on* (bit, out) - indicates brake is on
- *halui.spindle.*N.*brake-off* (bit, in) - pin for deactivating spindle/brake
- *halui.spindle.*N.*brake-on* (bit, in) - pin for activating spindle-brake
- *halui.spindle.*N.*decrease* (bit, in) - decreases spindle speed
- *halui.spindle.*N.*forward* (bit, in) - enciende el husillo con movimiento CW
- *halui.spindle.*N.*increase* (bit, in)- aumenta la velocidad del husillo
- *halui.spindle.*N.*is-on* (bit, out) - indica que el husillo esta encendido (cualquier dirección)
- *halui.spindle.*N.*reverse* (bit, in)- starts the spindle with a CCW motion
- *halui.spindle.*N.*runs-backward* (bit, out) - indica que el husillo esta encendido, y en reversa
- *halui.spindle.*N.*runs-forward* (bit, out) - indica que el husillo esta encendido, y en sentido normal
- *halui.spindle.*N.*start* (bit, in) - inicia el husillo
- *halui.spindle.*N.*stop* (bit, in) - detiene el husillo

5.11.4.19. Tool

- *halui.tool.length-offset.a* (float out) - current applied tool length offset for the A axis
- *halui.tool.length-offset.b* (float out) - current applied tool length offset for the B axis
- *halui.tool.length-offset.c* (float out) - current applied tool length offset for the C axis
- *halui.tool.length-offset.u* (float out) - current applied tool length offset for the U axis
- *halui.tool.length-offset.v* (float out) - current applied tool length offset for the V axis
- *halui.tool.length-offset.w* (float out) - current applied tool length offset for the W axis
- *halui.tool.length-offset.x* (float out) - current applied tool length offset for the X axis
- *halui.tool.length-offset.y* (float out) - current applied tool length offset for the Y axis
- *halui.tool.length-offset.z* (float out) - current applied tool length offset for the Z axis
- *halui.tool.diameter* (float out) - Current tool diameter, or 0 if no tool is loaded.
- *halui.tool.number* (u32, out) - indica la herramienta seleccionada actual

5.12. Halui Examples

For any Halui examples to work you need to add the following line to the [HAL] section of the INI file.

```
HALUI = halui
```

5.12.1. Remote Start

To connect a remote program start button to LinuxCNC you use the `halui.program.run` pin and the `halui.mode.auto` pin. You have to ensure that it is OK to run first by using the `halui.mode.is-auto` pin. You do this with an `and2` component. The following figure shows how this is done. When the Remote Run Button is pressed it is connected to both `halui.mode.auto` and `and2.0.in0`. If it is OK for auto mode the pin `halui.mode.is-auto` will be on. If both the inputs to the `and2.0` component are on the `and2.0.out` will be on and this will start the program.

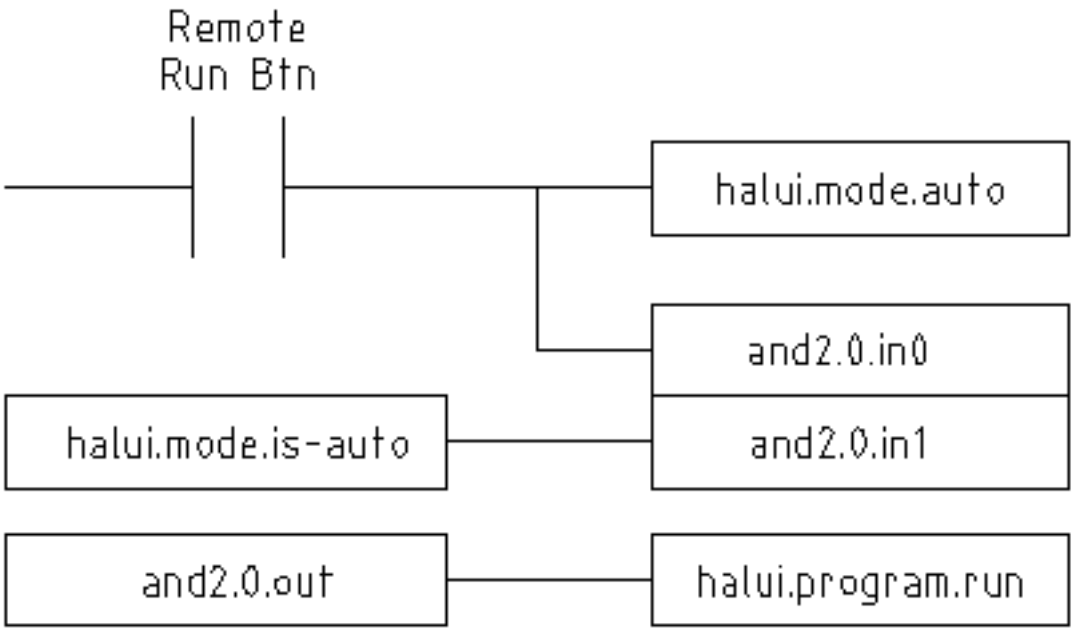


Figura 5.20: Remote Start Example

The hal commands needed to accomplish the above are:

```
net program-start-btn halui.mode.auto and2.0.in0 <= <your input pin>
net program-run-ok and2.0.in1 <= halui.mode.is-auto
net remote-program-run halui.program.run <= and2.0.out
```

Notice on line one that there are two reader pins, this can also be split up to two lines like this:

```
net program-start-btn halui.mode.auto <= <your input pin>
net program-start-btn and2.0.in0
```

5.12.2. Pause & Resume

This example was developed to allow LinuxCNC to move a rotary axis on a signal from an external machine. The coordination between the two systems will be provided by two Halui components:

- `halui.program.is-paused`
- `halui.program.resume`

In your customized HAL file, add the following two lines that will be connected to your I/O to turn on the program pause or to resume when the external system wants LinuxCNC to continue.

```
net ispaused halui.program.is paused => ”your output pin”
net resume halui.program.resume <= ”your input pin”
```

Your input and output pins are connected to the pins wired to the other controller. They may be parallel port pins or any other I/O pins that you have access to.

This system works in the following way. When an M0 is encountered in your G-code, the `halui.program.is-p` signal goes true. This turns on your output pin so that the external controller knows that LinuxCNC is paused.

To resume the LinuxCNC G-code program, when the external controller is ready it will make its output true. This will signal LinuxCNC that it should resume executing G-code.

Difficulties in timing

- The ”resume” input return signal should not be longer than the time required to get the G-code running again.
- The ”is-paused” output should no longer be active by the time the ”resume” signal ends.

These timing problems could be avoided by using ClassicLadder to activate the ”is-paused” output via a monostable timer to deliver one narrow output pulse. The ”resume” pulse could also be received via a monostable timer.

5.13. Creating Non-realtime Python Components

This section explains principles behind the implementation of HAL components with the Python programming language.

5.13.1. Basic usage example

A non-realtime component begins by creating its pins and parameters, then enters a loop which will periodically drive all the outputs from the inputs. The following component copies the value seen on its input pin (*passthrough.in*) to its output pin (*passthrough.out*) approximately once per second.

```
#!/usr/bin/env python3
import hal, time
h = hal.component(”passthrough”)
h.newpin(”in”, hal.HAL_FLOAT, hal.HAL_IN)
h.newpin(”out”, hal.HAL_FLOAT, hal.HAL_OUT)
h.ready()
try:
    while 1:
        time.sleep(1)
        h[’out’] = h[’in’]
except KeyboardInterrupt:
    raise SystemExit
```

Copy the above listing into a file named ”passthrough”, make it executable (*chmod +x*), and place it on your *$PATH*. Then try it out:

Screen copy with details on the execution of the newly created passthrough HAL module.

```
$ halrun

halcmd: loadusr passthrough
```

```
halcmd: show pin

    Component Pins:
    Owner Type  Dir     Value  Name
     03   float IN          0  passthrough.in
     03   float OUT         0  passthrough.out

halcmd: setp passthrough.in 3.14

halcmd: show pin

    Component Pins:
    Owner Type  Dir     Value  Name
     03   float IN       3.14  passthrough.in
     03   float OUT      3.14  passthrough.out
```

5.13.2. Non-realtime components and delays

If you typed ”show pin” quickly, you may see that *passthrough.out* still had its old value of 0. This is because of the call to *time.sleep(1)*, which makes the assignment to the output pin occur at most once per second. Because this is a non-realtime component, the actual delay between assignments can be much longer if the memory used by the passthrough component is swapped to disk, as the assignment could be delayed until that memory is swapped back in.

Thus, non-realtime components are suitable for user-interactive elements such as control panels (delays in the range of milliseconds are not noticed, and longer delays are acceptable), but not for sending step pulses to a stepper driver board (delays must always be in the range of microseconds, no matter what).

5.13.3. Create pins and parameters

```
h = hal.component(”passthrough”)
```

The component itself is created by a call to the constructor *hal.component*. The arguments are the HAL component name and (optionally) the prefix used for pin and parameter names. If the prefix is not specified, the component name is used.

```
h.newpin(”in”, hal.HAL_FLOAT, hal.HAL_IN)
```

Then pins are created by calls to methods on the component object. The arguments are: pin name suffix, pin type, and pin direction. For parameters, the arguments are: parameter name suffix, parameter type, and parameter direction.

Cuadro 5.31: HAL Option Names

Tipos de pines y parámetros:	HAL_BIT	HAL_FLOAT	HAL_S32	HAL_U32
HAL_S64	HAL_U64	Direcciones de pin:	HAL_IN	HAL_OUT
HAL_IO				Direcciones de parámetro:
HAL_RO	HAL_RW			

The full pin or parameter name is formed by joining the prefix and the suffix with a ”.”, so in the example the pin created is called *passthrough.in*.

```
h.ready()
```

Once all the pins and parameters have been created, call the *.ready()* method.

5.13.3.1. Changing the prefix

The prefix can be changed by calling the *.setprefix()* method. The current prefix can be retrieved by calling the *.getprefix()* method.

5.13.4. Reading and writing pins and parameters

For pins and parameters which are also proper Python identifiers, the value may be accessed or set using the attribute syntax:

```
h.out = h.in
```

For all pins, whether or not they are also proper Python identifiers, the value may be accessed or set using the subscript syntax:

```
h['out'] = h['in']
```

To see all pins with their values, getpins returns all values in a dictionary of that component.

```
h.getpins()
>>>{'in': 0.0, 'out': 0.0}
```

5.13.4.1. Driving output (HAL_OUT) pins

Periodically, usually in response to a timer, all HAL_OUT pins should be ”driven” by assigning them a new value. This should be done whether or not the value is different than the last one assigned. When a pin is connected to a signal, its old output value is not copied into the signal, so the proper value will only appear on the signal once the component assigns a new value.

5.13.4.2. Driving bidirectional (HAL_IO) pins

The above rule does not apply to bidirectional pins. Instead, a bidirectional pin should only be driven by the component when the component wishes to change the value. For instance, in the canonical encoder interface, the encoder component only sets the *index-enable* pin to **FALSE** (when an index pulse is seen and the old value is **TRUE**), but never sets it to **TRUE**. Repeatedly driving the pin **FALSE** might cause the other connected component to act as though another index pulse had been seen.

5.13.5. Exiting

A *halcmd unload* request for the component is delivered as a *KeyboardInterrupt* exception. When an unload request arrives, the process should either exit in a short time, or call the *.exit()* method on the component if substantial work (such as reading or writing files) must be done to complete the shutdown process.

5.13.6. Helpful Functions

See Python HAL Interface for an overview of available functions.

5.13.7. Constantes

Use these to specify details rather then the value they hold.

- HAL_BIT
- HAL_FLOAT
- HAL_S32
- HAL_U32
- HAL_S64
- HAL_U64
- HAL_IN
- HAL_OUT
- HAL_RO
- HAL_RW
- MSG_NONE
- MSG_ALL
- MSG_DBG
- MSG_ERR
- MSG_INFO
- MSG_WARN

5.13.8. System Information

Read these to acquire information about the realtime system.

- is_kernelspace
- is_rt
- is_sim
- is_userspace

5.14. Canonical Device Interfaces

5.14.1. Introducción

The following sections show the pins, parameters, and functions that are supplied by ”canonical devices”. All HAL device drivers should supply the same pins and parameters, and implement the same behavior.

Note that only the `_<io-type>_` and `_<specific-name>_` fields are defined for a canonical device. The `_<device-name>`, `_<device-num>_`, and `_<chan-num>_` fields are set based on the characteristics of the real device.

5.14.2. Digital Input

The canonical digital input (I/O type field: `digin`) is quite simple.

5.14.2.1. Pines

(bit) **in**:: State of the hardware input. (bit) **in-not**:: Inverted state of the input.

5.14.2.2. Parámetros

None

5.14.2.3. Funciones

(funct) **read**:: Read hardware and set `in` and `in-not` HAL pins.

5.14.3. Digital Output

The canonical digital output (I/O type field: `digout`) is also very simple.

5.14.3.1. Pines

(bit) **out**:: Value to be written (possibly inverted) to the hardware output.

5.14.3.2. Parámetros

(bit) **invert**:: If TRUE, **out** is inverted before writing to the hardware.

5.14.3.3. Funciones

(funct) **write**:: Read **out** and **invert**, and set hardware output accordingly.

5.14.4. Analog Input

The canonical analog input (I/O type: `adcin`). This is expected to be used for analog to digital converters, which convert e.g. voltage to a continuous range of values.

5.14.4.1. Pines

(float) **value**:: La lectura del hardware, escalada según los parámetros **scale** y **offset**.
value = ((lectura de entrada, en unidades dependientes del hardware) * **scale**) - **offset**

5.14.4.2. Parámetros

(float) **scale**:: The input voltage (or current) will be multiplied by **scale** before being output to **value**. (float) **offset**:: This will be subtracted from the hardware input voltage (or current) after the scale multiplier has been applied. (float) **bit_weight**:: El valor del bit menos significativo(LSB). Es efectivamente la granularidad de la lectura de entrada. (float) **hw_offset**:: The value present on the input when 0 Volts is applied to the input pin(s).

5.14.4.3. Funciones

(funct) **read**:: Lee los valores de este canal de entrada analógica. Se puede usar para leer un canal individual, o puede hacer que se lean todos los canales.

5.14.5. Analog Output

The canonical analog output (I/O Type: **adcout**). This is intended for any kind of hardware that can output a more-or-less continuous range of values. Examples are digital to analog converters or PWM generators.

5.14.5.1. Pines

(float) **value**:: The value to be written. The actual value output to the hardware will depend on the scale and offset parameters. (bit) **enable**:: If false, then output 0 to the hardware, regardless of the **value** pin.

5.14.5.2. Parámetros

(float) **offset**:: This will be added to the **value** before the hardware is updated. (float) **scale**:: This should be set so that an input of 1 on the **value** pin will cause the analog output pin to read 1 volt. (float) **high_limit** (optional):: When calculating the value to output to the hardware, if **value** + **offset** is greater than **high_limit**, then **high_limit** will be used instead. (float) **low_limit** (optional):: When calculating the value to output to the hardware, if **value** + **offset** is less than **low_limit**, then **low_limit** will be used instead. (float) **bit_weight** (optional):: El valor del bit menos significativo (LSB), en voltios (o mA, para salidas de corriente). (float) **hw_offset** (optional):: The actual voltage (or current) that will be output if 0 is written to the hardware.

5.14.5.3. Funciones

(funct) **write**:: This causes the calculated value to be output to the hardware. If enable is false, then the output will be 0, regardless of **value**, **scale**, and **offset**. The meaning of ”0” is dependent on the hardware. For example, a bipolar 12-bit A/D may need to write 0x1FF (mid scale) to the D/A get 0 volts from the hardware pin. If enable is true, read scale, offset and value and output to the adc (**scale** * **value**) + **offset**. If enable is false, then output 0.

5.15. HAL Tools

5.15.1. Halcmd

`halcmd` is a command line tool for manipulating the HAL. There is a rather complete man page for `halcmd`, which will be installed if you have installed LinuxCNC from either source or a package. The manpage provides usage info:

```
man halcmd
```

If you have compiled LinuxCNC for ”run-in-place”, you must source the rip-environment script to make the man page available:

```
cd toplevel_directory_for_rip_build
. scripts/rip-environment
man halcmd
```

The HAL Tutorial has a number of examples of halcmd usage, and is a good tutorial for `halcmd`.

5.15.2. Halmeter

Halmeter is a *voltmeter* for the HAL. It lets you look at a pin, signal, or parameter, and displays the current value of that item. It is pretty simple to use. Start it by typing `halmeter` in an X windows shell. Halmeter is a GUI application. It will pop up a small window, with two buttons labeled ”Select” and ”Exit”. Exit is easy - it shuts down the program. Select pops up a larger window, with three tabs. One tab lists all the pins currently defined in the HAL. The next lists all the signals, and the last tab lists all the parameters. Click on a tab, then click on a pin/signal/parameter. Then click on ”OK”. The lists will disappear, and the small window will display the name and value of the selected item. The display is updated approximately 10 times per second. If you click ”Accept” instead of ”OK”, the small window will display the name and value of the selected item, but the large window will remain on the screen. This is convenient if you want to look at a number of different items quickly.

You can have many halmeters running at the same time, if you want to monitor several items. If you want to launch a halmeter without tying up a shell window, type `halmeter &` to run it in the background. You can also make halmeter start displaying a specific item immediately, by adding `pin|sig|par[am] _<name>_` to the command line. It will display the pin, signal, or parameter *<name>* as soon as it starts - if there is no such item, it will simply start normally. And finally, if you specify an item to display, you can add `-s` before the `pin|sig|param` to tell halmeter to use a small window. The item name will be displayed in the title bar instead of under the value, and there will be no buttons. Useful when you want a lot of meters in a small amount of screen space.

Refer to Halmeter Tutorial section for more information.

`halmeter` can be loaded from a terminal or from AXIS. `halmeter` is faster than `halshow` at displaying values. `halmeter` has two windows, one to pick the pin, signal, or parameter to monitor and one that displays the value. Multiple ``halmeter``s can be open at the same time. If you use a script to open multiple ``halmeter``s you can set the position of each one with `-g X Y` relative to the upper left corner of your screen. For example:

```
loadusr halmeter pin hm2.0.stepgen.00.velocity-fb -g 0 500
```

See the man page for more options and the section Halmeter.

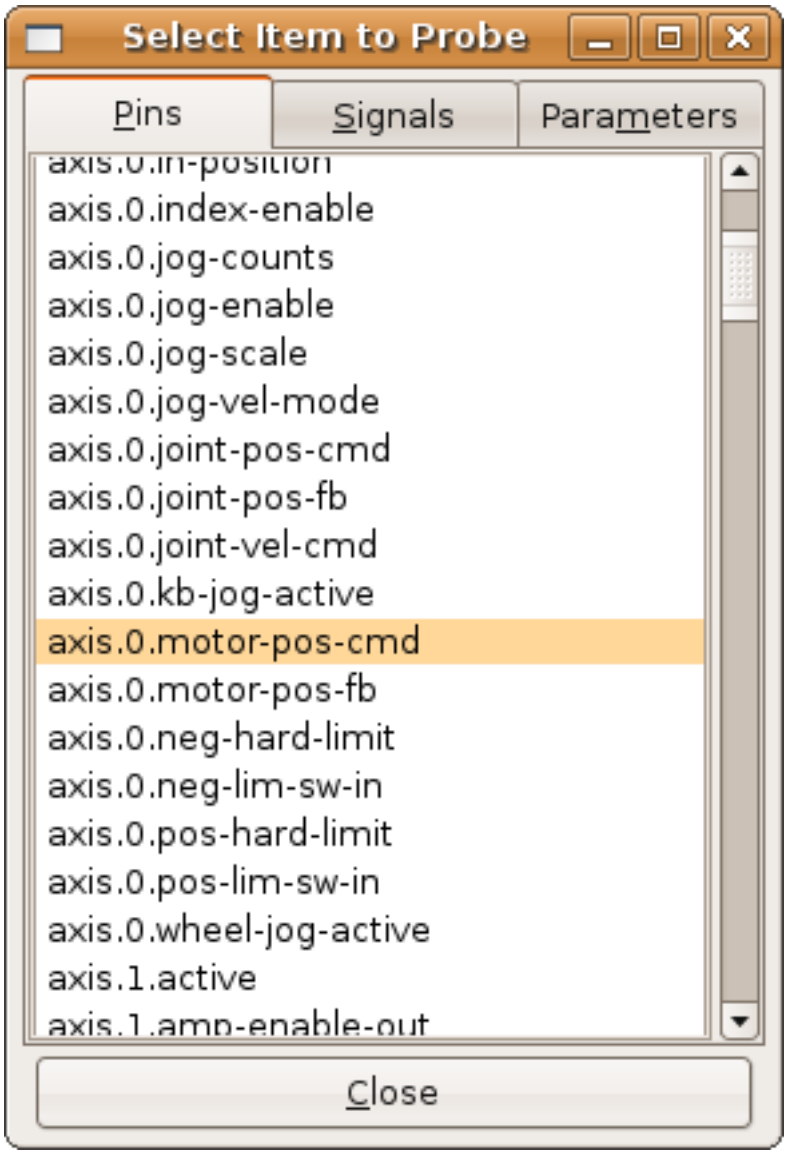


Figura 5.21: Halmeter selection window

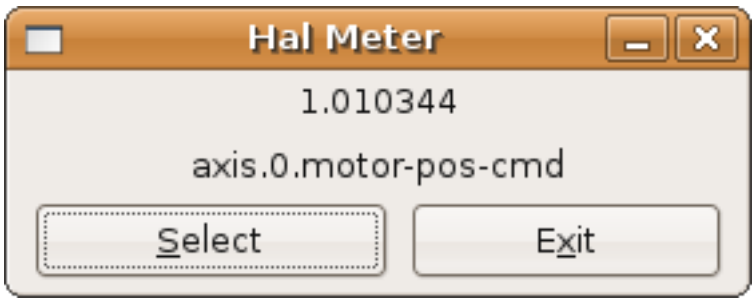


Figura 5.22: Halmeter watch window

5.15.3. Halshow

`halshow` (complete usage description) can be started from the command line to show details for selected components, pins, parameters, signals, functions, and threads of a running HAL. The WATCH tab provides a continuous display of selected pin, parameters, and signal items. The File menu provides buttons to save the watch items to a watch list and to load an existing watch list. The watch list items can also be loaded automatically on startup. For command line usage:

```
halshow --help
Usage:
  halshow [Options] [watchfile]
  Options:
           --help    (this help)
           --fformat format_string_for_float
           --iformat format_string_for_int

Notes:
  Create watchfile in halshow using: 'File/Save Watch List'.
  LinuxCNC must be running for standalone usage.
```

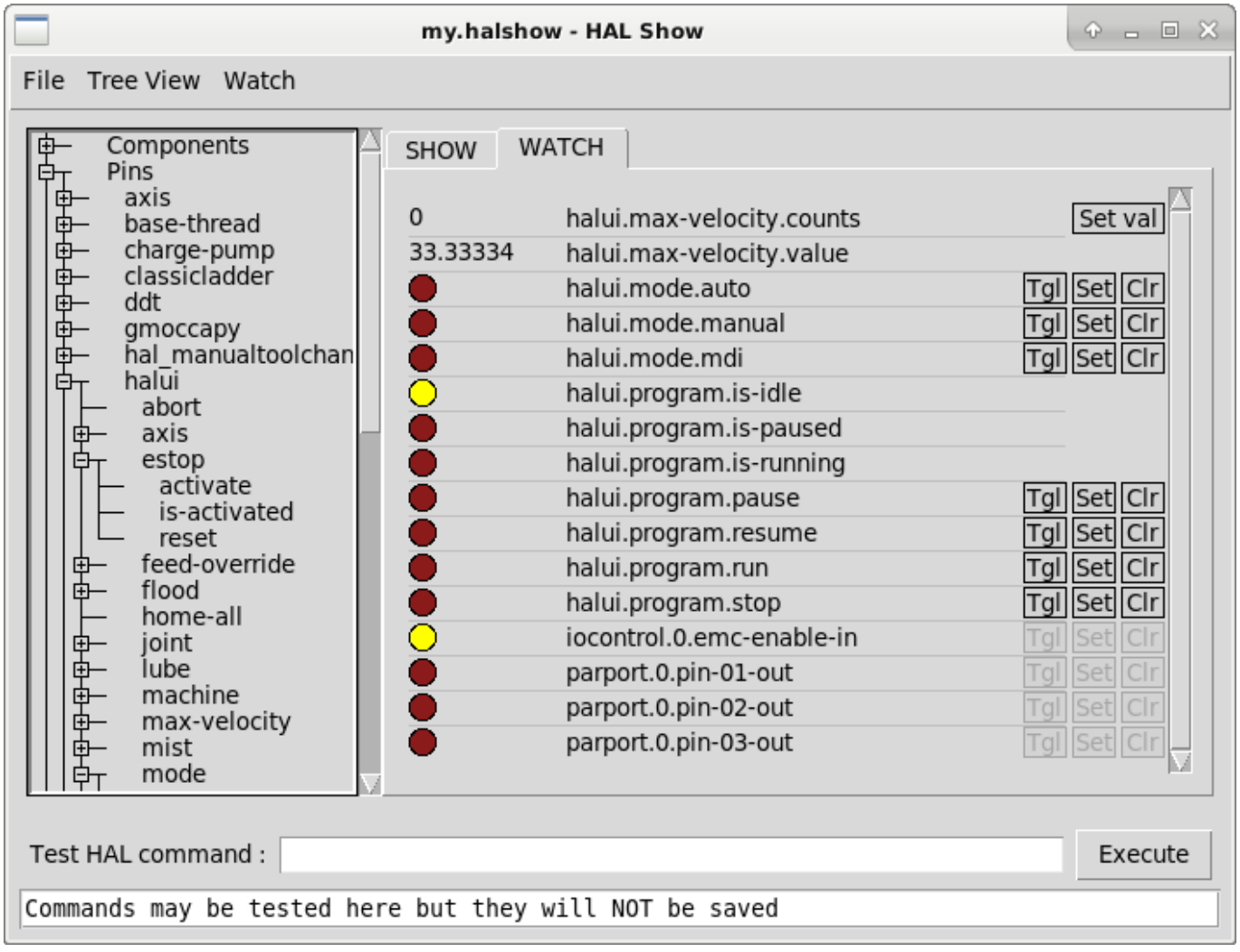


Figura 5.23: Halshow Watch Tab

A watchfile created using the *File/Save Watch List* menu item is formatted as a single line with tokens

”pin+”, ”param+”, ”sig=+”, followed by the appropriate pin, param, or signal name. The token-name pairs are separated by a space character.

Single Line Watchfile Example

```
pin+joint.0.pos-hard-limit pin+joint.1.pos-hard-limit sig+estop-loop
```

A watchfile created using the *File/Save Watch List (multiline)* menu item is formatted with separate lines for each item identified with token-name pairs as described above.

Separated Lines Watchfile Example

```
pin+joint.0.pos-hard-limit
pin+joint.1.pos-hard-limit
sig+estop-loop
```

When loading a watchfile with the *File/Load Watch List* menu item, the token-name pairs may appear as single or multiple lines. Blank lines and lines beginning with a # character are ignored.

5.15.4. Halscope

`Halscope` is an *oscilloscope* for the HAL. It lets you capture the value of pins, signals, and parameters as a function of time. Complete operating instructions should be located here eventually. For now, refer to section Halscope in the tutorial chapter, which explains the basics.

The halscope ”File” menu selector provides buttons to save a configuration or open a previously saved configuration. When halscope is terminated, the last configuration is saved in a file named autosave.halscope.

Configuration files may also be specified when starting halscope from the commandline. Commandline help (`-h`) usage:

```
halscope -h
Usage:
  halscope [-h] [-i infile] [-o outfile] [num_samples]
```

5.15.5. Sim Pin

sim_pin is a command line utility to display and update any number of writable pins, parameters or signals.

sim_pin Usage

```
Usage:
        sim_pin [Options] name1 [name2 ...] &

Options:
        --help                (this text)
        --title title_string  (window title, default: sim_pin)

Note:  LinuxCNC (or a standalone HAL application) must be running
        A named item can specify a pin, param, or signal
        The item must be writable, e.g.:
          pin:    IN or I/O (and not connected to a signal with a writer)
          param:  RW
          signal: connected to a writable pin

        HAL item types bit,s32,u32,float are supported.

        When a bit item is specified, a pushbutton is created
```

```
        to manage the item in one of three manners specified
        by radio buttons:
            toggle: Toggle value when button pressed
            pulse:  Pulse item to 1 once when button pressed
            hold:   Set to 1 while button pressed
        The bit pushbutton mode can be specified on the command
        line by formatting the item name:
            namei/mode=[toggle | pulse | hold]
        If the mode begins with an uppercase letter, the radio
        buttons for selecting other modes are not shown
```

For complete information, see the man page:

```
man sim_pin
```

sim_pin Example (with LinuxCNC running)

```
halcmd loadrt mux2 names=example; halcmd net sig_example example.in0
sim_pin example.sel example.in1 sig_example &
```

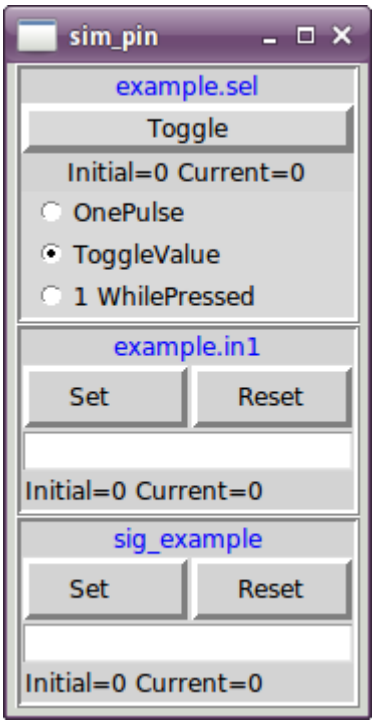


Figura 5.24: sim_pin Window

5.15.6. Simulate Probe

`simulate_probe` is a simple GUI to simulate activation of the pin motion.probe-input. Usage:

```
simulate_probe &
```

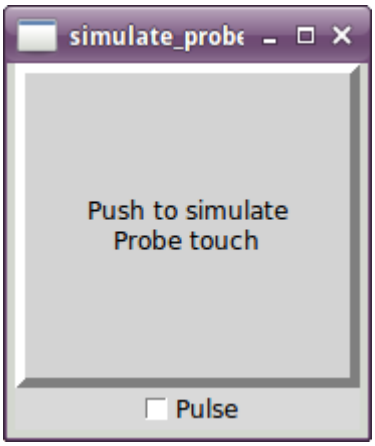


Figura 5.25: simulate_probe Window

5.15.7. HAL Histogram

`hal-histogram` is a command line utility to display histograms for HAL pins.

Uso:

```
   hal-histogram --help | -?
o
   hal-histogram [Opciones] [nombre-de-pin]
```

Cuadro 5.32: Opciones:

Option	Valor	Descripción
--minvalue	minvalue	minimum bin, default: 0
--binsize	binsize	binsize, default: 100
--nbins	nbins	number of bins, default: 50
--logscale	0/1	y axis log scale, default: 1
--text	note	text display, default: ""
--show		show count of undisplayed nbins, default off
--verbose		progress and debug, default off

Notas:

1. LinuxCNC (or another HAL application) must be running.
2. If no pinname is specified, default is: `motion-command-handler.time`.
3. This app may be opened for 5 pins.
4. Se admiten pines tipo float, s32, u32 y bit.
5. The pin must be associated with a realtime thread.

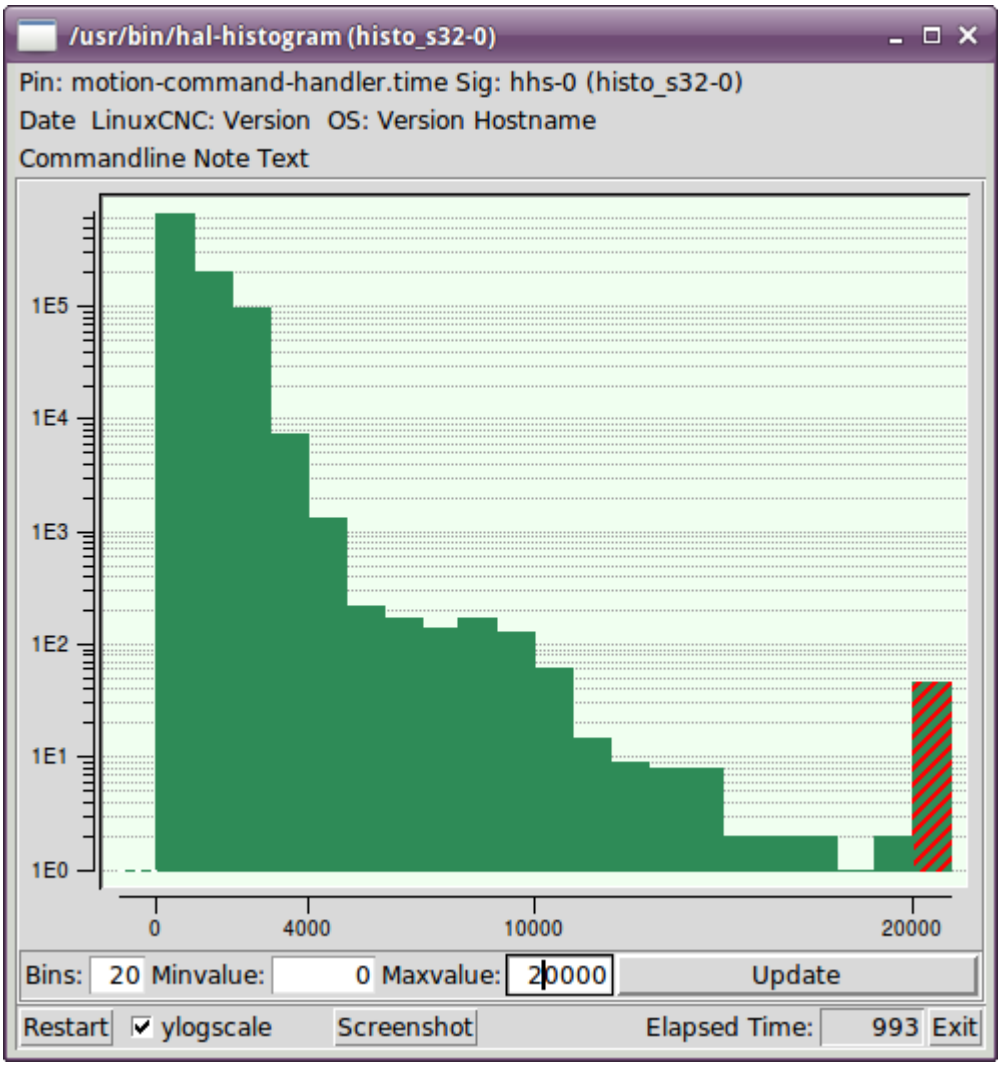


Figura 5.26: hal-histogram Window

5.15.8. Halreport

`halreport` is a command-line utility that generates a report about HAL connections for a running LinuxCNC (or other HAL) application. The report shows all signal connections and flags potential problems. Information included:

1. System description and kernel version.
2. Signals and all connected output, io, and input pins.
3. Each pin's component_function, thread, and addf-order.
4. Non-realtime component pins having non-ordered functions.
5. Identification of unknown functions for unhandled components.
6. Signals with no output.
7. Signals with no inputs.
8. Functions with no addf.

9. Warning tags for components marked as deprecated/obsolete in docs.
10. Real names for pins that use alias names.

The report can be generated from the command line and directed to an output file (or stdout if no outfilename is specified):

halreport Usage

```
Usage:
  halreport -h | --help (this help)
or
  halreport [outfilename]
```

To generate the report for every LinuxCNC startup, include halreport and an output filename as an [APPLICATIONS]APP entry in the INI file.

halreport Example

```
[APPLICATIONS]
APP = halreport /tmp/halreport.txt
```

El orden addf de funciones puede ser importante para bucles servo donde la secuencia de las funciones calculadas en cada período servo es importante. Por lo general, el orden es:

1. Read input pins,
2. do the motion command-handler and motion-controller functions,
3. perform pid calculations, and finally
4. write output pins.

For each signal in a critical path, the addf-order of the output pin should be numerically lower than the addf-order of the critical input pins that it connects to.

For routine signal paths that handle switch inputs, non-realtime pins, etc., the addf-ordering is often not critical. Moreover, the timing of non-realtime pin value changes cannot be controlled or guaranteed at the intervals typically employed for HAL threads.

Example report file excerpts showing a pid loop for a hostmot2 stepgen operated in velocity mode on a trivkins machine with `joint.0` corresponding to the X axis coordinate:

```
SIG:    pos-fb-0
  OUT:    h.00.position-fb                    hm2_7i92.0.read        servo-thread 001
          (=hm2_7i92.0.stepgen.00.position-fb)
    IN:     X_pid.feedback                    X_pid.do-pid-calcs     servo-thread 004
    IN:     joint.0.motor-pos-fb              motion-command-handler servo-thread 002
            ..................                motion-controller      servo-thread 003
...
SIG:    pos-cmd-0
  OUT:    joint.0.motor-pos-cmd               motion-command-handler servo-thread 002
          ..................                motion-controller      servo-thread 003
    IN:     X_pid.command                     X_pid.do-pid-calcs     servo-thread 004
...
SIG:    motor-cmd-0
  OUT:    X_pid.output                        X_pid.do-pid-calcs     servo-thread 004
    IN:     h.00.velocity-cmd                 hm2_7i92.0.write       servo-thread 008
            (=hm2_7i92.0.stepgen.00.velocity-cmd)
```

In the example above, the HALFILE uses halcmd aliases to simplify pin names for an hostmot2 FPGA board with commands like:

```
alias pin hm2_7i92.0.stepgen.00.position-fb h.00.position-fb
```

nota
Questionable component function detection may occur for

1. componentes no soportados (obsoletos),
2. user-created components that use multiple functions or unconventional function naming, or
3. GUI-created non-realtime components that lack distinguishing characteristics such as a prefix based on the GUI program name.

Questionable functions are tagged with a question mark ”?”.

nota
Component pins that cannot be associated with a known thread function report the function as ”Unknown”.

`halreport` generates a connections report (without pin types, and current values) for a running HAL application to aid in designing and verifying connections. This helps with the understanding what the source of a pin value is. Use this information with applications like `halshow`, `halmeter`, `halscope` or the `halcmd show` command in a terminal.

Capítulo 6

Controladores de Hardware

6.1. Parallel Port Driver

The hal_parport component is a driver for the traditional PC parallel port. The port has a total of 17 physical pins. The original parallel port divided those pins into three groups: data, control, and status. The data group consists of 8 output pins, the control group consists of 4 pins, and the status group consists of 5 input pins.

In the early 1990s, the bidirectional parallel port was introduced, which allows the data group to be used for output or input. The HAL driver supports the bidirectional port, and allows the user to set the data group as either input or output. If configured as *out*, a port provides a total of 12 outputs and 5 inputs. If configured as *in*, it provides 4 outputs and 13 inputs.

In some parallel ports, the control group pins are open collectors, which may also be driven low by an external gate. On a board with open collector control pins. If configured as *x*, it provides 8 outputs, and 9 inputs.

In some parallel ports, the control group has push-pull drivers and cannot be used as an input.

HAL and Open Collectors
HAL cannot automatically determine if the *x* mode bidirectional pins are actually open collectors (OC). If they are not, they cannot be used as inputs, and attempting to drive them LOW from an external source can damage the hardware.
To determine whether your port has *open collector* pins, load hal_parport in *x* mode. With no device attached, HAL should read the pin as TRUE. Next, insert a 470 Ω resistor from one of the control pins to GND. If the resulting voltage on the control pin is close to 0 V, and HAL now reads the pin as FALSE, then you have an OC port. If the resulting voltage is far from 0 V, or HAL does not read the pin as FALSE, then your port cannot be used in *x* mode.
The external hardware that drives the control pins should also use open collector gates, e.g. 74LS05.
On some computers, BIOS settings may affect whether *x* mode can be used. *SPP* mode is most likely to work.

No other combinations are supported, and a port cannot be changed from input to output once the driver is installed.

The parport driver can control up to 8 ports (defined by MAX_PORTS in hal_parport.c). The ports are numbered starting at zero.

6.1.1. Loading

The hal_parport driver is a real time component so it must be loaded into the real time thread with *loadrt*. The configuration string describes the parallel ports to be used, and (optionally) their types. If

the configuration string does not describe at least one port, it is an error.

```
loadrt hal_parport cfg="port [type] [port [type] ...]"
```

Specifying the Port Numbers below 16 refer to parallel ports detected by the system. This is the simplest way to configure the `hal_parport` driver and cooperates with the Linux parport_pc driver if it is loaded. A port of 0 is the first parallel port detected on the system, 1 is the next and so on.

Basic configuration This will use the first parallel port Linux detects:

```
loadrt hal_parport cfg="0"
```

Using the Port Address Instead, the port address may be specified using the hex notation with the *0x* prefix.

The config string represents the hexadecimal address of the port, optionally followed by a direction, all repeated for each port. The directions are *in*, *out*, or *x*, and determine the direction of the physical pins 2 to 9 of the D-Sub 25 connector. If the direction is not specified, the data group will by default be configured as outputs. For example:

Command to load the real-time module hal_partport with the additional <config-string> to specify the port at which the parallel-port card is expected.

```
loadrt hal_parport cfg="0x278 0x378 in 0x20A0 out"
```

This example installs the drivers for a port 0x0278, with pins 2 to 9 as outputs (by default, since neither *in* nor *out* is specified), a port 0x0378, with pins 2 to 9 as inputs and a 0x20A0 port, with pins 2 to 9 explicitly specified as outputs. Note that you must know the base address of the parallel ports to configure the drivers correctly. For ISA bus ports, this is usually not a problem, since the ports are almost always at a well-known address, such as 0x278 or 0x378 which are typically configured in the BIOS. The addresses of PCI bus cards are usually found with `lspci -v` in an *I/O ports* line, or in a kernel message after running `sudo modprobe -a parport_pc`. There is no default address, so if `<config-string>` does not contain at least one address, it is an error.

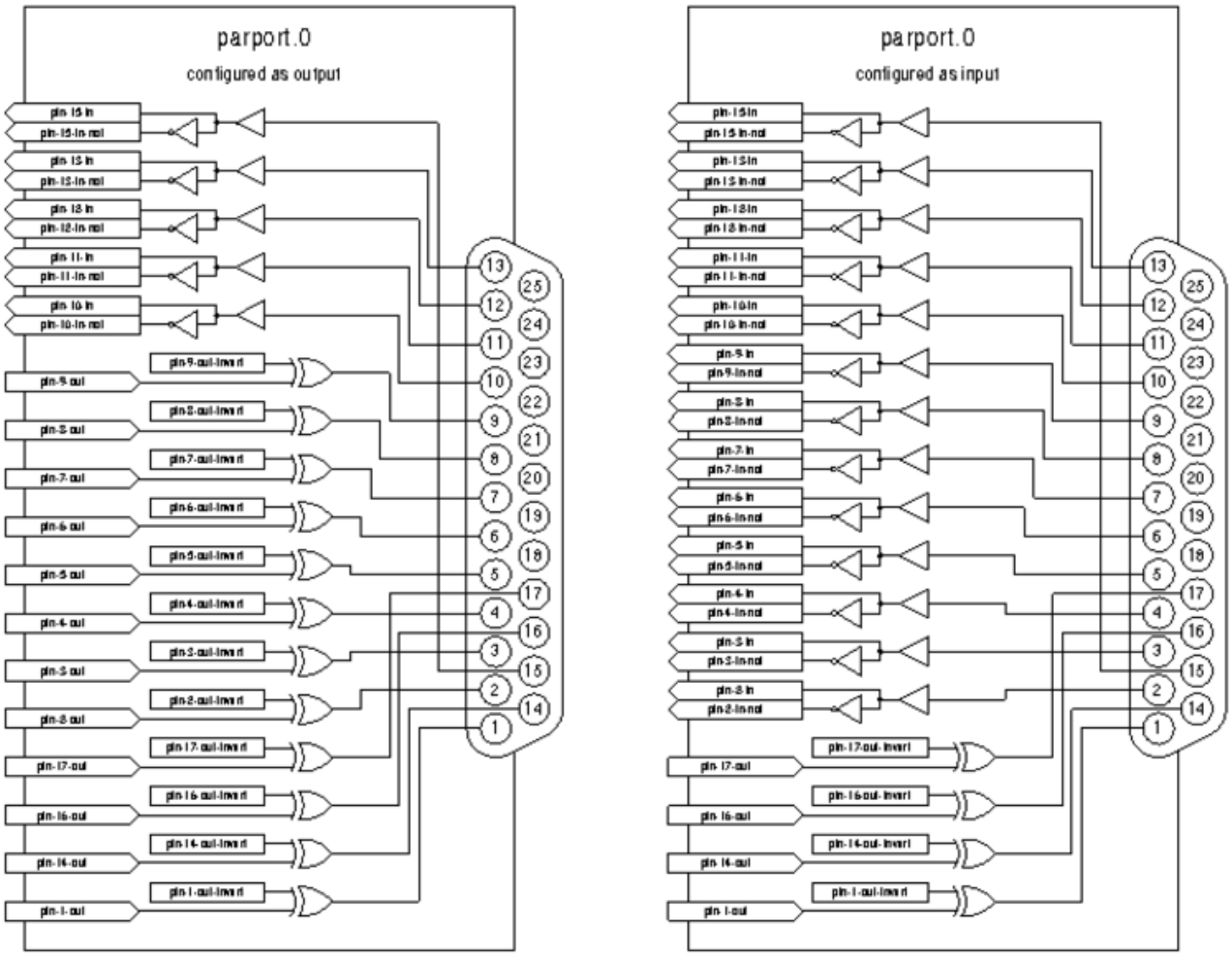


Figura 6.1: Parport block diagram

Tipo For each parallel port handled by the hal_parport driver, a *type* can optionally be specified. The type is one of *in*, *out*, *epp*, or *x*.

Cuadro 6.1: Parallel Port Direction

Pin	in	out/epp	x
1	out	out	in
2	in	out	out
3	in	out	out
4	in	out	out
5	in	out	out
6	in	out	out
7	in	out	out
8	in	out	out
9	in	out	out
10	in	in	in
11	in	in	in
12	in	in	in
13	in	in	in

Cuadro 6.1: (continued)

Pin	in	out/epp	x
14	out	out	in
15	in	in	in
16	out	out	in
17	out	out	in

If the type is not specified, the default is *out*.

A type of *epp* is the same as *out*, but the hal_parport driver requests that the port switch into EPP mode. The hal_parport driver does **not** use the EPP bus protocol, but on some systems EPP mode changes the electrical characteristics of the port in a way that may make some marginal hardware work better. The Gecko G540's charge pump is known to require this on some parallel ports.

See the Note above about mode *x*.

Example with two parallel ports This will enable two system-detected parallel ports, the first in output mode and the second in input mode:

```
loadrt hal_parport cfg="0 out 1 in"
```

Parport R/W Functions You must also direct LinuxCNC to run the *read* and *write* functions.

```
addf parport.0.read base-thread
addf parport.0.write base-thread
```

6.1.2. PCI Port Address

One good PCI parport card is made with the Netmos 9815 chipset. It has good +5 V signals, and can come in a single or dual ports.

To find the I/O addresses for PCI cards open a terminal window and use the list pci command:

```
lspci -v
```

Look for the entry with "Netmos" in it. Example of a 2-port card:

```
0000:01:0a.0 Communication controller: \
      Netmos Technology PCI 9815 Multi-I/O Controller (rev 01)
Subsystem: LSI Logic / Symbios Logic 2POS (2 port parallel adapter)
Flags: medium devsel, IRQ 5
I/O ports at b800 [size=8]
I/O ports at bc00 [size=8]
I/O ports at c000 [size=8]
I/O ports at c400 [size=8]
I/O ports at c800 [size=8]
I/O ports at cc00 [size=16]
```

From experimentation, I've found the first port (the on-card port) uses the third address listed (c000), and the second port (the one that attaches with a ribbon cable) uses the first address listed (b800). The following example shows the onboard parallel port and a PCI parallel port using the default out direction.

```
loadrt hal_parport cfg="0x378 0xc000"
```

Please note that your values will differ. The Netmos cards are Plug-N-Play, and might change their settings depending on which slot you put them into, so if you like to *get under the hood* and re-arrange things, be sure to check these values before you start LinuxCNC.

6.1.3. Pines

- parport.<p>.pin-__<n>__-out (bit) Drives a physical output pin.
- parport.<p>.pin-__<n>__-in (bit) Tracks a physical input pin.
- parport.<p>.pin-__<n>__-in-not (bit) Tracks a physical input pin, but inverted.

For each pin, <*p*> is the port number, and <*n*> is the physical pin number in the 25 pin D-shell connector.

For each physical output pin, the driver creates a single HAL pin, for example: parport.0.pin-14-out.

For each physical input pin, the driver creates two HAL pins, for example: parport.0.pin-12-in and parport.0.pin-12-in-not.

El pin HAL *-in* es VERDADERO si el pin físico es alto, y FALSO si el pin físico es bajo. El pin HAL *-in-not* está invertido y es FALSO si el pin físico es alto.

6.1.4. Parámetros

- parport.__<p>__.pin-__<n>__-out-invert (bit) Inverts an output pin.
- parport.__<p>__.pin-__<n>__-out-reset (bit) (only for -out pins) TRUE if this pin should be reset when the -reset function is executed.
- parport.__<p>__.reset-time (U32) El tiempo (en nanosegundos) entre que un pin se establece mediante -'write y se restablece mediante la función -reset si está habilitado.

The -invert parameter determines whether an output pin is active high or active low. If -invert is FALSE, setting the HAL -out pin TRUE drives the physical pin high, and FALSE drives it low. If -invert is TRUE, then setting the HAL -out pin TRUE will drive the physical pin low.

6.1.5. Funciones

- parport.__<p>__.read (funct) Reads physical input pins of port number <*p*> and updates HAL -in and -in-not pins.
- parport.read-all (funct) Reads physical input pins of all ports and updates HAL -in and -in-not pins.
- parport.__<p>__.write (funct) Reads HAL -out pins of port number <*p*> and updates that port's physical output pins.
- parport.write-all (funct) Reads HAL -out pins of all ports and updates all physical output pins.
- parport.__<p>__.reset (funct) Waits until reset-time has elapsed since the associated write, then resets pins to values indicated by -out-invert and -out-invert settings. reset must be later in the same thread as write. If -reset is TRUE, then the reset function will set the pin to the value of -out-invert. This can be used in conjunction with stepgen's *doublefreq* to produce one step per period. The stepgen stepspace for that pin must be set to 0 to enable doublefreq.

The individual functions are provided for situations where one port needs to be updated in a very fast thread, but other ports can be updated in a slower thread to save CPU time. It is probably not a good idea to use both an -all function and an individual function at the same time.

6.1.6. Common problems

If loading the module reports

```
insmod: error inserting '/home/jepler/emc2/rtlib/hal_parport.ko':
-1 Device or resource busy
```

then ensure that the standard kernel module *parport_pc* is not loaded [1] and that no other device in the system has claimed the I/O ports.

Si el módulo se carga pero no parece funcionar, entonces el puerto o la dirección puede ser incorrecta.

6.1.7. Using DoubleStep

To setup DoubleStep on the parallel port you must add the function parport.n.reset after parport.n.write and configure stepspace to 0 and the reset time wanted. So that step can be asserted on every period in HAL and then toggled off by parport after being asserted for time specified by `parport. `__n__.reset-time`.

For example:

```
loadrt hal_parport cfg="0x378 out"
setp parport.0.reset-time 5000
loadrt stepgen step_type=0,0,0
addf parport.0.read base-thread
addf stepgen.make-pulses base-thread
addf parport.0.write base-thread
addf parport.0.reset base-thread
addf stepgen.capture-position servo-thread
...
setp stepgen.0.steplen 1
setp stepgen.0.stepspace 0
```

More information on DoubleStep can be found on the wiki.

6.1.8. probe_parport

In today's PCs, parallel ports may require a plug and play (PNP) configuration before they can be used. The kernel module *probe_parport* configures all PNP ports present. It must be loaded before *hal_parport*. On machines without a PNP port, it can be loaded but will have no effect.

6.1.8.1. Installing probe_parport

If, when `parport_pc` kernel module is loaded with command:

```
sudo modprobe -a parport_pc; sudo rmmod parport_pc
```

Linux kernel outputs a message similar to:

```
parport: PnPBIOS parport detected.
```

Then use of this module will probably be necessary.

Finally, HAL parport components should be loaded:

```
loadrt probe_parport
loadrt hal_parport ...
```

[1] In the LinuxCNC packages for Ubuntu, the file /etc/modprobe.d/emc2 generally prevents *parport_pc* from being automatically loaded.

6.2. Controlador AX5214H

El AX5214H de Axiom Measurement & Control es una placa de E/S digital de 48 canales. Se conecta en un bus ISA y se asemeja a un par de chips 8255; de hecho, podrían ser un par de chips 8255, pero no estamos seguros. Cuando alguien inicia un controlador para un 8255 se debería considerar el código del ax5214, la mayoría del trabajo ya esta hecho.

6.2.1. Instalación

```
loadrt hal_ax5214h cfg="<cadena-de-configuración>"
```

La cadena de configuración consiste de una dirección de puerto hexadecimal, seguida de una cadena de 8 caracteres de "I" y "O" que establece grupos de pines como entradas y salidas. Los dos primeros caracteres establecen la dirección de los dos primeros bloques de pines de 8 bits (0-7 y 8-15). Los siguientes dos establecen bloques de 4 pines (16-19 y 20-23). El patrón se repite, dos bloques más de 8 bits (24-31 y 32-39) y dos bloques de 4 bits (40-43 y 44-47). Si se tiene instalada más de una placa, los datos de la segunda placa siguen a los de la primera. Como ejemplo, la cadena *"0x220 IIIOIIOO 0x300 OIOOIOIO"* instala controladores para dos placas. La primer placa está en la dirección 0x220, y tiene 36 entradas (0-19 y 24-39) y 12 salidas (20-23 y 40-47). La segunda placa está en la dirección 0x300, y tiene 20 entradas (8-15, 24-31 y 40-43) y 28 salidas (0-7, 16-23, 32-39 y 44-47). Se pueden usar hasta 8 placas en un sistema.

6.2.2. Pines

- *(bit) ax5214.<número-de-placa>.out-<número-de-pin>* — Maneja un pin de salida física.
- *(bit) ax5214.<número-de-placa>.in-<número-de-pin>* — Realiza el seguimiento de un pin de entrada física.
- *(bit) ax5214.<número-de-placa>.in-<número-de-pin>-not* — Realiza el seguimiento de un pin de entrada física, pero invertido.

Para cada pin, <número-de-placa> es el número de placa (comienza desde cero), y <número-de-pin> es el número de canal de E/S (de 0 a 47).

Nótese que el controlador asume señales activo BAJO. Esto es para que módulos como el OPTO-22 funcionen correctamente (verdadero significa salida encendida, o entrada energizada). Si las señales se usan directamente sin búfer o aislamiento se debe considerar la inversión. Los pines de entrada HAL son verdadero si el pin físico es bajo (módulo OPTO-22 energizado), y falso si el pin físico es alto (módulo OPTO-22 apagado). El pin HAL in-<número-de-pin>-not esta invertido — es falso si el pin físico es bajo (módulo OPTO-22 energizado). El usuario puede determinar el estado de la entrada conectando una señal a uno u otro.

6.2.3. Parámetros

- *(bit) ax5214.<número-de-placa>.out-<número-de-pin>-invert* — Invierte un pin de salida.

El parámetro -invert determina si un pin de salida está activo alto o activo bajo. Si -invert es FALSE, establecer el pin HAL de salida en TRUE lleva al pin físico a bajo, encendiendo un módulo OPTO-22 conectado, y FALSO lo lleva a alto, apagando el módulo OPTO-22. Si -invert es TRUE, entonces al establecer el pin HAL de salida en TRUE llevará el pin físico a alto y apagará el módulo.

6.2.4. Funciones

- *(funct) ax5214.<número-de-placa>.read* — Lee todas las entradas digitales en una placa.
- *(funct) ax5214.<número-de-placa>.write* — Escribe todas las salidas digitales en una placa.

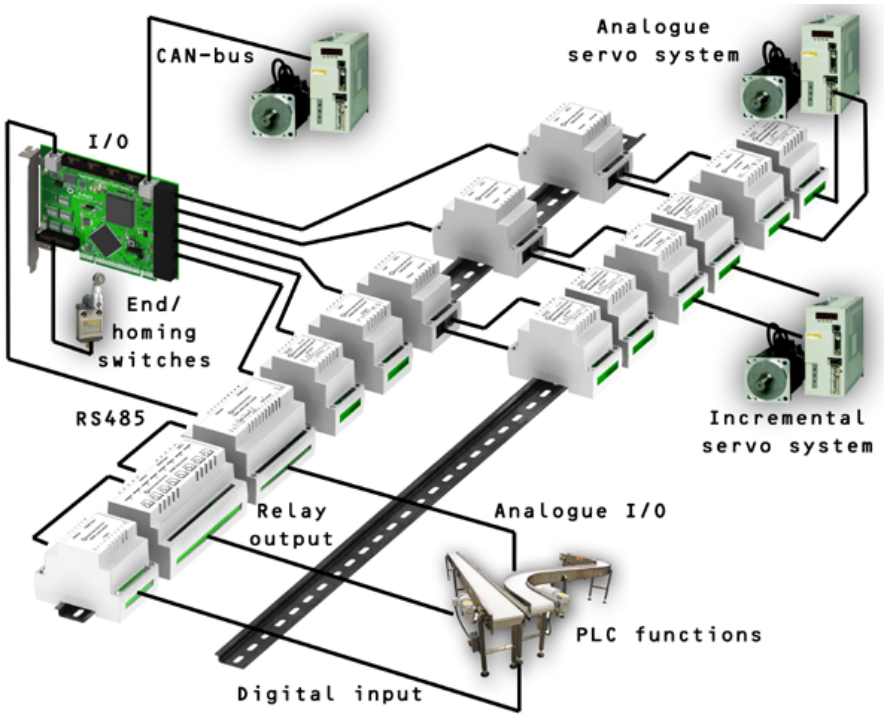
6.3. Controlador General Mechatronics

Sistema de control de movimiento basado en la tarjeta GMP6-PCI de General Mechatronics

Para una descripción detallada, por favor consulte el Manual de integración de sistema.

La tarjeta de control de movimiento GM6-PCI se basa en un FPGA y un ASIC de interfaz de puente PCI. Se puede controlar un pequeña célula de fabricación automatizada con un procedimiento de integración de sistema de poco tiempo. La figura siguiente demuestra la conexión típica de dispositivos relacionados al sistema de control:

- Puede controlar hasta seis ejes, cada uno puede ser motor a pasos o interfaz de bus CAN o servo analógico.
- GPIO: 4 grupos de ocho pines de E/S colocados en conectores de cable plano estándar.
- Módulos de expansión de E/S RS485: El bus RS485 está diseñado para interfazar con módulos de expansión montados en riel DIN compacto. Ahora están disponibles una entrada digital de 8 canales, una salida de relevadores de 8 canales y una E/S analógica (4 salidas de +/-10 Voltios y 8 entradas de +/-5 Voltios). Se pueden conectar al bus hasta 16 módulos en conjunto.
- 20 pines de entrada aislados ópticamente: Seis grupos de tres pines para la conexión directa de dos interruptores de límite y un sensor de casa para cada articulación. Y adicionalmente, dos entradas E-stop aisladas ópticamente.



Instalación:

```
loadrt hal_gm
```

Durante la carga (o intento de carga) el controlador imprime algunos mensajes de depuración útiles en la bitácora del kernel, los cuales pueden ser revisados con dmesg.

Se pueden usar hasta 3 placas en un sistema.

En la tarjeta GM6-PCI se encuentran los conectores siguientes:

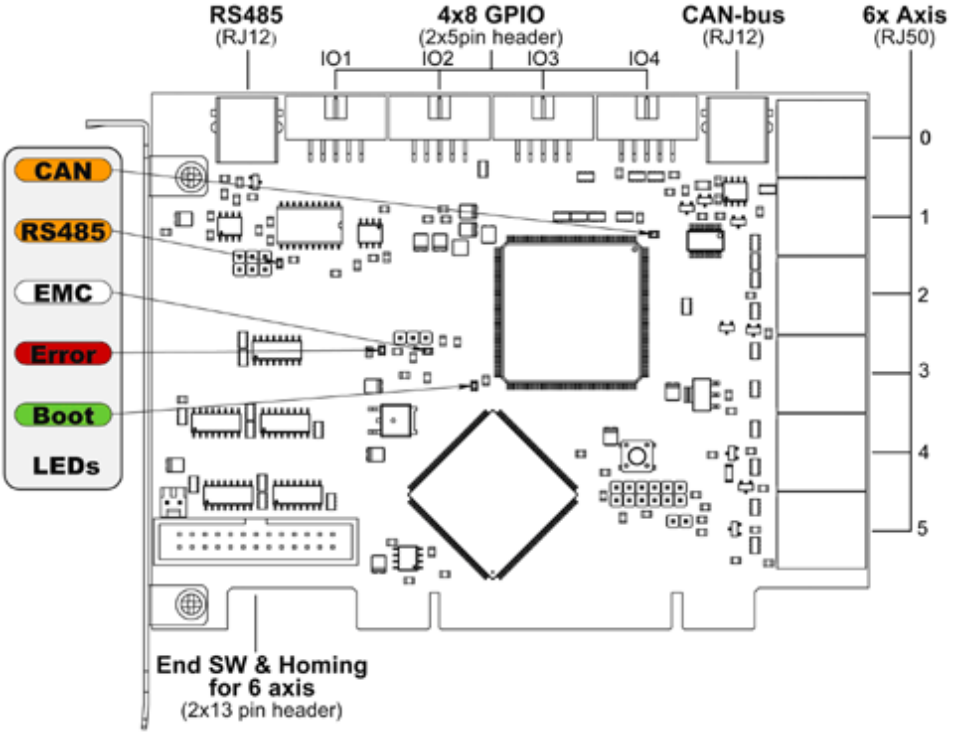


Figura 6.2: Conectores y LEDs de la tarjeta GM6-PCI

6.3.1. Conectores de E/S

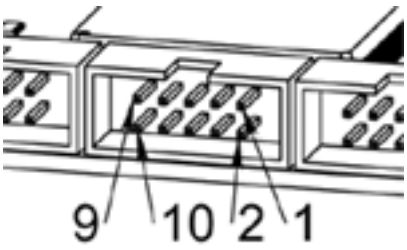


Figura 6.3: Numeración de pines de conectores GPIO

Cuadro 6.2: Asignación de pines de conectores GPIO

9	7	5	3	1
IOx/7	IOx/5	IOx/3	IOx/1	VCC

10	8	6	4	2
GND	IOx/6	IOx/4	IOx/2	IOx/0

Cada pin puede ser configurado como entrada o salida digital. La tarjeta de control de movimiento GM6-PCI tiene 4 conectores de E/S (GPIO) de propósito general, con ocho E/S configurables en cada uno. Cada pin y nombre de parámetro GPIO comienza así:

```
gm.<número_de_tarjeta>.gpio.<número_de_conector_gpio>
```

donde <*número_de_conector*> va de 1 a 3.

Estado del primer pin del primer conector GPIO en la tarjeta GM6-PCI.

```
gm.0.gpio.0.in-0
```

Los pines HAL se actualizan por función

```
gm.<número_de_tarjeta>.read
```

6.3.1.1. Pines

Cuadro 6.4: Pines GPIO

Pines	Tipo y dirección	Descripción del pin
`.in-<0-7>`	(bit, Out)	Pin de entrada
`.in-not-<0-7>`	(bit, Out)	Pin de entrada negado
`.out-<0-7>`	(bit, In)	Pin de salida. Usado solo cuando GPIO está configurado como salida.

6.3.1.2. Parámetros

Cuadro 6.5: Parámetros GPIO

Pines	Tipo y dirección	Descripción del parámetro
`.is-out-<0-7>`	(bit, R/W)	Cuando es verdadero, el GPIO correspondiente es configurado como salida en contrafase, de lo contrario se configura como entrada de alta impedancia.

Cuadro 6.5: (continued)

Pines	Tipo y dirección	Descripción del parámetro
`.invert-out-<0-7>`	(bit, R/W)	Cuando es verdadero, el valor del pin será invertido. Se usa cuando el pin está configurado como salida.

6.3.2. Conectores de eje

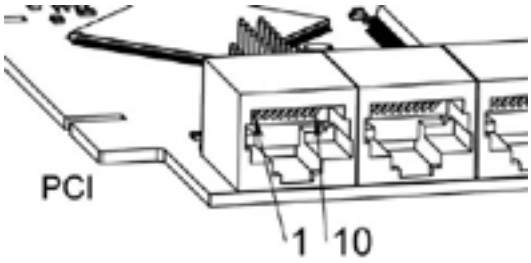


Figura 6.4: Numeración de pines en conectores de eje

Cuadro 6.6: Asignación de pines en conectores de eje

1	Codificador A
2	+5 Voltios (PC)
3	Codificador B
4	Índice de codificador
5	Falla
6	Energía habilitada
7	Paso/CCW/B
8	Dirección/CW/A
9	Tierra (PC)
10	Línea serie DAC

6.3.2.1. Módulos de interfaz de eje

Los módulos de interfaz montados en riel DIN de tamaño pequeño ofrecen una forma fácil de conectar diferentes tipos de módulo servo a los conectores de eje. Se presentan siete configuraciones de sistema diferentes en el Manual de integración de sistema para evaluar aplicaciones típicas. También ahí se pueden encontrar en descripciones detalladas de los módulos de eje.

Para evaluar la estructura servo-controlador apropiada, los módulos tienen que estar conectados como se muestra en el diagrama de bloques siguiente:

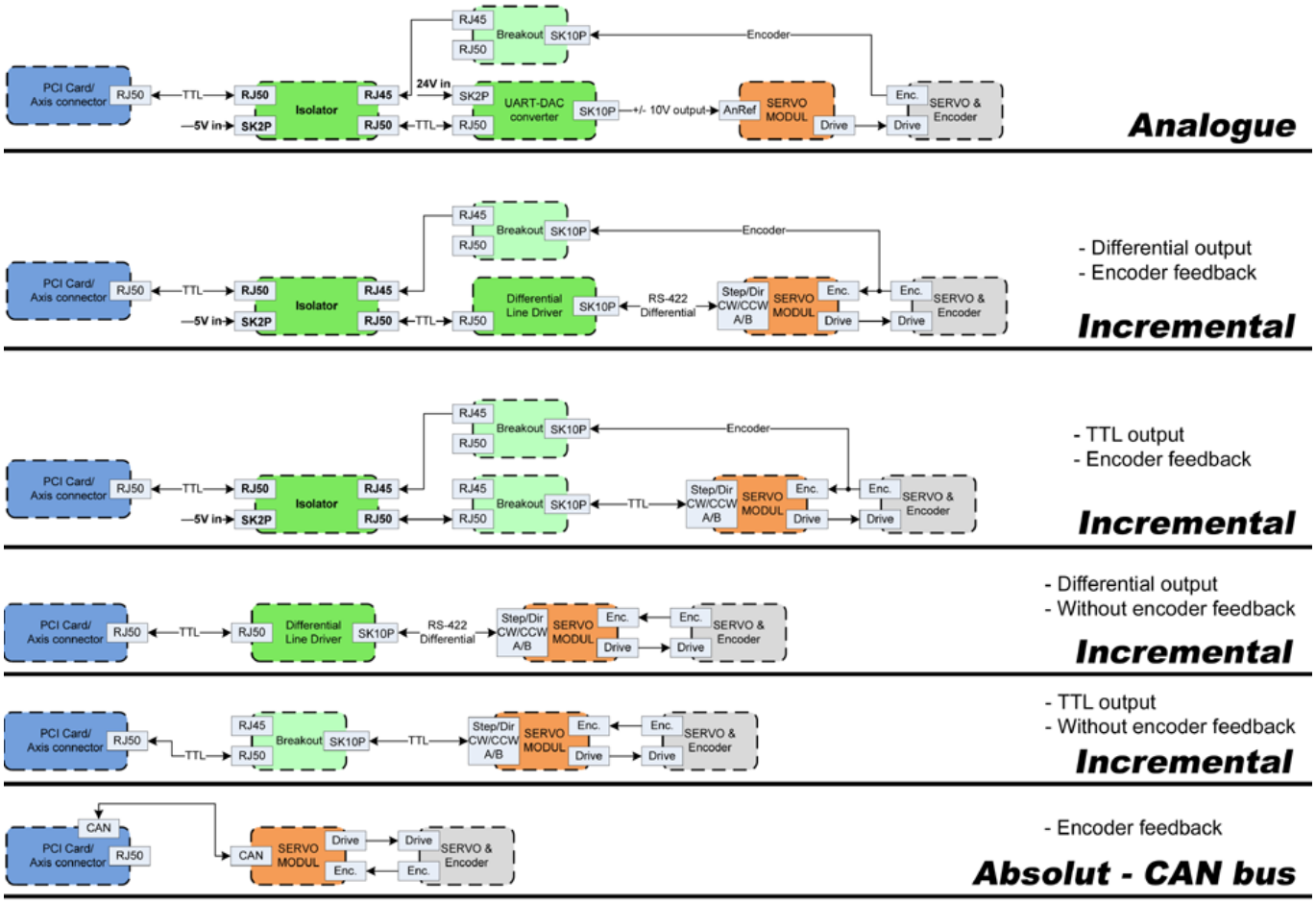


Figura 6.5: Interfaces de servoeje

6.3.2.2. Codificador

La tarjeta de control de movimiento GM6-PCI tiene seis módulos codificadores. Cada módulo codificador tiene tres canales:

- Canal A
- Canal B
- Canal I (índice)

Es capaz de contar señales de codificador en cuadratura o señales paso/dirección. Cada módulo codificador se conecta a las entradas del conector de eje RJ50 correspondiente.

Cada pin y nombre de parámetro de codificador comienza como sigue:

```
gm.<número_de_tarjeta>.encoder.<número_de_eje>
```

donde *<número_de_eje>* va de 0 a 5. Por ejemplo, `gm.0.encoder.0.position` se refiere a la posición del módulo codificador del eje 0.

La tarjeta GM6-PCI cuenta la señal del codificador independientemente de LinuxCNC. Los pines HAL se actualizan por función:

```
gm.<número_de_tarjeta>.read
```

Cuadro 6.7: Pines del codificador

Pines	**Tipo y dirección**	**Descripción del pin**
`.reset`	(bit, In)	Cuando es verdadero, reinicia conteos y posición a cero.
`.rawcounts`	(s32, Out)	La cuenta bruta es el conteo, pero no es afectada por reinicio o el pulso índice.
`.counts`	(s32, Out)	Posición en conteo de codificador.
`.position`	(float, Out)	Posición en unidades escaladas (=.counts/.position-escala).
`.index-enabled`	(bit, IO)	Cuando es True, conteos y posición se redondean o reinician (dependiendo del modo de índice) en el siguiente flanco ascendente del canal I. Cada posición de tiempo se reinicia por el índice, el pin `index-enabled` se pone en 0 y permanece así hasta que un pin HAL conectado no lo establece.
`.velocity`	(float, Out)	Velocidad en unidades escaladas por segundo. El codificador GM usa temporizador por hardware de alta frecuencia para medir el tiempo entre pulsos del codificador con el fin de calcular velocidad. Reduce en gran medida la cuantificación de ruido comparado con una simple diferenciación de la salida de posición. Cuando la velocidad medida es por debajo de min-speed-estimate, la salida de velocidad es 0.

Cuadro 6.8: Parámetros del codificador

Parámetros	**Tipo y Lectura/Escritura**	**Descripción del parámetro**
`.counter-mode`	(bit, R/W)	Cuando es verdadero, el contador cuenta cada flanco ascendente de la entrada del canal A hacia la dirección determinada por el canal B. Esto es útil para contar la salida de un solo canal (sin cuadratura) o del sensor de señal de paso/dirección. Cuando es false, cuenta en modo cuadratura.

Cuadro 6.8: (continued)

Parámetros	**Tipo y Lectura/Escritura**	**Descripción del parámetro**
`.index-mode`	(bit, R/W)	Cuando es verdadero y .index-enabled también es verdadero, .counts y .position son redondeados (con base en .counts-per-rev) en el flanco ascendente del canal I. Esto es útil para corregir algunos pulsos de error causados por ruido. En modo redondeo, es esencial establecer correctamente el parámetro .counts-per-rev. Cuando .index-mode es falso y .index-enabled es verdadero, .counts y .position son restablecidos en el pulso de canal I.
`.counts-per-rev`	(s32, R/V)	Determina cuántos conteos hay entre dos pulsos de índice. Se usa solo en modo redondeo, para cuando los parámetros .index-enabled y .index-mode son verdadero. El codificador GM procesa la señal del codificador en modo 4x, así que por ejemplo, en caso de un codificador de 500 CPR debería establecerse en 2000. Este parámetro puede medirse fácilmente estableciendo .index-enabled en True y .index-mode en False (para que .counts se reinicie en el pulso de canal I), y luego mover el eje a mano y ver la magnitud máxima del pin .counts en halmeter.
`.index-invert`	(bit, R/W)	Cuando es verdadero, el evento del canal I (reseteo o redondeo) ocurre en el flanco descendente de la señal del canal I, de lo contrario en el flanco ascendente.
`.min-speed-estimate`	(float, R/W)	Determina la magnitud mínima de velocidad medida a la que .velocity se establecerá como distinta de cero. Ajustar este parámetro demasiado bajo hará que tome mucho tiempo para que la velocidad pase a 0 después de que hayan dejado de llegar pulsos del codificador.
`.position-scale`	(float, R/W)	Factor de escala, en cuentas por unidad de longitud. .posititon=.counts/.position-scale. Por ejemplo, si la escala de posición es 2000, entonces 1000 conteos del codificador producirán una posición de 0.5 unidades.

Configuración del módulo codificador del eje 0 para recibir señal de codificador en cuadratura de 500 CPR y usar reinicio para redondear posición.

```
setp gm.0.encoder.0.counter-mode 0         # 0: cuadradura, 1: paso/dirección
setp gm.0.encoder.0.index-mode 1           # 0: reinicia posición en el índice, 1: redondea ↩
     posición en el índice
setp gm.0.encoder.0.counts-per-rev 2000    # Codificador de proceso GM en modo 4x, 4x500 ↩
    =2000
setp gm.0.encoder.0.index-invert 0         #
setp gm.0.encoder.0.min-speed-estimate 0.1 # en unidades de posición/seg.
setp gm.0.encoder.0.position-scale 20000   # 10 revoluciones del codificador hacen que la  ↩
    máquina mueva una unidad de posición (10x2000)
```

Conexión de la posición del codificador a la retroalimentación de posición articular de LinuxCNC

```
net Xpos-fb gm.0.encoder.0.position => joint.0.motor-pos-fb
```

6.3.2.3. Módulo StepGen

La tarjeta de control de movimiento GM6-PCI tiene seis módulos StepGen, una para cada articulación. Cada módulo tiene dos señales de salida. Puede producir paso/dirección, arriba/abajo o pulsos en cuadratura (A/B). Cada módulo StepGen se conecta a los pines del conector RJ50 de eje correspondiente.

Todo nombre de pin o parámetro de StepGen comienza así:

```
gm.<número_de_tarjeta>.stepgen.<número_de_eje>
```

donde *<número_de_eje>* va de 0 a 5. Por ejemplo, `gm.0.stepgen.0.position-cmd` se refiere al comando de posición del módulo StepGen del eje 0 en la tarjeta 0.

La tarjeta GM6-PCI genera pulsos de paso independientemente de LinuxCNC. Los pines HAL son actualizados por función

```
gm.<número_de_tarjeta>.write
```

Cuadro 6.9: Pines del módulo StepGen

Pines	Tipo y dirección	Descripción del pin
`.enable`	(bit, In)	StepGen produce pulsos solo cuando este pin es verdadero.
`.count-fb`	(s32, Out)	Retroalimentación de posición en unidades de conteo.
`.position-fb`	(float, Out)	Retroalimentación de posición en unidades de posición.
`.position-cmd`	(float, In)	Posición ordenada en unidades de posición. Usado solo en modo posición.
`.velocity-cmd`	(float, In)	Velocidad ordenada en unidades de posición por segundo. Se usa solo en modo velocidad.

Cuadro 6.10: Parámetros del módulo StepGen

Parámetros	**Tipo y Lectura/Escritura**	**Descripción del parámetro**
`.step-type`	(u32, R/W)	Con 0, el módulo produce señal paso/dirección. Con 1 produce señales de paso arriba/abajo. Y con 2 produce señales de salida en cuadratura.
`.control-type`	(bit, R/W)	Cuando es True, se usa .velocity-cmd como referencia y el control de velocidad calcula la tasa de pulso de salida. Cuando es False, se usa .position-cmd como referencia y el control de posición calcula la tasa de pulso de salida.
`.invert-step1`	(bit, R/W)	Invierte la salida del canal 1 (Señal de paso en modo paso/dirección)
`.invert-step2`	(bit, R/W)	Invierte la salida del canal 2 (Señal de dirección en modo paso/dirección)
`.maxvel`	(float, R/W)	Velocidad máxima en unidades de posición por segundo. Si se establece 0.0 se ignora el parámetro .maxvel.
`.maxaccel`	(float, R/W)	Aceleración máxima en unidades de posición por segundo al cuadrado. mf es 0.0, el parámetro .maxaccel es ignorado.
`.position-scale`	(float, R/W)	Escala en pasos por unidad de longitud.
`.steplen`	(u32, R/W)	Longitud de pulso de paso en nanosegundos.
`.stepspace`	(u32, R/W)	Tiempo mínimo entre dos pulsos de paso en nanosegundos.
`.dirdelay`	(u32, R/W)	Tiempo mínimo entre el cambio de pulso de paso y dirección en nanosegundos.

Para evaluar los valores apropiados ver los diagramas de tiempo abajo:

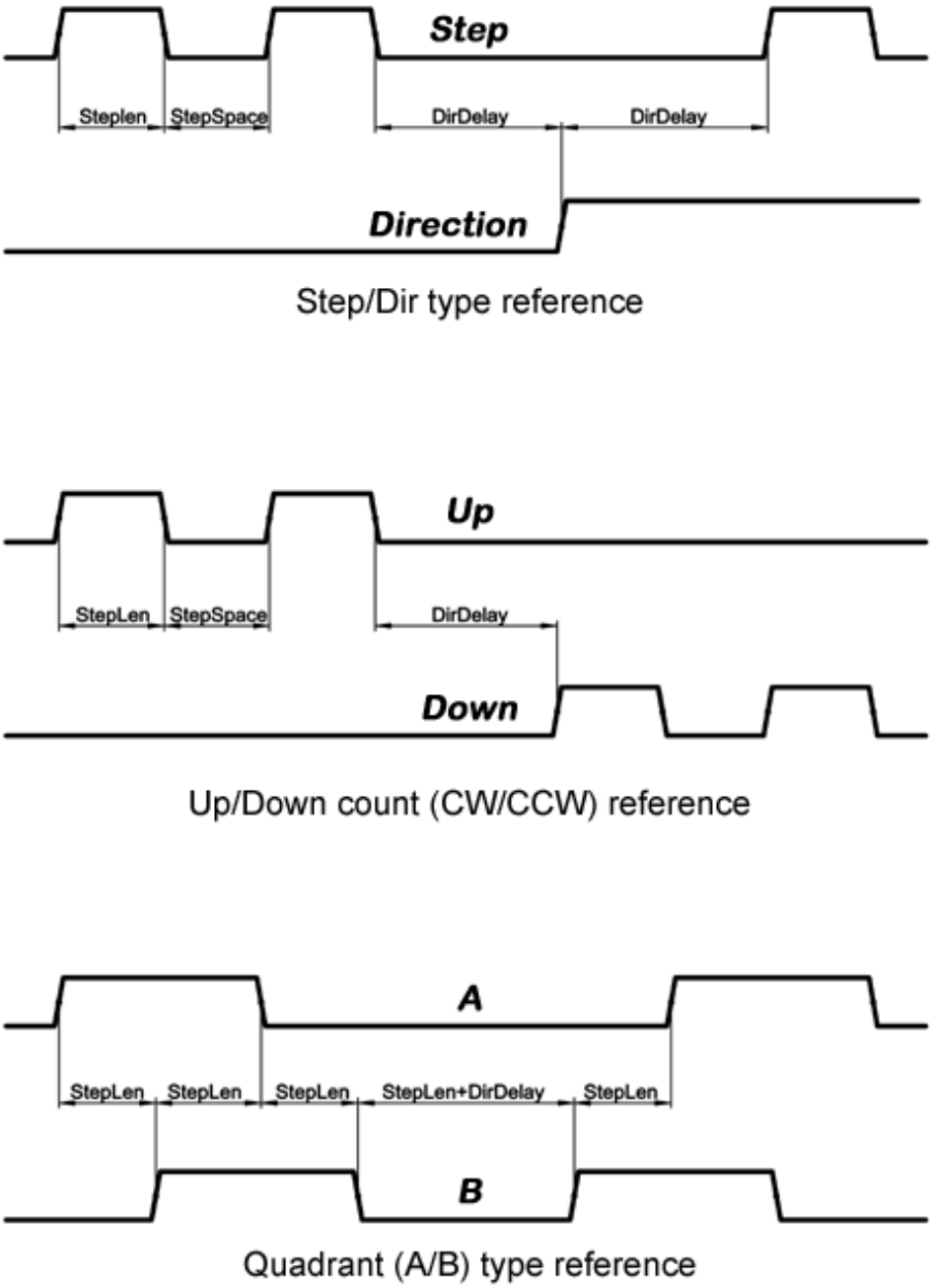


Figura 6.6: Diagramas de referencia de tiempo de señal

Configuración del módulo StepGen del eje 0 para generar 1000 pulsos de paso por unidad de posición

```
setp gm.0.stepgen.0.step-type 0         # 0:paso/dirección, 1:arriba/abajo, 2:cuadratura
setp gm.0.stepgen.0.control-type 0      # 0:Control de posición, 1:Control de velocidad
setp gm.0.stepgen.0.invert-step1 0
setp gm.0.stepgen.0.invert-step2 0
setp gm.0.stepgen.0.maxvel 0            # no se asigna maxvel para generador de pasos
                                        # deja al interpolador controlarla.
setp gm.0.stepgen.0.maxaccel 0          # no se asigna aceleración máxima para generador de ↩
     pasos
                                        # deja al interpolador controlarla.
setp gm.0.stepgen.0.position-scale 1000 # 1000 pasos/unidades de posición
```

```
setp gm.0.stepgen.0.steplen 1000        # 1000 ns = 1 µs
setp gm.0.stepgen.0.stepspace1000       # 1000 ns = 1 µs
setp gm.0.stepgen.0.dirdelay 2000       # 2000 ns = 2 µs
```

Conexión de StepGen a la posición de referencia del eje 0 y habilitación de pines

```
net Xpos-cmd joint.0.motor-pos-cmd => gm.0.stepgen.0.position-cmd
net Xen joint.0.amp-enable-out => gm.0.stepgen.0.enable
```

6.3.2.4. Señales de habilitación y falla

La tarjeta de control de movimiento GM6-PCI tiene una salida de habilitación y una entrada de falla, ambos pines HAL están conectados a cada conector de eje RJ50 y al conector CAN.

Los pines HAL son actualizados por función:

```
gm.<número_de_tarjeta>.read
```

Cuadro 6.11: Pines de señal de habilitación y falla

Pines	Tipo y dirección	Descripción del pin
gm.<número_de_tarjeta>.power-enable	bit, Out	Si este pin es True, * y el temporizador de Watch Dog no ha expirado * y no hay fallo de energía entonces los pines de energía habilitada del eje y los conectores CAN se establecen en alto, de lo contrario en bajo.
gm.<número_de_tarjeta>.power-fault	bit, In	Entrada de fallo de energía.

6.3.2.5. Axis DAC

The GM6-PCI motion control card has six serial axis DAC driver modules, one for each joint. Each module is connected to the pin of the corresponding RJ50 axis connector. Every axis DAC pin and parameter name begins as follows:

```
gm.<card_no>.dac.<axis_no>
```

where <*axis_no*> is from 0 to 5. For example, `gm.0.dac.0.value` refers to the output voltage of DAC module of axis 0.

Los pines HAL son actualizados por función:

```
gm.<número_de_tarjeta>.write
```

Cuadro 6.12: Axis DAC pins

Pines	Tipo y dirección	Descripción del pin
.enable	(bit, In)	Enable DAC output. When enable is false, DAC output is 0.0 V.
.value	(float, In)	Value of DAC output in Volts.

Cuadro 6.13: Axis DAC parameters

Parámetros	Tipo y dirección	Descripción del parámetro
.offset	(float, R/W)	Offset is added to the value before the hardware is updated.
.high-limit	(float, R/W)	Maximum output voltage of the hardware in Volts.
.low-limit	(float, R/W)	Minimum output voltage of the hardware in Volts.
.invert-serial	(float, R/W)	GM6-PCI card is communicating with DAC hardware via fast serial communication to highly reduce time delay compared to PWM. DAC module is recommended to be isolated which is negating serial communication line. In case of isolation, leave this parameter to default (0), while in case of none-isolation, set this parameter to 1.

6.3.3. CAN-bus servo amplifiers

The GM6-PCI motion control card has CAN module to drive CAN servo amplifiers. Implementation of higher level protocols like CANopen is further development. Currently GM produced power amplifiers has upper level driver which export pins and parameters to HAL. They receive position reference and provide encoder feedback via CAN bus.

The frames are standard (11 bit) ID frames, with 4 byte data length. The baud rate is 1 Mbit/s. The position command IDs for axis 0..5 are 0x10..0x15. The position feedback IDs for axis 0..5 are 0x20..0x25.

These configuration can be changed with the modification of hal_gm.c and recompiling LinuxCNC.

Every CAN pin and parameter name begins as follows:

```
gm.<card_no>.can-gm.<axis_no>
```

where <*axis_no*> is from 0 to 5. For example, `gm.0.can-gm.0.position` refers to the output position of axis 0 in position units.

Los pines HAL son actualizados por función:

```
gm.<número_de_tarjeta>.write
```

6.3.3.1. Pines

Cuadro 6.14: CAN module pins

Pines	Tipo y dirección	Descripción del pin
`.enable`	(bit, In)	Enable sending position references.
`.position-cmd`	(float, In)	Commanded position in position units.
`.position-fb`	(float, In)	Feed back position in position units.

6.3.3.2. Parámetros

Cuadro 6.15: CAN module parameters

Parámetros	Tipo y dirección	Descripción del parámetro
`.position-scale`	(float, R/W)	Scale in per length unit.

6.3.4. Watchdog timer

El temporizador de watchdog se reinicia en la función:

```
gm.<número_de_tarjeta>.read
```

6.3.4.1. Pines

Cuadro 6.16: Watchdog pins

Pines	Tipo y dirección	Descripción del pin
`gm.<card_no>.watchdog-expired`	(bit, Out)	Indicates that watchdog timer is expired.

Watchdog timer overrun causes the set of power-enable to low in hardware.

6.3.4.2. Parámetros

Cuadro 6.17: Watchdog parameters

Parámetros	Tipo y dirección	Descripción del parámetro
gm.<*card_no*>.watchdog-enable	(bit, R/W)	Enables watchdog timer. It is strongly recommended to enable the watchdog timer, because it can disable all the servo amplifiers by pulling down all enable signals in case of a PC error.
gm.<*card_no*>.watchdog-timeout	(float, R/W)	Time interval in within the gm.<*card_no*>.read function must be executed. The gm.<*card_no*>.read is typically added to servo-thread, so watch timeout is typically set to 3 times of the servo period.

6.3.5. End-, homing- and E-stop switches

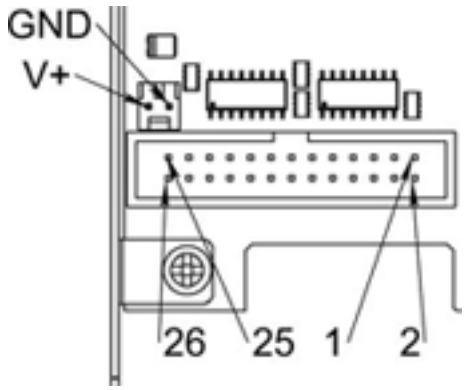


Figura 6.7: Pin numbering of homing & end switch connector

Cuadro 6.18: End- and homing switch connector pinout

25	23	21	19	17	15	13	11	9	7	5	3	1
GND		1/End-	2/End+	2/Homing	3/End-	4/End+	4/Homing	5/End-	6/End+	6/Homing	E-Stop 2	V+ (Ext.)

26	24	22	20	18	16	14	12	10	8	6	4	2
GND		1/End+	1/Homing	2/End-	3/End+	3/Homing	4/End-	5/End+	5/Homing	6/End-	E-Stop 1	V+ (Ext.)

The GM6-PCI motion control card has two limit- and one homing switch input for each joint. All the names of these pins begin as follows:

```
gm.<card_no>.joint.<axis_no>
```

where *<axis_no>* is from 0 to 5. For example, `gm.0.joint.0.home-sw-in` indicates the state of the axis 0 home switch.

Los pines HAL son actualizados por función:

```
gm.<número_de_tarjeta>.read
```

6.3.5.1. Pines

Cuadro 6.20: Pines de interruptores End- y homing

Pines	Tipo y dirección	Descripción del pin
`.home-sw-in`	(bit, Out)	Home switch input
`.home-sw-in-not`	(bit, Out)	Negated home switch input
`.neg-lim-sw-in`	(bit, Out)	Negative limit switch input
`.neg-lim-sw-in-not`	(bit, Out)	Negated negative limit switch input
`.pos-lim-sw-in`	(bit, Out)	Positive limit switch input
`.pos-lim-sw-in-not`	(bit, Out)	Negated positive limit switch input

6.3.5.2. Parámetros

Cuadro 6.21: E-stop switch parameters

Parámetros	Tipo y dirección	Descripción del parámetro
`gm.0.estop.0.in`	(bit, Out)	Estop 0 input
`gm.0.estop.0.in-not`	(bit, Out)	Entrada Estop 0 invertida
`gm.0.estop.1.in`	(bit, Out)	Estop 1 input
`gm.0.estop.1.in-not`	(bit, Out)	Entrada Estop 1 negada

6.3.6. Status LEDs

6.3.6.1. CAN

Color: Orange

- Blink, during data communication.
- On, when any of the buffers are full - communication error.
- Off, when no data communication.

6.3.6.2. RS485

Color: Orange

- Blink, during initialization of modules on the bus
- On, when the data communication is up between all initialized modules.
- Off, when any of the initialized modules dropped off because of an error.

6.3.6.3. EMC

Color: White

- Parpadea, cuando LinuxCNC esta ejecutándose.
- De lo contrario apagado.

6.3.6.4. Boot

Color: Green

- On, when system booted successfully.
- De lo contrario apagado.

6.3.6.5. Error

Color: Red

- Off, when there is no fault in the system.
- Blink, when PCI communication error.
- On, when watchdog timer overflowed.

6.3.7. RS485 I/O expander modules

These modules were developed for expanding the I/O and function capability along an RS485 line of the GM6-PCI motion control card.

Available module types:

- 8-channel relay output module - gives eight NO-NC relay output on a three pole terminal connector for each channel.
- 8-channel digital input module - gives eight optical isolated digital input pins.
- 8 channel ADC and 4-channel DAC module - gives four digital-to-analogue converter outputs and eight analogue-to-digital inputs. This module is also optically isolated from the GM6-PCI card.

Automatic node recognizing Each node connected to the bus was recognized by the GM6-PCI card automatically. During starting LinuxCNC, the driver export pins and parameters of all available modules automatically.

Fault handling If a module does not answer regularly the GM6-PCI card drops down the module. If a module with output do not gets data with correct CRC regularly, the module switch to error state (green LED blinking), and turns all outputs to error state.

Connecting the nodes The modules on the bus have to be connected in serial topology, with termination resistors on the end. The start of the topology is the PCI card, and the end is the last module.

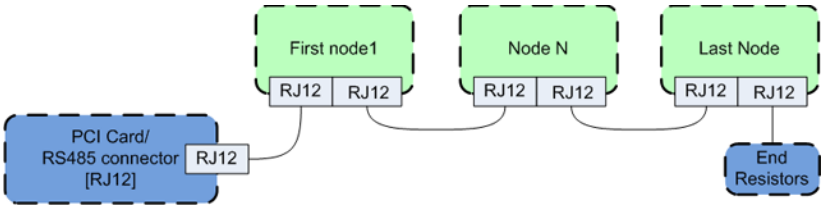


Figura 6.8: Connecting the RS485 nodes to the GM6-PCI card

Addressing Each node on the bus has a 4 bit unique address that can be set with a red DIP switch.

Status LED A green LED indicates the status of the module:

- Blink, when the module is only powered, but not jet identified, or when module is dropped down.
- Off, during identification (computer is on, but LinuxCNC not started)
- On, when it communicates continuously.

6.3.7.1. Relay output module

For pinout, connection and electrical charasteristics of the module, please refer to the System integration manual.

Todos los pines y parámetros se actualizan en la función siguiente:

```
gm.<card_no>.rs485
```

It should be added to servo thread or other thread with larger period to avoid CPU overload. Every RS485 module pin and parameter name begins as follows:

```
gm.<card_no>.rs485.<module ID>
```

where <*module ID*> is from 00 to 15.

Cuadro 6.22: Relay output module pins

Pines	Tipo y dirección	Descripción del pin
`.relay-<0-7>`	(bit, Out)	Output pin for relay

Cuadro 6.23: Relay output module parameters

Parámetros	Tipo y dirección	Descripción del parámetro
`.invert-relay-<0-7>`	(bit, R/W)	Negate relay output pin

Ejemplo HAL

```
  gm.0.rs485.0.relay-0   # First relay of the node.
# gm.0                   # Identifies the first GM6-PCI motion control card (PCI card
    address = 0)
#     .rs485.0           # Selects node with address 0 on the RS485 bus
#             .relay-0   # Selects the first relay
```

6.3.7.2. Digital input module

For pinout, connection and electrical charasteristics of the module, please refer to the System integration manual.

Todos los pines y parámetros se actualizan en la función siguiente:

```
gm.<card_no>.rs485
```

It should be added to servo thread or other thread with larger period to avoid CPU overload. Every RS485 module pin and parameter name begins as follows:

```
gm.<card_no>.rs485.<module ID>
```

where <*module ID*> is from 00 to 15.

Cuadro 6.24: Digital input output module pins

Pines	Tipo y dirección	Descripción del pin
`.in-<0-7>`	(bit, Out)	Pin de entrada
`.in-not-<0-7>`	(bit, Out)	Pin de entrada negado

Ejemplo HAL

```
  gm.0.rs485.0.in-0   # First input of the node.
# gm.0                # Identifies the first GM6-PCI motion control card (PCI card address
    = 0)
#     .rs485.0        # Selects node with address 0 on the RS485 bus
#             .in-0   # Selects the first digital input module
```

6.3.7.3. DAC & ADC module

For pinout, connection and electrical charasteristics of the module, please refer to the System integration manual.

Todos los pines y parámetros se actualizan en la función siguiente:

```
gm.<card_no>.rs485
```

It should be added to servo thread or other thread with larger period to avoid CPU overload. Every RS485 module pin and parameter name begins as follows:

```
gm.<card_no>.rs485.<module ID>
```

where <*module ID*> is from 00 to 15.

Cuadro 6.25: DAC & ADC module pins

Pines	Tipo y dirección	Descripción del pin
.adc-<*0-7*>	(float, Out)	Value of ADC input in Volts.
.dac-enable-<*0-3*>	(bit, In)	Enable DAC output. When enable is false then DAC output is set to 0.0 V.
.dac-<*0-3*>	(float, In)	Value of DAC output in Volts.

Cuadro 6.26: DAC & ADC module parameters

Parámetros	Tipo y dirección	Descripción del parámetro
.adc-scale-<*0-7*>	(float, R/W)	The input voltage will be multiplied by scale before being output to .adc- pin.
.adc-offset-<*0-7*>	(float, R/W)	Offset is subtracted from the hardware input voltage after the scale multiplier has been applied.
.dac-offset-<*0-3*>	(float, R/W)	Offset is added to the value before the hardware is updated.
.dac-high-limit-<*0-3*>	(float, R/W)	Maximum output voltage of the hardware in Volts.
.dac-low-limit-<*0-3*>	(float, R/W)	Minimum output voltage of the hardware in Volts.

Ejemplo HAL

```
  gm.0.rs485.0.adc-0   # First analogue channel of the node.
# gm.0                 # Identifies the first GM6-PCI motion control card (PCI card address ↩
    = 0)
#     .rs485.0         # Selects node with address 0 on the RS485 bus
#             .adc-0   # Selects the first analogue input of the module
```

6.3.7.4. Teach Pendant module

For pinout, connection and electrical charasteristics of the module, please refer to the System integration manual.

Todos los pines y parámetros se actualizan en la función siguiente:

```
gm.<card_no>.rs485
```

It should be added to servo thread or other thread with larger period to avoid CPU overload. Every RS485 module pin and parameter name begins as follows:

```
gm.<card_no>.rs485.<module ID>
```

where *<module ID>* is from 00 to 15. Note that on the Teach Pendant module it cannot be changed, and pre-programmed as zero. Upon request it can be delivered with firmware pre-programmed different ID.

Cuadro 6.27: Teach Pendant module pins

Pines	Tipo y dirección	Descripción del pin
`.adc-<0-5>`	(float, Out)	Value of ADC input in Volts.
`.enc-reset`	(bit, In)	Cuando es verdadero, reinicia conteos y posición a cero.
`.enc-counts`	(s32, Out)	Posición en conteo de codificador.
`.enc-rawcounts`	(s32, Out)	The raw count is the counts, but unaffected by reset.
`.enc-position`	(float, Out)	Position in scaled units (=.enc-counts/.enc-position-scale).
`.in-<0-7>`	(bit, Out)	Pin de entrada
`.in-not-<0-7>`	(bit, Out)	Pin de entrada negado

Cuadro 6.28: Teach Pendant module parameters

Parámetros	Tipo y dirección	Descripción del parámetro
`.adc-scale-<0-5>`	(float, R/W)	The input voltage will be multiplied by scale before being output to .adc- pin.
`.adc-offset-<0-5>`	(float, R/W)	Offset is subtracted from the hardware input voltage after the scale multiplier has been applied.
`.enc-position-scale`	(float, R/W)	Scale in per length unit.

Ejemplo HAL

```
  gm.0.rs485.0.adc-0  # First analogue channel of the node.
# gm.0                # Identifies the first GM6-PCI motion control card (PCI card address
    = 0)
#     .rs485.0        # Selects node with address 0 on the RS485 bus
#             .adc-0  # Selects the first analogue input of the module
```

6.3.8. Errata

6.3.8.1. GM6-PCI card Errata

The revision number in this section refers to the revision of the GM6-PCI card device.

Rev. 1.2

- Error: The PCI card do not boot, when Axis 1. END B switch is active (low). Found on November 16, 2013.
- Reason: This switch is connected to a boot setting pin of FPGA
- Problem fix/workaround: Use other switch pin, or connect only normally open switch to this switch input pin.

6.4. GS2 VFD Driver

This is a non-realtime HAL program for the GS2 series of VFDs at Automation Direct. [2]

This component is loaded using the halcmd "loadusr" command:

```
loadusr -Wn spindle-vfd gs2_vfd -n spindle-vfd
```

The above command says: loadusr, wait for named to load, component gs2_vfd, named spindle-vfd. The HAL `loadusr` command is described in the loadusr chapter.

6.4.1. Command Line Options

- *-b* or *--bits* <n> (default: 8) Set number of data bits to *n*, where *n* must be from 5 to 8 inclusive.
- *-d* or *--device <path>* (default: /dev/ttyS0) Set the file path to the serial device node to use.
- *-g* or *--debug* Turn on debugging messages. This will also set the verbose flag. Debug mode will cause all modbus messages to be printed in hex on the terminal.
- *-n* or *--name <string>* (default: gs2_vfd) Set the name of the HAL module. The HAL comp name will be set to <string>, and all pin and parameter names will begin with <string>.
- *-p* or *--parity {even,odd,none}* (default: odd) Set serial parity to even, odd, or none.
- *-r* or *--rate* <n> (default: 38400) Set baud rate to *n*. It is an error if the rate is not one of the following: 110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.
- *-s* or *--stopbits {1,2}* (default: 1) Set serial stop bits to 1 or 2
- *-t* or *--target* <n> (default: 1) Set MODBUS target (slave) number. This must match the device number you set on the GS2.
- *-v* or *--verbose* Turn on debug messages.
- *-A* or *--accel-seconds <n>* (default: 10.0) Seconds to accelerate the spindle from 0 to max. RPM.
- *-D* or *--decel-seconds <n>* (default: 0.0) Seconds to decelerate the spindle from max. RPM to 0. If set to 0.0 the spindle will be allowed to coast to a stop without controlled deceleration.

[2] In Europe the equivalent can be found under the brand name Omron.

- *-R* or *--braking-resistor* This argument should be used when a braking resistor is installed on the GS2 VFD (see Appendix A of the GS2 manual). It disables deceleration over-voltage stall prevention (see GS2 modbus Parameter 6.05), allowing the VFD to keep braking even in situations where the motor is regenerating high voltage. The regenerated voltage gets safely dumped into the braking resistor.

---

nota
That if there are serial configuration errors, turning on verbose may result in a flood of timeout errors.

6.4.2. Pines

With <*name*> being ”gs2_vfd” or the name given during loading with the `-n` option:

- <name>.*DC-bus-volts* (float, out) DC bus voltage of the VFD
- <nombre>.*at-speed* (bit, out) cuando el controlador esta la velocidad ordenada
- <name>.*err-reset* (bit, in) reset errors sent to VFD
- <name>.*firmware-revision* (s32, out) from the VFD
- <name>.*frequency-command* (float, out) from the VFD
- <name>.*frequency-out* (float, out) from the VFD
- <name>.*is-stopped* (bit, out) when the VFD reports 0 Hz output
- <name>.*load-percentage* (float, out) from the VFD
- <name>.*motor-RPM* (float, out) from the VFD
- <name>.*output-current* (float, out) from the VFD
- <name>.*output-voltage* (float, out) from the VFD
- <name>.*power-factor* (float, out) from the VFD
- <name>.*scale-frequency* (float, out) from the VFD
- <name>.*speed-command* (float, in) speed sent to VFD in RPM It is an error to send a speed faster than the Motor Max RPM as set in the VFD.
- <name>.*spindle-fwd* (bit, in) 1 for FWD and 0 for REV sent to VFD
- <name>.*spindle-rev* (bit, in) 1 for REV and 0 if off
- <name>.*spindle-on* (bit, in) 1 for ON and 0 for OFF sent to VFD
- <name>.*status-1* (s32, out) Drive Status of the VFD (see the GS2 manual)
- <name>.*status-2* (s32, out) Drive Status of the VFD (see the GS2 manual)

---

nota
The status value is a sum of all the bits that are on. So a 163 which means the drive is in the run mode is the sum of 3 (run) + 32 (freq set by serial) + 128 (operation set by serial).

6.4.3. Parámetros

With <*name*> being gs2_vfd or the name given during loading with the `-n` option:

- <name>.*error-count* (s32, RW)
- <name>.*loop-time* (float, RW) how often the modbus is polled (default: 0.1)
- <name>.*nameplate-HZ* (float, RW) Nameplate Hz of motor (default: 60)
- <name>.*nameplate-RPM* (float, RW) Nameplate RPM of motor (default: 1730)
- <name>.*retval* (s32, RW) the return value of an error in HAL
- <name>.*tolerance* (s32, RW) speed tolerance (default: 0.01)
- <name>.*ack-delay* (s32, RW) number of read/write cycles before checking at-speed (default 2)

For an example of using this component to drive a spindle see the GS2 Spindle example.

6.5. HAL Driver for Raspberry Pi GPIO pins

Note: This driver will not be compiled into images aimed at non-ARM CPUS. It is only really intended to work on the Raspberry Pi. It may, or may not, work on similar boards or direct clones.

6.5.1. Purpose

This driver allows the use of the Rapberry Pi GPIO pins in a way analogous to the parallel port driver on x86 PCs. It can use the same step generators, encoder counters and similar components.

6.5.2. Utilización

```
loadrt hal_pi_gpio dir=0x13407 exclude=0x1F64BF8
```

The ”dir” mask determines whether the pins are inputs and outputs, the exclude mask prevents the driver from using the pins (and so allows them to be used for their normal RPi purposes such as SPI or UART).

The mask can be in decimal or hexadecimal (hex may be easier as there will be no carries).

To determine the value of the masks, add up the hex/decimal values for all pins that should be configured as output, and analogously for all pins that should be excluded according to the following table.

Cuadro 6.29: GPIO masks - mapping of GPIO numbers (leftmost column) to physical pin numbers as printed on the Raspberry Pi board (rightmost column) and the decimal/hexadecimal values that contribute to the value of the mask.

GPIO Num	Decimal	Hex	Pin Num
2	1	0x00000001	3
3	2	0x00000002	5
4	4	0x00000004	7

Cuadro 6.29: (continued)

GPIO Num	Decimal	Hex	Pin Num
5	8	0x00000008	29
6	16	0x00000010	31
7	32	0x00000020	26
8	64	0x00000040	24
9	128	0x00000080	21
10	256	0x00000100	19
11	512	0x00000200	23
12	1024	0x00000400	32
13	2048	0x00000800	33
14	4096	0x00001000	8
15	8192	0x00002000	10
16	16384	0x00004000	36
17	32768	0x00008000	11
18	65536	0x00010000	12
19	131072	0x00020000	35
20	262144	0x00040000	38
21	524288	0x00080000	40
22	1048576	0x00100000	15
23	2097152	0x00200000	16
24	4194304	0x00400000	18
25	8388608	0x00800000	22
26	16777216	0x01000000	37
27	33554432	0x02000000	13

Note: In the calculation of the individual pin's mask value its GPIO numbers are used, the value being derived as 2^(*GPIO number* - 2), whereas in the naming of the HAL pins it is the Raspberry Pi header pin numbers.

So, for example, if you enable GPIO 17 as an output (dir=0x8000) then that output will be controlled by the hal pin **hal_pi_gpio.pin-11-out**.

6.5.3. Pines

- hal_pi_gpio.pin-*NN*-out
- hal_pi_gpio.pin-*NN*-in

Depending on the dir and exclude masks.

6.5.4. Parámetros

Only the standard timing parameters which are created for all components exist:

- hal_pi_gpio.read.tmax
- hal_pi_gpio.read.tmax-increased
- hal_pi_gpio.write.tmax
- hal_pi_gpio.write.tmax-increased

For unknown reasons the driver also creates HAL *pins* to indicate timing:

- hal_pi_gpio.read.time
- hal_pi_gpio.write.time

6.5.5. Funciones

- `hal_pi_gpio.read` - Add this to the base thread to update the HAL pin values to match the physical input values.
- `hal_pi_gpio.write` - Add this to the base thread to update the physical pins to match the HAL values.

Typically the *read* function will be early in the call list, before any encoder counters and the *write* function will be later in the call list, after stepgen.make-pulses.

6.5.6. Pin Numbering

The GPIO connector and the pinout has been consistent since around 2015. These older Pi models are probably a poor choice for LinuxCNC anyway. However, this driver is designed to work with them, and will detect and correctly configure for the two alternative pinouts.

The current pinout mapping between GPIO numbers and connector pin numbers is included in the table above.

Note that the config string uses GPIO numbers, but once the driver is loaded the HAL pin names refer to connector pin numbers.

This may be more logical than it first appears. When setting up you need to configure enough pins of each type, whilst avoiding overwriting any other functions that your system needs. Then once the driver is loaded, in the HAL layer you just want to know where to connect the wires for each HAL pin.

6.5.7. Known Bugs

At the moment (2023-07-16) this driver only seems to work on Raspbian as the generic Debian image does not set up the correct interfaces in /dev/gpiomem and restricts access to the /sys/mem interface.

6.6. Generic driver for any GPIO supported by gpiod.

This driver has been tested on the Raspberry Pi, and should also work on Banana Pi, BeagleBone, Pine64 (et al.) and other single board computers, and potentially on other platforms.

6.6.1. Purpose

This driver allows the use of GPIO pins in a way analogous to the parallel port driver on x86 PCs. It can use the same step generators, encoder counters and similar components.

6.6.2. Utilización

```
loadrt hal_gpio inputs=GPIO5,GPIO6,GPIO12,GPIO13,GPIO16,GPIO17,GPIO18,GPIO19 \
                                outputs=GPIO20,GPIO21,GPIO22,GPIO23,GPIO24,GPIO25,GPIO26, ↩
                                    GPIO27 \
                                invert=GPIO20,GPIO27 \
                                reset=GPIO21,GPIO22
```

This driver relies on the libgpiod-dev library and the gpiod package, which contains a number of utilities for configuring and querying GPIO. The GPIO pin names in the ”loadrt” line of the HAL given above should be the names given by the `gpioinfo` command.

Sample output (truncated):

```
$ gpioinfo
gpiochip0 - 54 lines:
        line   0:     ”ID_SDA”       unused   input  active-high
        line   1:     ”ID_SCL”       unused   input  active-high
        line   2:       ”SDA1”       unused   input  active-high
        line   3:       ”SCL1”       unused   input  active-high
        line   4:  ”GPIO_GCLK”       unused   input  active-high
        line   5:      ”GPIO5”       unused   input  active-high
        line   6:      ”GPIO6”       unused   input  active-high
        line   7:  ”SPI_CE1_N”       unused   input  active-high
        line   8:  ”SPI_CE0_N”       unused   input  active-high
        line   9:   ”SPI_MISO”       unused   input  active-high
        line  10:   ”SPI_MOSI”       unused   input  active-high
        line  11:   ”SPI_SCLK”       unused   input  active-high
        line  12:     ”GPIO12”       unused   input  active-high
        line  13:     ”GPIO13”       unused   input  active-high
        line  14:       ”TXD1”       unused   input  active-high
        line  15:       ”RXD1”       unused   input  active-high
        line  16:     ”GPIO16”       unused   input  active-high
        line  17:     ”GPIO17”       unused   input  active-high
        line  18:     ”GPIO18”       unused   input  active-high
        line  19:     ”GPIO19”       unused   input  active-high
        line  20:     ”GPIO20”       unused  output  active-high
...
```

A list of input and/or output pins should be specified as shown in the sample above. The \ character is used for line continuation in HAL, and is used to improve readability. The pin names are case-sensitive and there must be no spaces in the strings, neither between the comma-separated pins lists nor the ”=” signs.

Additional modifiers are

invert
(valid for outputs only). Inverts the sense of the physical pin relative to the value in HAL.

reinicio
(valid for outputs only). If any pins are allocated to the ”reset” list then a HAL parameter **hal_gpio.reset_ns** will be created. This will have no effect unless the **hal_gpio.reset** function is added to a realtime thread. This should be placed after the **hal_gpio.write** function and must be in the same thread. The behaviour of this function is equivalent to the same function in the **hal_parport** driver, and it allows a step pulse every thread cycle. If the **hal_gpio.reset_ns** time is set longer than 1/4 of the period of the thread that it is added to, then the value will be reduced to 1/4 the thread period. There is a lower limit to how long the pulse can be. With 8 pins in the output list the pulse width can not reduce lower than 5000 ns on an RPi4, for example.

The following functions are accepted in all versions, but are only effective if a version of libgpiod_dev >= 1.6 is installed. They should be used in the same way as the parameters described above, and will alter the electrical parameters of the GPIO pins **if** this is supported by the hardware.

opendrain

opensource

biasdisable

pulldown

pullup

The version of libgpiod-dev installed can be determined by the command `gpioinfo -v`

6.6.3. Pines

- hal_gpio.NAME-in - HAL_OUT The value of an input pin presented in to HAL
- hal_gpio.NAME-in-not - HAL_OUT An inverted version of the above, for convenience
- hal_gpio.NAME-out - HAL_IN use this pin to transfer a HAL bit value to a physical output

6.6.4. Parámetros

- hal_gpio.reset_ns - HAL_RW - ”setp” this parameter to control the pulse length of pins added to the ”reset” list. The value will be limited between 0 and thread-period / 4.

6.6.5. Funciones

- *hal_gpio.read* - Add this to the base thread to update the HAL pin values to match the physical input values.
- *hal_gpio.write* - Add this to the base thread to update the physical pins to match the HAL values.
- *hal_gpio.reset* - Only exported if there are pins defined in the reset list. This should be placed after the ”write” function, and should be in the same thread.

Typically, the *read* function will be early in the call list, before any encoder counters and the *write* function will be later in the call list, after `stepgen.make-pulses`.

6.6.6. Pin Identification

Use the pin names returned by the `gpioinfo` utility. This uses the device-tree data. If the installed OS does not have a device-tree database then the pins will all be called ”unnamed” (or similar) and this driver can not be used.

A further update to this driver might allow access by index number, but this is not currently supported.

6.6.7. Troubleshooting permissions problems.

If ”access denied” messages are returned on loading the driver, try the following recipe: (Should not be needed for Raspbian, and will need to be modified to match the actual GPIO chip name on non-Pi platforms)

1. Create a new group `gpio` with the command

```
sudo groupadd gpio
```

2. Then to setup permissions for the ”gpio” group, create a file called `90-gpio-access` in the `/etc/udev/rules.d/` directory with the following contents (this is copied from the Raspbian install)

```
SUBSYSTEM==”bcm2835-gpiomem”, GROUP=”gpio”, MODE=”0660”
SUBSYSTEM==”gpio”, GROUP=”gpio”, MODE=”0660”
SUBSYSTEM==”gpio*”, PROGRAM=”/bin/sh -c ’\
        chown -R root:gpio /sys/class/gpio && chmod -R 770 /sys/class/gpio;\
        chown -R root:gpio /sys/devices/virtual/gpio &&\
        chmod -R 770 /sys/devices/virtual/gpio;\
        chown -R root:gpio /sys$devpath && chmod -R 770 /sys$devpath\
’”

SUBSYSTEM==”pwm*”, PROGRAM=”/bin/sh -c ’\
        chown -R root:gpio /sys/class/pwm && chmod -R 770 /sys/class/pwm;\
        chown -R root:gpio /sys/devices/platform/soc/*.pwm/pwm/pwmchip* &&\
        chmod -R 770 /sys/devices/platform/soc/*.pwm/pwm/pwmchip*\
’”
```

3. Add the user who runs LinuxCNC to the `gpio` group with

```
sudo usermod -aG gpio <username>
```

6.6.8. Author

Andy Pugh

6.6.9. Known Bugs

Ninguno en este momento.

6.7. Mesa HostMot2 Driver

6.7.1. Introducción

HostMot2 is an FPGA configuration developed by Mesa Electronics for their line of *Anything I/O* motion control cards. The firmware is open source, portable and flexible. It can be configured (at compile-time) with zero or more instances (an object created at runtime) of each of several Modules: encoders (quadrature counters), PWM generators, and step/dir generators. The firmware can be configured (at run-time) to connect each of these instances to pins on the I/O headers. I/O pins not driven by a Module instance revert to general-purpose bi-directional digital I/O.

6.7.2. Firmware Binaries

50 Pin Header FPGA cards Several pre-compiled HostMot2 firmware binaries are available for the different Anything I/O boards. This list is incomplete, check the hostmot2-firmware distribution for up-to-date firmware lists.

- 3x20 (144 I/O pins): using hm2_pci module
  - 24-channel servo
  - 16-channel servo plus 24 step/dir generators
- 5I22 (96 I/O pins): using hm2_pci module
  - 16-channel servo
  - 8-channel servo plus 24 step/dir generators
- 5I20, 5I23, 4I65, 4I68 (72 I/O pins): using hm2_pci module
  - 12-channel servo
  - 8-channel servo plus 4 step/dir generators
  - 4-channel servo plus 8 step/dir generators
- 7I43 (48 I/O pins): using hm2_7i43 module
  - 8-channel servo (8 PWM generators & 8 encoders)
  - 4-channel servo plus 4 step/dir generators

DB25 FPGA cards The 5I25 Superport FPGA card is preprogrammed when purchased and does not need a firmware binary.

6.7.3. Installing Firmware

Depending on how you installed LinuxCNC you may have to open the Synaptic Package Manager from the System menu and install the package for your Mesa card. The quickest way to find them is to do a search for *hostmot2* in the Synaptic Package Manager. Mark the firmware for installation, then apply.

6.7.4. Loading HostMot2

The LinuxCNC support for the HostMot2 firmware is split into a generic driver called *hostmot2* and two low-level I/O drivers for the Anything I/O boards. The low-level I/O drivers are *hm2_7i43* and *hm2_pci* (for all the PCI- and PC-104/Plus-based Anything I/O boards). The hostmot2 driver must be loaded first, using a HAL command like this:

```
loadrt hostmot2
```

See the hostmot2(9) man page for details.

The hostmot2 driver by itself does nothing, it needs access to actual boards running the HostMot2 firmware. The low-level I/O drivers provide this access. The low-level I/O drivers are loaded with commands like this:

```
loadrt hm2_pci config="firmware=hm2/5i20/SVST8_4.BIT
       num_encoders=3 num_pwmgens=3 num_stepgens=1"
```

Los parámetros de configuración se describen en la página del manual de hostmot2.

6.7.5. Watchdog

The HostMot2 firmware may include a watchdog Module; if it does, the hostmot2 driver will use it.

The watchdog must be petted by LinuxCNC periodically or it will bite. The hm2 write function (see below) pets the watchdog.

When the watchdog bites, all the board's I/O pins are disconnected from their Module instances and become high-impedance inputs (pulled high). The state of the HostMot2 firmware modules is not disturbed (except for the configuration of the I/O Pins). Encoder instances keep counting quadrature pulses, and pwm- and step-generators keep generating signals (which are not relayed to the motors, because the I/O Pins have become inputs).

Resetting the watchdog resets the I/O pins to the configuration chosen at load-time.

If the firmware includes a watchdog, the following HAL objects will be exported:

6.7.5.1. Pines

- *has_bit* - (bit i/o) True if the watchdog has bit, False if the watchdog has not bit. If the watchdog has bit and the has_bit bit is True, the user can reset it to False to resume operation.

6.7.5.2. Parámetros

- *timeout_ns* - (u32 read/write) Watchdog timeout, in nanoseconds. This is initialized to 5,000,000 (5 milliseconds) at module load time. If more than this amount of time passes between calls to the hm2 write function, the watchdog will bite.

6.7.6. HostMot2 Functions

- *hm2_<BoardType>.<BoardNum>.read* - Read all inputs, update input HAL pins.
- *hm2_<BoardType>.<BoardNum>.write* - Write all outputs.
- *hm2_<BoardType>.<BoardNum>.read_gpio* - Read the GPIO input pins only. (This function is not available on the 7I43 due to limitations of the EPP bus.)
- *hm2_<BoardType>.<BoardNum>.write_gpio* - Write the GPIO control registers and output pins only. (This function is not available on the 7I43 due to limitations of the EPP bus.)

---

nota
The above *read_gpio* and *write_gpio* functions should not normally be needed, since the GPIO bits are read and written along with everything else in the standard *read* and *write* functions above, which are normally run in the servo thread.
The *read_gpio* and *write_gpio* functions were provided in case some very fast (frequently updated) I/O is needed. These functions should be run in the base thread. If you have need for this, please send an email and tell us about it, and what your application is.

6.7.7. Pinouts

The hostmot2 driver does not have a particular pinout. The pinout comes from the firmware that the hostmot2 driver sends to the Anything I/O board. Each firmware has different pinout, and the pinout depends on how many of the available encoders, pwmgens, and stepgens are used. To get a pinout list for your configuration after loading LinuxCNC in the terminal window type:

```
dmesg > hm2.txt
```

The resulting text file will contain lots of information as well as the pinout for the HostMot2 and any error and warning messages.

To reduce the clutter by clearing the message buffer before loading LinuxCNC type the following in the terminal window:

```
sudo dmesg -c
```

Now when you run LinuxCNC and then do a *dmesg > hm2.txt* in the terminal only the info from the time you loaded LinuxCNC will be in your file along with your pinout. The file will be in the current directory of the terminal window. Each line will contain the card name, the card number, the I/O Pin number, the connector and pin, and the usage. From this printout you will know the physical connections to your card based on your configuration.

Un ejemplo de una configuración 5120:

```
[HOSTMOT2]
DRIVER=hm2_pci
BOARD=5i20
CONFIG="firmware=hm2/5i20/SVST8_4.BIT num_encoders=1 num_pwmgens=1 num_stepgens=3"
```

The above configuration produced this printout.

```
[ 1141.053386] hm2/hm2_5i20.0: 72 I/O Pins used:
[ 1141.053394] hm2/hm2_5i20.0: IO Pin 000 (P2-01): IOPort
[ 1141.053397] hm2/hm2_5i20.0: IO Pin 001 (P2-03): IOPort
[ 1141.053401] hm2/hm2_5i20.0: IO Pin 002 (P2-05): Encoder #0, pin B (Input)
[ 1141.053405] hm2/hm2_5i20.0: IO Pin 003 (P2-07): Encoder #0, pin A (Input)
[ 1141.053408] hm2/hm2_5i20.0: IO Pin 004 (P2-09): IOPort
[ 1141.053411] hm2/hm2_5i20.0: IO Pin 005 (P2-11): Encoder #0, pin Index (Input)
[ 1141.053415] hm2/hm2_5i20.0: IO Pin 006 (P2-13): IOPort
[ 1141.053418] hm2/hm2_5i20.0: IO Pin 007 (P2-15): PWMGen #0, pin Out0 (PWM or Up) (Output)
[ 1141.053422] hm2/hm2_5i20.0: IO Pin 008 (P2-17): IOPort
[ 1141.053425] hm2/hm2_5i20.0: IO Pin 009 (P2-19): PWMGen #0, pin Out1 (Dir or Down) ( ↩
    Output)
[ 1141.053429] hm2/hm2_5i20.0: IO Pin 010 (P2-21): IOPort
[ 1141.053432] hm2/hm2_5i20.0: IO Pin 011 (P2-23): PWMGen #0, pin Not-Enable (Output)
<snip>...
[ 1141.053589] hm2/hm2_5i20.0: IO Pin 060 (P4-25): StepGen #2, pin Step (Output)
[ 1141.053593] hm2/hm2_5i20.0: IO Pin 061 (P4-27): StepGen #2, pin Direction (Output)
[ 1141.053597] hm2/hm2_5i20.0: IO Pin 062 (P4-29): StepGen #2, pin (unused) (Output)
[ 1141.053601] hm2/hm2_5i20.0: IO Pin 063 (P4-31): StepGen #2, pin (unused) (Output)
[ 1141.053605] hm2/hm2_5i20.0: IO Pin 064 (P4-33): StepGen #2, pin (unused) (Output)
[ 1141.053609] hm2/hm2_5i20.0: IO Pin 065 (P4-35): StepGen #2, pin (unused) (Output)
[ 1141.053613] hm2/hm2_5i20.0: IO Pin 066 (P4-37): IOPort
[ 1141.053616] hm2/hm2_5i20.0: IO Pin 067 (P4-39): IOPort
[ 1141.053619] hm2/hm2_5i20.0: IO Pin 068 (P4-41): IOPort
[ 1141.053621] hm2/hm2_5i20.0: IO Pin 069 (P4-43): IOPort
[ 1141.053624] hm2/hm2_5i20.0: IO Pin 070 (P4-45): IOPort
[ 1141.053627] hm2/hm2_5i20.0: IO Pin 071 (P4-47): IOPort
[ 1141.053811] hm2/hm2_5i20.0: registered
[ 1141.053815] hm2_5i20.0: initialized AnyIO board at 0000:02:02.0
```

nota
That the I/O Pin nnn will correspond to the pin number shown on the HAL Configuration screen for GPIOs. Some of the StepGen, Encoder and PWMGen will also show up as GPIOs in the HAL Configuration screen.

6.7.8. PIN Files

The default pinout is described in a .PIN file (human-readable text). When you install a firmware package the .PIN files are installed in

```
/usr/share/doc/hostmot2-firmware-<board>/
```

6.7.9. Firmware

The selected firmware (.BIT file) and configuration is uploaded from the PC motherboard to the Mesa mothercard on LinuxCNC startup. If you are using Run In Place, you must still install a hostmot2-firmware-<board> package. There is more information about firmware and configuration in the *Configurations* section.

6.7.10. HAL Pins

The HAL pins for each configuration can be seen by opening up *Show HAL Configuration* from the Machine menu. All the HAL pins and parameters can be found there. The following figure is of the 5I20 configuration used above.

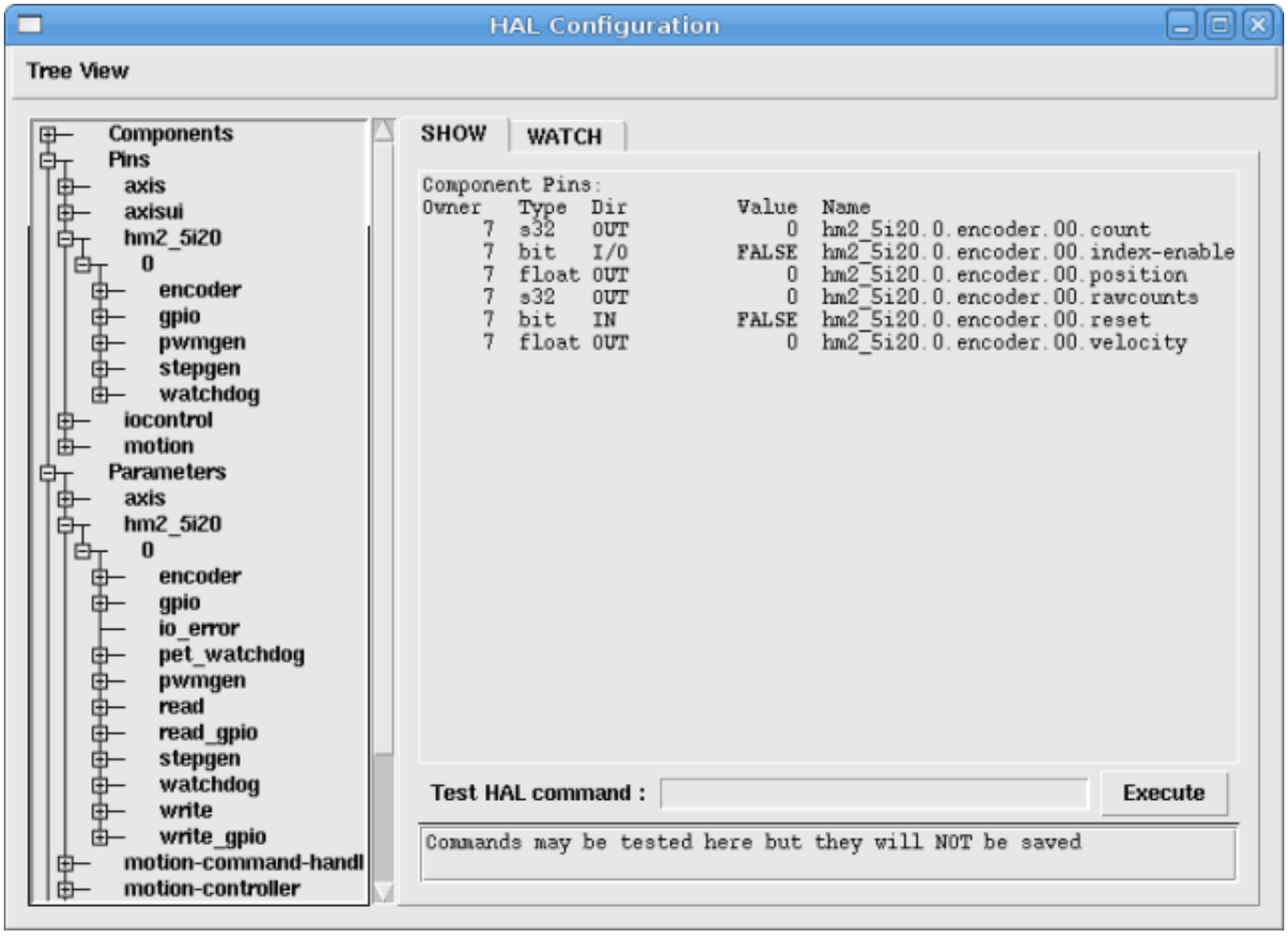


Figura 6.9: 5i20 HAL Pins

6.7.11. Configurations

The Hostmot2 firmware is available in several versions, depending on what you are trying to accomplish. You can get a reminder of what a particular firmware is for by looking at the name. Let's look at a couple of examples.

In the 7I43 (two ports), SV8 (*Servo 8*) would be for having 8 servos or fewer, using the *classic* 7I33 4-axis (per port) servo board. So 8 servos would use up all 48 signals in the two ports. But if you only needed 3 servos, you could say *num_encoders=3* and *num_pwmgens=3* and recover 5 servos at 6 signals each, thus gaining 30 bits of GPIO.

Or, in the 5I22 (four ports), SVST8_24 (*Servo 8, Stepper 24*) would be for having 8 servos or fewer (7I33 x2 again), and 24 steppers or fewer (7I47 x2). This would use up all four ports. If you only needed 4 servos you could say *num_encoders=4* and *num_pwmgens=4* and recover 1 port (and save a 7I33). And if you only needed 12 steppers you could say *num_stepgens=12* and free up one port (and save a 7I47). So in this way we can save two ports (48 bits) for GPIO.

Here are tables of the firmwares available in the official packages. There may be additional firmwares available at the Mesanet.com website that have not yet made it into the LinuxCNC official firmware packages, so check there too.

3x20 (6-port various) Default Configurations (The 3x20 comes in 1M, 1.5M, and 2M gate versions. So far, all firmware is available in all gate sizes.)

Firmware	Codificador	PWMGen	StepGen	GPIO
SV24	24	24	0	0
SVST16_24	16	16	24	0

5I22 (4-port PCI) Default Configurations (The 5I22 comes in 1M and 1.5M gate versions. So far, all firmware is available in all gate sizes.)

Firmware	Codificador	PWM	StepGen	GPIO
SV16	16	16	0	0
SVST2_4_7I47	4	2	4	72
SVST8_8	8	8	8	0
SVST8_24	8	8	24	0

5I23 (3-port PCI) Default Configurations (The 5I23 has 400k gates.)

Firmware	Codificador	PWM	StepGen	GPIO
SV12	12	12	0	0
SVST2_8	2	2	8 (tbl5)	12
SVST2_4_7I47	4	2	4	48
SV12_2X7I48_72	12	12	0	24
SV12IM_2X7I48_72	12 (+IM)	12	0	12
SVST4_8	4	4	8 (tbl5)	0
SVST8_4	8	8	4 (tbl5)	0
SVST8_4IM2	8 (+IM)	8	4	8
SVST8_8IM2	8 (+IM)	8	8	0
SVTP6_7I39	6	0 (6 BLDC)	0	0

5I20 (3-port PCI) Default Configurations (The 5I20 has 200k gates.)

Firmware	Codificador	PWM	StepGen	GPIO
SV12	12	12	0	0
SVST2_8	2	2	8 (tbl5)	12

Firmware	Codificador	PWM	StepGen	GPIO
SVST2_4_7I47	4	2	4	48
SV12_2X7I48_72	12	12	0	24
SV12IM_2X7I48_72	12 (+IM)	12	0	12
SVST8_4	8	8	4 (tbl5)	0
SVST8_4IM2	8 (+IM)	8	4	8

4I68 (3-port PC/104) Default Configurations (The 4I68 has 400k gates.)

Firmware	Codificador	PWM	StepGen	GPIO
SV12	12	12	0	0
SVST2_4_7I47	4	2	4	48
SVST4_8	4	4	8	0
SVST8_4	8	8	4	0
SVST8_4IM2	8 (+IM)	8	4	8
SVST8_8IM2	8 (+IM)	8	8	0

4I65 (3-port PC/104) Default Configurations (The 4I65 has 200k gates.)

Firmware	Codificador	PWM	StepGen	GPIO
SV12	12	12	0	0
SVST8_4	8	8	4	0
SVST8_4IM2	8 (+IM)	8	4	8

7I43 (2-port parallel) 400k gate versions, Default Configurations

Firmware	Codificador	PWM	StepGen	GPIO
SV8	8	8	0	0
SVST4_4	4	4	4 (tbl5)	0
SVST4_6	4	4	6 (tbl3)	0
SVST4_12	4	4	12	0
SVST2_4_7I47	4	2	4	24

7I43 (2-port parallel) 200k gate versions, Default Configurations

Firmware	Codificador	PWM	StepGen	GPIO
SV8	8	8	0	0
SVST4_4	4	4	4 (tbl5)	0
SVST4_6	4	4	6 (tbl3)	0
SVST2_4_7I47	4	2	4	24

Even though several cards may have the same named .BIT file you cannot use a .BIT file that is not for that card. Different cards have different clock frequencies so make sure you load the proper .BIT file for your card. Custom hm2 firmwares can be created for special applications and you may see some custom hm2 firmwares in the directories with the default ones.

When you load the board-driver (hm2_pci or hm2_7i43), you can tell it to disable instances of the three primary modules (pwmgen, stepgen, and encoder) by setting the count lower. Any I/O pins belonging to disabled module instances become GPIOs.

6.7.12. GPIO

General Purpose I/O pins on the board which are not used by a module instance are exported to HAL as *full* GPIO pins. Full GPIO pins can be configured at run-time to be inputs, outputs, or open drains, and have a HAL interface that exposes this flexibility. I/O pins that are owned by an active module instance are constrained by the requirements of the owning module, and have a restricted HAL interface.

GPIOs have names like *hm2_<BoardType>.<BoardNum>.gpio.<IONum>*. IONum is a three-digit number. The mapping from IONum to connector and pin-on-that-connector is written to the syslog when the driver loads, and it's documented in Mesa's manual for the Anything I/O boards.

The hm2 GPIO representation is modeled after the Digital Inputs and Digital Outputs described in the Canonical Device Interface (part of the HAL General Reference document).

GPIO pins default to input.

6.7.12.1. Pines

- *in* - (Bit, Out) Normal state of the hardware input pin. Both full GPIO pins and I/O pins used as inputs by active module instances have this pin.
- *in_not* - (Bit, Out) Inverted state of the hardware input pin. Both full GPIO pins and I/O pins used as inputs by active module instances have this pin.
- *out* - (Bit, In) Value to be written (possibly inverted) to the hardware output pin. Only full GPIO pins have this pin.

6.7.12.2. Parámetros

- *invert_output* - (Bit, RW) This parameter only has an effect if the *is_output* parameter is true. If this parameter is true, the output value of the GPIO will be the inverse of the value on the *out* HAL pin. Only full GPIO pins and I/O pins used as outputs by active module instances have this parameter. To invert an active module pin you have to invert the GPIO pin not the module pin.
- *is_opendrain* - (Bit, RW) This parameter only has an effect if the *is_output* parameter is true. If this parameter is false, the GPIO behaves as a normal output pin: the I/O pin on the connector is driven to the value specified by the *out* HAL pin (possibly inverted), and the value of the *in* and *in_not* HAL pins is undefined. If this parameter is true, the GPIO behaves as an open-drain pin. Writing 0 to the *out* HAL pin drives the I/O pin low, writing 1 to the *out* HAL pin puts the I/O pin in a high-impedance state. In this high-impedance state the I/O pin floats (weakly pulled high), and other devices can drive the value; the resulting value on the I/O pin is available on the *in* and *in_not* pins. Only full GPIO pins and I/O pins used as outputs by active module instances have this parameter.
- *is_output* - (Bit, RW) If set to 0, the GPIO is an input. The I/O pin is put in a high-impedance state (weakly pulled high), to be driven by other devices. The logic value on the I/O pin is available in the *in* and *in_not* HAL pins. Writes to the *out* HAL pin have no effect. If this parameter is set to 1, the GPIO is an output; its behavior then depends on the *is_opendrain* parameter. Only full GPIO pins have this parameter.

6.7.13. StepGen

StepGens have names like *hm2_<BoardType>.<BoardNum>.stepgen.<Instance>*. *Instance* is a two-digit number that corresponds to the HostMot2 stepgen instance number. There are *num_stepgens* instances, starting with 00.

Each stepgen allocates 2-6 I/O pins (selected at firmware compile time), but currently only uses two: Step and Direction outputs. [3]

The StepGen representation is modeled on the stepgen software component. StepGen default is active high step output (high during step time low during step space). To invert a StepGen output pin you invert the corresponding GPIO pin that is being used by StepGen. To find the GPIO pin being used for the StepGen output run *dmesg* as shown above.

Cada instancia StepGen tienen los siguientes pines y parámetros:

6.7.13.1. Pines

- *control-type* - (Bit, In) Switches between position control mode (0) and velocity control mode (1). Defaults to position control (0).
- *counts* - (s32, Out) Feedback position in counts (number of steps).
- *enable* - (Bit, In) Habilita la salida de pasos. Cuando es falso, no se generan pasos.
- *position-cmd* - (Float, In) Target position of stepper motion, in user-defined position units.
- *position-fb* - (Float, Out) Feedback position in user-defined position units (counts / position_scale).
- *velocity-cmd* - (Float, In) Target velocity of stepper motion, in user-defined position units per second. This pin is only used when the stepgen is in velocity control mode (control-type=1).
- *velocity-fb* - (Float, Out) Feedback velocity in user-defined position units per second.

6.7.13.2. Parámetros

- *dirhold* - (u32, RW) Minimum duration of stable Direction signal after a step ends, in nanoseconds.
- *dirsetup* - (u32, RW) Minimum duration of stable Direction signal before a step begins, in nanoseconds.
- *maxaccel* - (Float, RW) Maximum acceleration, in position units per second per second. If set to 0, the driver will not limit its acceleration.
- *maxvel* - (Float, RW) Maximum speed, in position units per second. If set to 0, the driver will choose the maximum velocity based on the values of steplen and stepspace (at the time that maxvel was set to 0).
- *position-scale* - (Float, RW) Converts from counts to position units. position = counts / position_scale
- *step_type* - (u32, RW) Output format, like the step_type modparam to the software stegen(9) component. 0 = Step/Dir, 1 = Up/Down, 2 = Quadrature. In Quadrature mode (step_type=2), the stepgen outputs one complete Gray cycle (00 -> 01 -> 11 -> 10 -> 00) for each *step* it takes.
- *steplen* - (u32, RW) Duration of the step signal, in nanoseconds.
- *stepspace* - (u32, RW) Minimum interval between step signals, in nanoseconds.

[3] At present, the firmware supports multi-phase stepper outputs, but the driver doesn't. Interested volunteers are solicited.

6.7.13.3. Output Parameters

The Step and Direction pins of each StepGen have two additional parameters. To find which I/O pin belongs to which step and direction output run *dmesg* as described above.

- *invert_output* - (Bit, RW) This parameter only has an effect if the *is_output* parameter is true. If this parameter is true, the output value of the GPIO will be the inverse of the value on the *out* HAL pin.
- *is_opendrain* - (Bit, RW) If this parameter is false, the GPIO behaves as a normal output pin: the I/O pin on the connector is driven to the value specified by the *out* HAL pin (possibly inverted). If this parameter is true, the GPIO behaves as an open-drain pin. Writing 0 to the *out* HAL pin drives the I/O pin low, writing 1 to the *out* HAL pin puts the I/O pin in a high-impedance state. In this high-impedance state the I/O pin floats (weakly pulled high), and other devices can drive the value; the resulting value on the I/O pin is available on the *in* and *in_not* pins. Only full GPIO pins and I/O pins used as outputs by active module instances have this parameter.

6.7.14. PWMGen

PWMgens have names like *hm2_<BoardType>.<BoardNum>.pwmgen.<Instance>*. *Instance* is a two-digit number that corresponds to the HostMot2 pwmgen instance number. There are *num_pwmgens* instances, starting with 00.

In HM2, each pwmgen uses three output I/O pins: Not-Enable, Out0, and Out1. To invert a PWMGen output pin you invert the corresponding GPIO pin that is being used by PWMGen. To find the GPIO pin being used for the PWMGen output run *dmesg* as shown above.

The function of the Out0 and Out1 I/O pins varies with output-type parameter (see below).

The hm2 pwmgen representation is similar to the software pwmgen component. Each pwmgen instance has the following pins and parameters:

6.7.14.1. Pines

- *enable* - (Bit, In) If true, the pwmgen will set its Not-Enable pin false and output its pulses. If *enable* is false, pwmgen will set its Not-Enable pin true and not output any signals.
- *value* - (Float, In) The current pwmgen command value, in arbitrary units.

6.7.14.2. Parámetros

- *output-type* - (s32, RW) This emulates the output_type load-time argument to the software pwmgen component. This parameter may be changed at runtime, but most of the time you probably want to set it at startup and then leave it alone. Accepted values are 1 (PWM on Out0 and Direction on Out1), 2 (Up on Out0 and Down on Out1), 3 (PDM mode, PDM on Out0 and Dir on Out1), and 4 (Direction on Out0 and PWM on Out1, *for locked antiphase*).
- *scale* - (Float, RW) Scaling factor to convert *value* from arbitrary units to duty cycle: dc = value / scale. Duty cycle has an effective range of -1.0 to +1.0 inclusive, anything outside that range gets clipped.
- *pdm_frequency* - (u32, RW) This specifies the PDM frequency, in Hz, of all the pwmgen instances running in PDM mode (mode 3). This is the *pulse slot frequency*; the frequency at which the pdm generator in the Anything I/O board chooses whether to emit a pulse or a space. Each pulse (and space) in the PDM pulse train has a duration of 1/pdm_frequency seconds. For example, setting the pdm_frequency to $2*10^6$ Hz (2 MHz) and the duty cycle to 50 % results in a 1 MHz square wave, identical to a 1 MHz PWM signal with 50 % duty cycle. The effective range of this parameter is

from about 1525 Hz up to just under 100 MHz. Note that the max frequency is determined by the ClockHigh frequency of the Anything I/O board; the 5I20 and 7I43 both have a 100 MHz clock, resulting in a 100 MHz max PDM frequency. Other boards may have different clocks, resulting in different max PDM frequencies. If the user attempts to set the frequency too high, then it will be clipped to the max supported frequency of the board.

- *pwm_frequency* - (u32, RW) This specifies the PWM frequency, in Hz, of all the pwmgen instances running in the PWM modes (modes 1 and 2). This is the frequency of the variable-duty-cycle wave. Its effective range is from 1 Hz up to 193 kHz. Note that the max frequency is determined by the ClockHigh frequency of the Anything I/O board; the 5i20 and 7i43 both have a 100 MHz clock, resulting in a 193 kHz max PWM frequency. Other boards may have different clocks, resulting in different max PWM frequencies. If the user attempts to set the frequency too high, then it will be clipped to the max supported frequency of the board. Frequencies below about 5 Hz are not terribly accurate, but above 5 Hz they are pretty close.

6.7.14.3. Output Parameters

The output pins of each PWMGen have two additional parameters. To find which I/O pin belongs to which output run `dmesg` as described above.

- `invert_output` - (Bit, RW) This parameter only has an effect if the `is_output` parameter is true. If this parameter is true, the output value of the GPIO will be the inverse of the value on the `out` HAL pin.
- `is_opendrain` - (Bit, RW) If this parameter is false, the GPIO behaves as a normal output pin: The I/O pin on the connector is driven to the value specified by the `out` HAL pin (possibly inverted). If this parameter is true, the GPIO behaves as an open-drain pin. Writing 0 to the `out` HAL pin drives the I/O pin low, writing 1 to the `out` HAL pin puts the I/O pin in a high-impedance state. In this high-impedance state the I/O pin floats (weakly pulled high), and other devices can drive the value; the resulting value on the I/O pin is available on the `in` and `in_not` pins. Only full GPIO pins and I/O pins used as outputs by active module instances have this parameter.

6.7.15. Codificador

Encoders have names like `hm2_<BoardType>.<BoardNum>.encoder.<Instance>.`. `Instance` is a two-digit number that corresponds to the HostMot2 encoder instance number. There are *num_encoders* instances, starting with 00.

Each encoder uses three or four input I/O pins, depending on how the firmware was compiled. Three-pin encoders use A, B, and Index (sometimes also known as Z). Four-pin encoders use A, B, Index, and Index-mask.

The hm2 encoder representation is similar to the one described by the Canonical Device Interface (in the HAL General Reference document), and to the software encoder component. Each encoder instance has the following pins and parameters:

6.7.15.1. Pines

- `count` - (s32, Out) Number of encoder counts since the previous reset.
- `index-enable` - (Bit, I/O) When this pin is set to True, the count (and therefore also position) are reset to zero on the next Index (Phase-Z) pulse. At the same time, index-enable is reset to zero to indicate that the pulse has occurred.
- `position` - (Float, Out) Encoder position in position units (count / scale).

- `rawcounts` - (s32, Out) Total number of encoder counts since the start, not adjusted for index or reset.
- `reset` - (Bit, In) When this pin is TRUE, the count and position pins are set to 0. The value of the velocity pin is not affected by this. The driver does not reset this pin to FALSE after resetting the count to 0, that is the user's job.
- `velocity` - (Float, Out) Estimated encoder velocity in position units per second.

6.7.15.2. Parámetros

- `counter-mode` - (Bit, RW) Set to False (the default) for Quadrature. Set to True for Up/Down or for single input on Phase A. Can be used for a frequency to velocity converter with a single input on Phase A when set to true.
- `filter` - (Bit, RW) If set to True (the default), the quadrature counter needs 15 clocks to register a change on any of the three input lines (any pulse shorter than this is rejected as noise). If set to False, the quadrature counter needs only 3 clocks to register a change. The encoder sample clock runs at 33 MHz on the PCI Anything I/O cards and 50 MHz on the 7I43.
- `index-invert` - (Bit, RW) If set to True, the rising edge of the Index input pin triggers the Index event (if index-enable is True). If set to False, the falling edge triggers.
- `index-mask` - (Bit, RW) If set to True, the Index input pin only has an effect if the Index-Mask input pin is True (or False, depending on the `index-mask-invert` pin below).
- `index-mask-invert` - (Bit, RW) If set to True, Index-Mask must be False for Index to have an effect. If set to False, the `Index-Mask` pin must be True.
- `scale` - (Float, RW) Converts from *count* units to *position* units. A quadrature encoder will normally have 4 counts per pulse so a 100 PPR encoder would be 400 counts per revolution. In `.counter-mode` a 100 PPR encoder would have 100 counts per revolution as it only uses the rising edge of A and direction is B.
- `vel-timeout` - (Float, RW) When the encoder is moving slower than one pulse for each time that the driver reads the count from the FPGA (in the hm2_read() function), the velocity is harder to estimate. The driver can wait several iterations for the next pulse to arrive, all the while reporting the upper bound of the encoder velocity, which can be accurately guessed. This parameter specifies how long to wait for the next pulse, before reporting the encoder stopped. This parameter is in seconds.

6.7.16. 5I25 Configuration

6.7.16.1. Firmware

The 5I25 firmware comes preloaded for the daughter card it is purchased with. So the `firmware=xxx.BIT` is not part of the hm2_pci configuration string when using a 5I25.

6.7.16.2. Configuración

Example configurations of the 5I25/7I76 and 5I25/7I77 cards are included in the Configuration Selector.

If you like to roll your own configuration the following examples show how to load the drivers in the HAL file.

5I25 + 7I76 Card

```
# load the generic driver
loadrt hostmot2

# load the PCI driver and configure
loadrt hm2_pci config="num_encoders=1 num_stepgens=5 sserial_port_0=0XXX"
```

5I25 + 7I77 Card

```
# load the generic driver
loadrt hostmot2

# load the PCI driver and configure
loadrt hm2_pci config="num_encoders=6 num_pwmgens=6 sserial_port_0=0XXX"
```

6.7.16.3. SSERIAL Configuration

The `sserial_port_0=0XXX` configuration string sets some options for the smart serial daughter card. These options are specific for each daughter card. See the Mesa manual for more information on the exact usage (typically in the section called SOFTWARE PROCESS DATA MODES) or see the manual page of SSERIAL(9).

6.7.16.4. 7I77 Limits

The minlimit and maxlimit are bounds on the pin value (in this case the analog out value) fullscalemax is the scale factor.

These are by default set to the analog in or analog range (most likely in Volts).

So for example on the 7I77 +-10 V analog outputs, the default values are:

minlimit: -10
maxlimit: +10
maxfullscale: 10

If you wanted to say scale the analog out of a channel to IPS for a velocity mode servo (say 24 IPS max) you could set the limits like this:

minlimit: -24
maxlimit: +24
maxfullscale: 24

If you wanted to scale the analog out of a channel to RPM for a 0 to 6000 RPM spindle with 0-10 V control you could set the limits like this:

minlimit: 0
maxlimit: 6000
maxfullscale: 6000
(this would prevent unwanted negative output voltages from being set)

6.7.17. Example Configurations

Several example configurations for Mesa hardware are included with LinuxCNC. The configurations are located in the hm2-servo and hm2-stepper sections of the Configuration Selector. Typically you will need the board installed for the configuration you pick to load. The examples are a good place to start and will save you time. Just pick the proper example from the LinuxCNC Configuration Selector

and save a copy to your computer so you can edit it. To see the exact pins and parameters that your configuration gave you, open the Show HAL Configuration window from the Machine menu, or do `dmesg` as outlined above.

6.8. MB2HAL

6.8.1. Introducción

MB2HAL is a generic non-realtime HAL component to communicate with one or more Modbus devices. So far, there are two options to communicate with a Modbus device:

1. One option is to create a HAL component as a driver see VFD Modbus.
2. Another option is to use Classic Ladder which has Modbus built in, see ClassicLadder.
3. Now there is a third option that consists of a ”generic” driver configured by text file, this is called MB2HAL.

Why MB2HAL? Consider using MB2HAL if:

- You have to write a new driver and you don’t know anything about programming.
- You need to use Classic Ladder ”only” to manage the Modbus connections.
- You have to discover and configure first time the Modbus transactions. MB2HAL have debug levels to facilitate the low level protocol debug.
- You have more than one device to connect. MB2HAL is very efficiently managing multiple devices, transactions and links. Currently I am monitoring two axis drivers using a Rs232 port, a VFD driver using another Rs232 port, and a remote I/O using TCP/IP.
- You want a protocol to connect your Arduino to HAL. Look the included sample configuration file, sketch and library for Arduino Modbus.

6.8.2. Utilización

a. Create a config file from the example below

   1. Set component name (optional)
      Set `HAL_MODULE_NAME=mymodule` (default `HAL_MODULE_NAME=mb2hal`)
   2. Carga el componente HAL de tiempo no-real modbus

b. Default component name: `loadusr -W mb2hal config=config_file.ini`

c. Custom component name: `loadusr -Wn mymodule mb2hal config=config_file.ini`

6.8.3. Opciones

6.8.3.1. Init Section

`[MB2HAL_INIT]`

Valor	Tipo	Required	Descripción
INIT_DEBUG	Integer	No	Debug level of init and INI file parsing. 0 = silent 1 = error messages (default) 2 = OK confirmation messages 3 = debugging messages 4 = maximum debugging messages (only in transactions)
VERSION	String	No	Version number in the format N.N[NN]. Defaults to 1.0.
HAL_MODULE_NAME	String	No	HAL module (component) name. Defaults to ”mb2hal”.
SLOWDOWN	Float	No	Insert a delay of ”FLOAT seconds” between transactions in order to not to have a lot of logging and facilitate the debugging. Useful when using DEBUG=3 (NOT INIT_DEBUG=3). It affects ALL transactions. Use ”0.0” for normal activity.
TOTAL_TRANSACTIONS	Integer	Yes	The number of total Modbus transactions. There is no maximum.

6.8.3.2. Transaction Sections

One transaction section is required per transaction, starting at [TRANSACTION_00] and counting up sequentially. If there is a new link (not transaction), you must provide the REQUIRED parameters 1st time. Warning: Any OPTIONAL parameter not specified are copied from the previous transaction.

Valor	Tipo	Required	Descripción
LINK_TYPE	String	Yes	You must specify either a ”serial” or ”tcp” link for the first transaction. Later transactions will use the previous transaction link if not specified.
TCP_IP	IP address	If LINK_TYPE=tcp	The Modbus slave device IP address. Ignored if LINK_TYPE=serial.
TCP_PORT	Integer	No	The Modbus slave device TCP port. Defaults to 502. Ignored if LINK_TYPE=serial.
SERIAL_PORT	String	If LINK_TYPE=serial	The serial port. For example ”/dev/ttyS0”. Ignored if LINK_TYPE=tcp.
SERIAL_BAUD	Integer	If LINK_TYPE=serial	The baud rate. Ignored if LINK_TYPE=tcp.
SERIAL_BITS	Integer	If LINK_TYPE=serial	Data bits. One of 5, 6, 7, 8. Ignored if LINK_TYPE=tcp.
SERIAL_PARITY	String	If LINK_TYPE=serial	Data parity. One of: even, odd, none. Ignored if LINK_TYPE=tcp.
SERIAL_STOP	Integer	If LINK_TYPE=serial	Stop bits. One of 1, 2. Ignored if LINK_TYPE=tcp.
SERIAL_DELAY_MS	Integer	If LINK_TYPE=serial	Serial port delay between transactions of this section only. In ms. Defaults to 0. Ignored if LINK_TYPE=tcp.
MB_SLAVE_ID	Integer	Yes	Modbus slave number.
FIRST_ELEMENT	Integer	Yes	The first element address.
NELEMENTS	Integer	Unless PIN_NAMES is specified	The number of elements. It is an error to specify both NELEMENTS and PIN_NAMES. The pin names will be sequential numbers, e.g. mb2hal.plcin.01.
PIN_NAMES	List	Unless NELEMENTS is specified	A list of element names. These names will be used for the pin names, e.g. mb2hal.plcin.cycle_start. **NOTE:** There must be no white space characters in the list. Example: PIN_NAMES=cycle_start,stop,feed_hold

Valor	Tipo	Required	Descripción
MB_TX_CODE	String	Yes	Modbus transaction function code (see specifications): • fnct_01_read_coils • fnct_02_read_discrete_inputs • fnct_03_read_holding_registers • fnct_04_read_input_registers • fnct_05_write_single_coil • fnct_06_write_single_register • fnct_15_write_multiple_coils • fnct_16_write_multiple_registers
MB_RESPONSE_TIMEOUT_MS	Integer	No	Response timeout for this transaction. In ms. Defaults to 500 ms. This is how much to wait for 1st byte before raise an error.
MB_BYTE_TIMEOUT_MS	Integer	No	Byte timeout for this transaction. In ms. Defaults to 500 ms. This is how much to wait from byte to byte before raise an error.
HAL_TX_NAME	String	No	Instead of giving the transaction number, use a name. Example: `mb2hal.00.01` could become `mb2hal.plcin.01`. The name must not exceed 28 characters. **NOTE:** when using names be careful that you don't end up with two transactions using the same name.
MAX_UPDATE_RATE	Float	No	Maximum update rate in Hz. Defaults to 0.0 (0.0 = as soon as available = infinite). **NOTE:** This is a maximum rate and the actual rate may be lower. If you want to calculate it in ms use (1000 / required_ms). Example: 100 ms = `MAX_UPDATE_RATE=10.0`, because 1000.0 ms / 100.0 ms = 10.0 Hz.
DEBUG	String	No	Debug level for this transaction only. See `INIT_DEBUG` parameter above.

6.8.3.3. Error codes

While debugging transactions, note the returned "`ret[]`" value correspond to:

Modbus protocol exceptions:

- 0x01 - ILLEGAL_FUNCTION - the FUNCTION code received in the query is not allowed or invalid.
- 0x02 - ILLEGAL_DATA_ADDRESS - the DATA ADDRESS received in the query is not an allowable address for the slave or is invalid.
- 0x03 - ILLEGAL_DATA_VALUE - a VALUE contained in the data query field is not an allowable value or is invalid.
- 0x04 - SLAVE_DEVICE_FAILURE - SLAVE (or MASTER) device unrecoverable FAILURE while attempting to perform the requested action.
- 0x04 - SERVER_FAILURE - (see above).
- 0x05 - ACKNOWLEDGE - This response is returned to PREVENT A TIMEOUT in the master. A long duration of time is required to process the request in the slave.
- 0x06 - SLAVE_DEVICE_BUSY - The slave (or server) is BUSY. Retransmit the request later.
- 0x06 - SERVER_BUSY - (see above).
- 0x07 - NEGATIVE_ACKNOWLEDGE - Unsuccessful programming request using function code 13 or 14.

- 0x08 - MEMORY_PARITY_ERROR - SLAVE parity error in MEMORY.
- 0x0A (-10) - GATEWAY_PROBLEM_PATH - Gateway path(s) not available.
- 0x0B (-11) - GATEWAY_PROBLEM_TARGET - The target device failed to respond (generated by master, not slave).

Programa o conexión:

- 0x0C (-12) - COMM_TIME_OUT
- 0x0D (-13) - PORT_SOCKET_FAILURE
- 0x0E (-14) - SELECT_FAILURE
- 0x0F (-15) - TOO_MANY_DATAS
- 0x10 (-16) - INVALID_CRC
- 0x11 (-17) - INVALID_EXCEPTION_CODE

6.8.4. Example config file

Click here to download.

```
#This .INI file is also the HELP, MANUAL and HOW-TO file for mb2hal.

#Load the Modbus HAL userspace module as the examples below,
#change to match your own HAL_MODULE_NAME and INI file name
#Using HAL_MODULE_NAME=mb2hal or nothing (default): loadusr -W mb2hal config=config_file. ↩
    ini
#Using HAL_MODULE_NAME=mymodule: loadusr -Wn mymodule mb2hal config=config_file.ini

# ++++++++++++++++++++++++
# Common section
# ++++++++++++++++++++++++
[MB2HAL_INIT]

#OPTIONAL: Debug level of init and INI file parsing.
# 0 = silent.
# 1 = error messages (default).
# 2 = OK confirmation messages.
# 3 = debugging messages.
# 4 = maximum debugging messages (only in transactions).
INIT_DEBUG=3

#OPTIONAL: Set to 1.1 to enable the new functions:
# - fnct_01_read_coils
# - fnct_05_write_single_coil
# - changed pin names (see https://linuxcnc.org/docs/2.9/html/drivers/mb2hal.html#_pins).
VERSION=1.1

#OPTIONAL: HAL module (component) name. Defaults to ”mb2hal”.
HAL_MODULE_NAME=mb2hal

#OPTIONAL: Insert a delay of ”FLOAT seconds” between transactions in order
#to not to have a lot of logging and facilitate the debugging.
#Useful when using DEBUG=3 (NOT INIT_DEBUG=3)
#It affects ALL transactions.
#Use ”0.0” for normal activity.
SLOWDOWN=0.0
```

```
#REQUIRED: The number of total Modbus transactions. There is no maximum.
TOTAL_TRANSACTIONS=9

# ++++++++++++++++++++++++
# Transactions
# ++++++++++++++++++++++++
#One transaction section is required per transaction, starting at 00 and counting up  ↩
    sequentially.
#If there is a new link (not transaction), you must provide the REQUIRED parameters 1st  ↩
    time.
#Warning: Any OPTIONAL parameter not specified are copied from the previous transaction.
[TRANSACTION_00]

#REQUIRED: You must specify either a ”serial” or ”tcp” link for the first transaction.
#Later transaction will use the previous transaction link if not specified.
LINK_TYPE=tcp

#if LINK_TYPE=tcp then REQUIRED (only 1st time): The Modbus slave device ip address.
#if LINK_TYPE=serial then IGNORED
TCP_IP=192.168.2.10

#if LINK_TYPE=tcp then OPTIONAL.
#if LINK_TYPE=serial then IGNORED
#The Modbus slave device tcp port. Defaults to 502.
TCP_PORT=502

#if LINK_TYPE=serial then REQUIRED (only 1st time).
#if LINK_TYPE=tcp then IGNORED
#The serial port.
SERIAL_PORT=/dev/ttyS0

#if LINK_TYPE=serial then REQUIRED (only 1st time).
#if LINK_TYPE=tcp then IGNORED
#The baud rate.
SERIAL_BAUD=115200

#if LINK_TYPE=serial then REQUIRED (only 1st time).
#if LINK_TYPE=tcp then IGNORED
#Data bits. One of 5,6,7,8.
SERIAL_BITS=8

#if LINK_TYPE=serial then REQUIRED (only 1st time).
#if LINK_TYPE=tcp then IGNORED
#Data parity. One of: even, odd, none.
SERIAL_PARITY=none

#if LINK_TYPE=serial then REQUIRED (only 1st time).
#if LINK_TYPE=tcp then IGNORED
#Stop bits. One of 1, 2.
SERIAL_STOP=2

#if LINK_TYPE=serial then OPTIONAL:
#if LINK_TYPE=tcp then IGNORED
#Serial port delay between for this transaction only.
#In ms. Defaults to 0.
SERIAL_DELAY_MS=10

#REQUIRED (only 1st time).
#Modbus slave number.
MB_SLAVE_ID=1
```

```
#REQUIRED: The first element address (decimal integer).
FIRST_ELEMENT=0

#REQUIRED unless PIN_NAMES is specified: The number of elements.
#It is an error to specify both NELEMENTS and PIN_NAMES
#The pin names will be sequential numbers e.g mb2hal.plcin.01
#NELEMENTS=4

#REQUIRED unless NELEMENTS is specified: A list of element names.
#these names will be used for the pin names, e.g mb2hal.plcin.cycle_start
#NOTE: there must be no white space characters in the list
PIN_NAMES=cycle_start,stop,feed_hold

#REQUIRED: Modbus transaction function code (see www.modbus.org specifications).
#    fnct_01_read_coils               (01 = 0x01) (new in 1.1)
#    fnct_02_read_discrete_inputs     (02 = 0x02)
#    fnct_03_read_holding_registers   (03 = 0x03)
#    fnct_04_read_input_registers     (04 = 0x04)
#    fnct_05_write_single_coil        (05 = 0x05) (new in 1.1)
#    fnct_06_write_single_register    (06 = 0x06)
#    fnct_15_write_multiple_coils     (15 = 0x0F)
#    fnct_16_write_multiple_registers (16 = 0x10)
#
# Created pins:
# fnct_01_read_coils:
# fnct_02_read_discrete_inputs:
#     mb2hal.m.n.bit     (output)
#     mb2hal.m.n.bit-inv (output)
# fnct_03_read_holding_registers:
# fnct_04_read_input_registers:
#     mb2hal.m.n.float   (output)
#     mb2hal.m.n.int     (output)
# fnct_05_write_single_coil:
#     mb2hal.m.n.bit     (input)
#         NELEMENTS needs to be 1 or PIN_NAMES must contain just one name.
# fnct_06_write_single_register:
#     mb2hal.m.n.float   (input)
#     mb2hal.m.n.int     (input)
#     NELEMENTS needs to be 1 or PIN_NAMES must contain just one name.
#     Both pin values are added and limited to 65535 (UINT16_MAX). Normally use one and let ↩
    the other open (read as 0).
# fnct_15_write_multiple_coils:
#     mb2hal.m.n.bit     (input)
# fnct_16_write_multiple_registers:
#     mb2hal.m.n.float   (input)
#     mb2hal.m.n.int     (input)
#     Both pin values are added and limited to 65535 (UINT16_MAX). Normally use one and let ↩
    the other open (read as 0).
#
# m = HAL_TX_NAME or transaction number if not set, n = element number (NELEMENTS) or name  ↩
   from PIN_NAMES
# Example: mb2hal.00.01.<type> (transaction=00, second register=01 (00 is the first one))
#          mb2hal.TxName.01.<type> (HAL_TX_NAME=TxName, second register=01 (00 is the first ↩
    one))
MB_TX_CODE=fnct_03_read_holding_registers

#OPTIONAL: Response timeout for this transaction. In INTEGER ms. Defaults to 500 ms.
#This is how much to wait for 1st byte before raise an error.
MB_RESPONSE_TIMEOUT_MS=500

#OPTIONAL: Byte timeout for this transaction. In INTEGER ms. Defaults to 500 ms.
#This is how much to wait from byte to byte before raise an error.
```

```
MB_BYTE_TIMEOUT_MS=500

#OPTIONAL: Instead of giving the transaction number, use a name.
#Example: mb2hal.00.01 could become mb2hal.plcin.01
#The name must not exceed 28 characters.
#NOTE: when using names be careful that you dont end up with two transactions
#using the same name.
HAL_TX_NAME=remoteIOcfg

#OPTIONAL: Maximum update rate in HZ. Defaults to 0.0 (0.0 = as soon as available =  ↩
    infinite).
#NOTE: This is a maximum rate and the actual rate may be lower.
#If you want to calculate it in ms use (1000 / required_ms).
#Example: 100 ms = MAX_UPDATE_RATE=10.0, because 1000.0 ms / 100.0 ms = 10.0 Hz
MAX_UPDATE_RATE=0.0

#OPTIONAL: Debug level for this transaction only.
#See INIT_DEBUG parameter above.
DEBUG=2

#While DEBUGGING transactions note the returned ”ret[]” value correspond to:
#/* Modbus protocol exceptions */
#ILLEGAL_FUNCTION        -0x01 the FUNCTION code received in the query is not allowed or  ↩
    invalid.
#ILLEGAL_DATA_ADDRESS    -0x02 the DATA ADDRESS received in the query is not an allowable  ↩
    address for the slave or is invalid.
#ILLEGAL_DATA_VALUE      -0x03 a VALUE contained in the data query field is not an  ↩
    allowable value or is invalid.
#SLAVE_DEVICE_FAILURE    -0x04 SLAVE (or MASTER) device unrecoverable FAILURE while  ↩
    attempting to perform the requested action.
#SERVER_FAILURE          -0x04 (see above).
#ACKNOWLEDGE             -0x05 This response is returned to PREVENT A TIMEOUT in the master ↩
    .
#                             A long duration of time is required to process the request  ↩
    in the slave.
#SLAVE_DEVICE_BUSY       -0x06 The slave (or server) is BUSY. Retrasmit the request later.
#SERVER_BUSY             -0x06 (see above).
#NEGATIVE_ACKNOWLEDGE    -0x07 Unsuccessful programming request using function code 13 or  ↩
    14.
#MEMORY_PARITY_ERROR     -0x08 SLAVE parity error in MEMORY.
#GATEWAY_PROBLEM_PATH    -0x0A (-10) Gateway path(s) not available.
#GATEWAY_PROBLEM_TARGET  -0x0B (-11) The target device failed to respond (generated by  ↩
    master, not slave).
#/* Program or connection */
#COMM_TIME_OUT           -0x0C (-12)
#PORT_SOCKET_FAILURE     -0x0D (-13)
#SELECT_FAILURE          -0x0E (-14)
#TOO_MANY_DATAS          -0x0F (-15)
#INVALID_CRC             -0x10 (-16)
#INVALID_EXCEPTION_CODE  -0x11 (-17)

[TRANSACTION_01]
MB_TX_CODE=fnct_01_read_coils
FIRST_ELEMENT=1024
NELEMENTS=24
HAL_TX_NAME=remoteIOin
MAX_UPDATE_RATE=0.0
DEBUG=1

[TRANSACTION_02]
MB_TX_CODE=fnct_02_read_discrete_inputs
FIRST_ELEMENT=1280
```

```
NELEMENTS=8
HAL_TX_NAME=readStatus
MAX_UPDATE_RATE=0.0

[TRANSACTION_03]
MB_TX_CODE=fnct_05_write_single_coil
FIRST_ELEMENT=100
NELEMENTS=1
HAL_TX_NAME=setEnableout
MAX_UPDATE_RATE=0.0

[TRANSACTION_04]
MB_TX_CODE=fnct_15_write_multiple_coils
FIRST_ELEMENT=150
NELEMENTS=10
HAL_TX_NAME=remoteIOout
MAX_UPDATE_RATE=0.0

[TRANSACTION_05]
LINK_TYPE=serial
SERIAL_PORT=/dev/ttyS0
SERIAL_BAUD=115200
SERIAL_BITS=8
SERIAL_PARITY=none
SERIAL_STOP=2
SERIAL_DELAY_MS=50
MB_SLAVE_ID=1
MB_TX_CODE=fnct_03_read_holding_registers
FIRST_ELEMENT=1
NELEMENTS=2
HAL_TX_NAME=XDrive01
MAX_UPDATE_RATE=0.0
DEBUG=1

[TRANSACTION_06]
MB_TX_CODE=fnct_04_read_input_registers
FIRST_ELEMENT=12
NELEMENTS=3
HAL_TX_NAME=XDrive02
MAX_UPDATE_RATE=10.0
DEBUG=1

[TRANSACTION_07]
MB_TX_CODE=fnct_06_write_single_register
FIRST_ELEMENT=20
NELEMENTS=1
HAL_TX_NAME=XDrive03
MAX_UPDATE_RATE=0.0
DEBUG=1

[TRANSACTION_08]
MB_TX_CODE=fnct_16_write_multiple_registers
FIRST_ELEMENT=55
NELEMENTS=8
HAL_TX_NAME=XDrive04
MAX_UPDATE_RATE=10.0
DEBUG=1
```

6.8.5. Pines

nota
Yellow = Nuevo en MB2HAL 1.1 (LinuxCNC 2.9) Para usar estas nuevas características debes asignar `VERSION = 1.1`.
m = Value of `HAL_TX_NAME` if set or transaction number
n = Element number (`NELEMENTS`) or name from `PIN_NAMES`
Ejemplo:

- `mb2hal.00.01.int` (TRANSACTION_00, second register)
- `mb2hal.readStatus.01.bit` (HAL_TX_NAME=readStatus, first bit)

6.8.5.1. fnct_01_read_coils

- mb2hal.*m*.*n*.bit *bit out*
- mb2hal.*m*.*n*.bit-inv *bit out*

6.8.5.2. fnct_02_read_discrete_inputs

- mb2hal.*m*.*n*.bit *bit out*
- mb2hal.*m*.*n*.bit-inv *bit out*

6.8.5.3. fnct_03_read_holding_registers

- mb2hal.*m*.*n*.float *float out*
- mb2hal.*m*.*n*.int *s32 out*

6.8.5.4. fnct_04_read_input_registers

- mb2hal.*m*.*n*.float *float out*
- mb2hal.*m*.*n*.int *s32 out*

6.8.5.5. fnct_05_write_single_coil

- mb2hal.*m*.*n*.bit *bit in*

`NELEMENTS` needs to be 1 or `PIN_NAMES` must contain just one name.

6.8.5.6. fnct_06_write_single_register

- mb2hal.*m*.*n*.float *float in*
- mb2hal.*m*.*n*.int *s32 in*

`NELEMENTS` needs to be 1 or `PIN_NAMES` must contain just one name. Both pin values are added and limited to 65535 (UINT16_MAX). Use one and let the other open (read as 0).

6.8.5.7. fnct_15_write_multiple_coils

- mb2hal.*m*.*n*.bit *bit in*

6.8.5.8. fnct_16_write_multiple_registers

- mb2hal.*m*.*n*.float *float in*
- mb2hal.*m*.*n*.int *s32 in*

Both pin values are added and limited to 65535 (UINT16_MAX). Use one and let the other open (read as 0).

6.9. Mitsub VFD Driver

This is a non-realtime HAL program, written in Python, to control VFDs from Mitsubishi.
Specifically the A500 F500 E500 A500 D700 E700 F700 series - others may work.
mitsub_vfd supports serial control using the RS485 protocol.
Conversion from USB or serial port to RS485 requires special hardware.

---

nota
Since this is a non-realtime program it can be affected by computer loading and latency. It is possible to lose control of the VFDs. It is optional to set the VFD to stop if it loses communication if that is desirable. One should always have an Estop circuit that kills the power to the unit in case of emergency.

This component is loaded using the halcmd ”loadusr” command:

```
loadusr -Wn coolant mitsub_vfd spindle=02 coolant=01
```

El comando anterior dice:
loadusr, espera a que los pines de refrigerante estén listos, componente mitsub_vfd, con 2 esclavos nombrados husillo (esclavo #2) y refrigerante (esclavo #1)

6.9.1. Command Line Options

The command line options are:

- *-b or --baud <rate>* : set the baud rate - all networked VFDs must be the same
- *-p or --port <device path>* : sets the port to use such as /dev/ttyUSB0
- *<name>=<slave#>* : sets the HAL component/pin name and slave number.

Debugging can be toggled by setting the debug pin true.

---

nota
Turning on debugging will result in a flood of text in the terminal.

6.9.2. Pines

Where <n> is `mitsub_vfd` or the name given during loading.

- *<n>.fwd* (bit, in) True sets motion forward, False sets reverse.
- *<n>.run* (bit, in) True sets the VFD in motion based on the .fwd pin.
- *<n>.debug* (bit, in) Prints debug info to the terminal.
- *<n>.alarm* (bit, out) signals an alarm state of VFD.
- *<n>.up-to-speed* (bit, out) when drive is at commanded speed (speed-tolerance is set on vfd)
- *<n>.monitor* (bit, in) some models (eg E500) cannot monitor status - set the monitor pin to false in this case pins such as up-to-speed, amps, alarm and status bits are not updated.
- *<n>.motor-cmd* (float, in) commanded speed to the VFD (scaled to hertz by default).
- *<n>.motor-fb* (float, out) feedback speed from the VFD (scaled to hertz by default).
- *<n>.motor-amps* (float, out) Current amperage output of motor.
- *<n>.motor-power* (float, out) Current power output of motor.
- *<n>.scale-cmd* (float, in) Scales the motor-cmd pin to arbitrary units. default 1 = Hertz.
- *<n>.scale-fb* (float, in) Scales the motor-fb pin to arbitrary units. default 1 = Hertz.
- *<n>.scale-amps* (float, in) Scales the motor-amps pin to arbitrary units. default 1 = amps.
- *<n>.scale-power* (float, in) Scales the motor-power pin to arbitrary units. default 1 = .
- *<n>.estop* (bit, in) puts the VFD into emergency-stopped status.
- *<n>.status-bit-N* (bit, out) N = 0 to 7, status bits are user configurable on the VFD. Bit 3 should be set to at speed and bit 7 should be set to alarm. Others are free to be set as required.

6.9.3. Ejemplo HAL

```
#
# example usage of the Mitsubishi VFD driver
#
loadusr -Wn coolant mitsub_vfd spindle=02 coolant=01

# *************** Spindle VFD setup slave 2 *********************
net spindle-vel-cmd               spindle.motor-cmd
net spindle-cw                    spindle.fwd
net spindle-on                    spindle.run
net spindle-at-speed              spindle.up-to-speed
net estop-out                     spindle.estop
#       cmd scaled to RPM
setp spindle.scale-cmd .135
#       feedback is in rpm
setp spindle.scale-fb 7.411
#       allows us to see status
setp spindle.monitor 1

net spindle-speed-indicator spindle.motor-fb        gladevcp.spindle-speed

# *************** Coolant vfd setup slave 3 ***********************
net coolant-flood               coolant.run
net coolant-is-on               coolant.up-to-speed   gladevcp.coolant-on-led
```

```
net estop-out                   coolant.estop
#       cmd and feedback scaled to hertz
setp coolant.scale-cmd 1
setp coolant.scale-fb 1
#       command full speed
setp coolant.motor-cmd 60
#       allows us to see status
setp coolant.monitor 1
```

6.9.4. Configuring the Mitsubishi VFD for serial usage

6.9.4.1. Connecting the Serial Port

The Mitsubishi VFDs have an RJ-45 jack for serial communication.
Since they use RS485 protocol, they can be networked together point to point.
This driver was tested using the Opto22 AC7A to convert from RS232 to RS485.

6.9.4.2. Modbus setup

Manuales referenciados:

communication option reference manual and *A500 technical manual* for 500 series.
Fr-A700 F700 E700 D700 technical manual for the 700 series

The VFD must have PR settings adjusted manually for serial communication.
One must power cycle the VFD for some of these to register eg PR 79

- *PR 77* set to 1 *-to unlock other PR modification.*
- *PR 79* set to 1 or 0 *-for communication thru serial.*
- *PR 117* set to 0-31 *-slave number, driver must reference same number.*
- *PR 118* tested with 96 *-baud rate (can be set to 48,96,192) if driver is also set.*
- *PR 119* set to 0 *-stop bit/data length (8 bits, two stop)*
- *PR 120* set to 0 *-no parity*
- *PR 121* set to 1-10 *-if 10 (maximum) COM errors then VFD faults.*
- *PR 122* tested with 9999 *-if communication is lost VFD will not error.*
- *PR 123* set to 9999 *-no wait time is added to the serial data frame.*
- *PR 124* set to 0 *-no carriage return at end of line.*

6.10. Motenc Driver

Vital Systems Motenc-100 and Motenc-LITE

The Vital Systems Motenc-100 and Motenc-LITE are 8- and 4-channel servo control boards. The Motenc-100 provides 8 quadrature encoder counters, 8 analog inputs, 8 analog outputs, 64 (68?) digital inputs, and 32 digital outputs. The Motenc-LITE has only 4 encoder counters, 32 digital inputs and 16 digital outputs, but it still has 8 analog inputs and 8 analog outputs. The driver automatically identifies the installed board and exports the appropriate HAL objects.

Instalación:

```
loadrt hal_motenc
```

Durante la carga (o intento de carga) el controlador imprime algunos mensajes de depuración útiles en la bitácora del kernel, los cuales pueden ser revisados con dmesg.

Up to 4 boards may be used in one system.

6.10.1. Pines

In the following pins, parameters, and functions, <board> is the board ID. According to the naming conventions the first board should always have an ID of zero. However this driver sets the ID based on a pair of jumpers on the board, so it may be non-zero even if there is only one board.

- *(s32) motenc.<board>.enc-<channel>-count* - Encoder position, in counts.
- *(float) motenc.<board>.enc-<channel>-position* - Encoder position, in user units.
- *(bit) motenc.<board>.enc-<channel>-index* - Current status of index pulse input.
- *(bit) motenc.<board>.enc-<channel>-idx-latch* - Driver sets this pin true when it latches an index pulse (enabled by latch-index). Cleared by clearing latch-index.
- *(bit) motenc.<board>.enc-<channel>-latch-index* - If this pin is true, the driver will reset the counter on the next index pulse.
- *(bit) motenc.<board>.enc-<channel>-reset-count* - If this pin is true, the counter will immediately be reset to zero, and the pin will be cleared.
- *(float) motenc.<board>.dac-<channel>-value* - Analog output value for DAC (in user units, see -gain and -offset)
- *(float) motenc.<board>.adc-<channel>-value* - Analog input value read by ADC (in user units, see -gain and -offset)
- *(bit) motenc.<board>.in-<channel>* - State of digital input pin, see canonical digital input.
- *(bit) motenc.<board>.in-<channel>-not* - Inverted state of digital input pin, see canonical digital input.
- *(bit) motenc.<board>.out-<channel>* - Value to be written to digital output, seen canonical digital output.
- *(bit) motenc.<board>.estop-in* - Dedicated estop input, more details needed.
- *(bit) motenc.<board>.estop-in-not* - Inverted state of dedicated estop input.
- *(bit) motenc.<board>.watchdog-reset* - Bidirectional, - Set TRUE to reset watchdog once, is automatically cleared.

6.10.2. Parámetros

- *(float) motenc.<board>.enc-<channel>-scale* - The number of counts / user unit (to convert from counts to units).
- *(float) motenc.<board>.dac-<channel>-offset* - Sets the DAC offset.
- *(float) motenc.<board>.dac-<channel>-gain* - Sets the DAC gain (scaling).
- *(float) motenc.<board>.adc-<channel>-offset* - Sets the ADC offset.

- *(float) motenc.<board>.adc-<channel>-gain* - Sets the ADC gain (scaling).
- *(bit) motenc.<board>.out-<channel>-invert* - Inverts a digital output, see canonical digital output.
- *(u32) motenc.<board>.watchdog-control* - Configures the watchdog.
  The value may be a bitwise OR of the following values:

Bit #	Valor	Meaning
0	1	Timeout is 16ms if set, 8ms if unset
1	2	
2	4	Watchdog is enabled
3	8	
4	16	Watchdog is automatically reset by DAC writes (the HAL dac-write function)

Typically, the useful values are 0 (watchdog disabled) or 20 (8ms watchdog enabled, cleared by dac-write).

- *(u32) motenc.<board>.led-view* - Maps some of the I/O to onboard LEDs.

6.10.3. Funciones

- *(funct) motenc.<board>.encoder-read* - Reads all encoder counters.
- *(funct) motenc.<board>.adc-read* - Reads the analog-to-digital converters.
- *(funct) motenc.<board>.digital-in-read* - Reads digital inputs.
- *(funct) motenc.<board>.dac-write* - Writes the voltages to the DACs.
- *(funct) motenc.<board>.digital-out-write* - Writes digital outputs.
- *(funct) motenc.<board>.misc-update* - Updates misc stuff.

6.11. Opto22 Driver

PCI AC5 ADAPTER CARD / HAL DRIVER

6.11.1. The Adapter Card

This is a card made by Opto22 for adapting the PCI port to solid state relay racks such as their standard or G4 series. It has 2 ports that can control up to 24 points each and has 4 on board LEDs. The ports use 50 pin connectors the same as Mesa boards. Any relay racks/breakout boards that work with Mesa Cards should work with this card with the understanding any encoder counters, PWM, etc., would have to be done in software. The AC5 does not have any *smart* logic on board, it is just an adapter.

Vea el sitio web del fabricante para mayor información:

https://www.opto22.com/site/pr_details.aspx?cid=4&item=PCI-AC5

I would like to thank Opto22 for releasing info in their manual, easing the writing of this driver!

6.11.2. The Driver

This driver is for the PCI AC5 card and will not work with the ISA AC5 card. The HAL driver is a realtime module. It will support 4 cards as is (more cards are possible with a change in the source code). Load the basic driver like so:

```
loadrt opto_ac5
```

This will load the driver which will search for max 4 boards. It will set I/O of each board's 2 ports to a default setting. The default configuration is for 12 inputs then 12 outputs. The pin name numbers correspond to the position on the relay rack. For example the pin names for the default I/O setting of port 0 would be:

- *opto_ac5.0.port0.in-00* - They would be numbered from 00 to 11
- *opto_ac5.0.port0.out-12* - They would be numbered 12 to 23 port 1 would be the same.

6.11.3. Pines

- *opto_ac5.[BOARDNUMBER].port[PORTNUMBER].in-[PINNUMBER] OUT bit* -
- *opto_ac5.[BOARDNUMBER].port[PORTNUMBER].in-[PINNUMBER]-not OUT bit* - Connect a HAL bit signal to this pin to read an I/O point from the card. The PINNUMBER represents the position in the relay rack. Eg. PINNUMBER 0 is position 0 in a Opto22 relay rack and would be pin 47 on the 50 pin header connector. The -not pin is inverted so that LOW gives TRUE and HIGH gives FALSE.
- *opto_ac5.[BOARDNUMBER].port[PORTNUMBER].out-[PINNUMBER] IN bit* - Connect a HAL bit signal to this pin to write to an I/O point of the card. The PINNUMBER represents the position in the relay rack.Eg. PINNUMBER 23 is position 23 in a Opto22 relay rack and would be pin 1 on the 50 pin header connector.
- *opto_ac5.[BOARDNUMBER].led[NUMBER] OUT bit* - Turns one of the 4 onboard LEDs on/off. LEDs are numbered 0 to 3.

BOARDNUMBER can be 0-3 PORTNUMBER can be 0 or 1. Port 0 is closest to the card bracket.

6.11.4. Parámetros

- *opto_ac5.[BOARDNUMBER].port[PORTNUMBER].out-[PINNUMBER]-invert W bit* - When TRUE, invert the meaning of the corresponding -out pin so that TRUE gives LOW and FALSE gives HIGH.

6.11.5. FUNCTIONS

- *opto_ac5.0.digital-read* - Add this to a thread to read all the input points.
- *opto_ac5.0.digital-write* - Add this to a thread to write all the output points and LEDs.

For example the pin names for the default I/O setting of port 0 would be:

```
opto_ac5.0.port0.in-00
```

They would be numbered from 00 to 11

```
opto_ac5.0.port0.out-12
```

They would be numbered 12 to 23 port 1 would be the same.

6.11.6. Configuring I/O Ports

To change the default setting load the driver something like so:

```
loadrt opto_ac5 portconfig0=0xffff portconfig1=0xff0000
```

Of course changing the numbers to match the I/O you would like. Each port can be set up different.

Here's how to figure out the number: The configuration number represents a 32 bit long code to tell the card which I/O points are output vrs input. The lower 24 bits are the I/O points of one port. The 2 highest bits are for 2 of the on board LEDs. A one in any bit position makes the I/O point an output. The two highest bits must be output for the LEDs to work. The driver will automatically set the two highest bits for you, we won't talk about them.

The easiest way to do this is to fire up the calculator under APPLICATIONS/ACCESSORIES. Set it to scientific (click view). Set it BINARY (radio button Bin). Press 1 for every output you want and/or zero for every input. Remember that HAL pin 00 corresponds to the rightmost bit. 24 numbers represent the 24 I/O points of one port. So for the default setting (12 inputs then 12 outputs) you would push 1 twelve times (that's the outputs) then 0 twelve times (that's the inputs). Notice the first I/O point is the lowest (rightmost) bit. (that bit corresponds to HAL pin 00 .looks backwards) You should have 24 digits on the screen. Now push the Hex radio button. The displayed number (fff000) is the configport number (put a *0x* in front of it designating it as a HEX number).

Another example: To set the port for 8 outputs and 16 inputs (the same as a Mesa card). Here is the 24 bits represented in a BINARY number. Bit 1 is the rightmost number:

16 zeros for the 16 inputs and 8 ones for the 8 outputs

```
000000000000000011111111
```

This converts to FF on the calculator, so 0xff is the number to use for portconfig0 and/or portconfig1 when loading the driver.

6.11.7. Pin Numbering

HAL pin 00 corresponds to bit 1 (the rightmost) which represents position 0 on an Opto22 relay rack. HAL pin 01 corresponds to bit 2 (one spot to the left of the rightmost) which represents position 1 on an Opto22 relay rack. HAL pin 23 corresponds to bit 24 (the leftmost) which represents position 23 on an Opto22 relay rack.

HAL pin 00 connects to pin 47 on the 50 pin connector of each port. HAL pin 01 connects to pin 45 on the 50 pin connector of each port. HAL pin 23 connects to pin 1 on the 50 pin connector of each port.

Note that Opto22 and Mesa use opposite numbering systems: Opto22 position 23 = connector pin 1, and the position goes down as the connector pin number goes up. Mesa Hostmot2 position 1 = connector pin 1, and the position number goes up as the connector pin number goes up.

6.12. Pico Drivers

Pico Systems has a family of boards for doing analog servo, stepper, and PWM (digital) servo control. The boards connect to the PC through a parallel port working in EPP mode. Although most users connect one board to a parallel port, in theory any mix of up to 8 or 16 boards can be used on a single parport. One driver serves all types of boards. The final mix of I/O depends on the connected board(s). The driver doesn't distinguish between boards, it simply numbers I/O channels (encoders, etc) starting from 0 on the first board. The driver is named hal_ppmc.ko The analog servo interface is also called the PPMC for Parallel Port Motion Control. There is also the Universal Stepper Controller, abbreviated the USC. And the Universal PWM Controller, or UPC.

Instalación:

```
loadrt hal_ppmc port_addr=<addr1>[,<addr2>[,<addr3>...]]
```

The *port_addr* parameter tells the driver what parallel port(s) to check. By default, <*addr1*> is 0x0378, and <*addr2*> and following are not used. The driver searches the entire address space of the enhanced parallel port(s) at *port_addr*, looking for any board(s) in the PPMC family. It then exports HAL pins for whatever it finds. During loading (or attempted loading) the driver prints some useful debugging messages to the kernel log, which can be viewed with *dmesg*.

Up to 3 parport buses may be used, and each bus may have up to 8 (or possibly 16 PPMC) devices on it.

6.12.1. Command Line Options

There are several options that can be specified on the loadrt command line. First, the USC and UPC can have an 8-bit DAC added for spindle speed control and similar functions. This can be specified with the extradac=*0xnn[,0xmm]* parameter. The part enclosed in [ ] allows you to specify this option on more than one board of the system. The first hex digit tells which EPP bus is being referred to, it corresponds to the order of the port addresses in the port_addr parameter, where <addr1> would be zero here. So, for the first EPP bus, the first USC or UPC board would be described as *0x00*, the second USC or UPC on the same bus would be *0x02*. (Note that each USC or UPC takes up two addresses, so if one is at 00, the next would have to be 02.)

Alternatively, the 8 digital output pins can be used as additional digital outputs, it works the same way as above with the syntax : extradout=0xnn'. The extradac and extradout options are mutually exclusive on each board, you can only specify one.

The UPC and PPMC encoder boards can timestamp the arrival of encoder counts to refine the derivation of axis velocity. This derived velocity can be fed to the PID hal component to produce smoother D term response. The syntax is : timestamp=*0xnn[,0xmm]*, this works the same way as above to select which board is being configured. Default is to not enable the timestamp option. If you put this option on the command line, it enables the option. The first *n* selects the EPP bus, the second one matches the address of the board having the option enabled. The driver checks the revision level of the board to make sure it has firmware supporting the feature, and produces an error message if the board does not support it.

The PPMC encoder board has an option to select the encoder digital filter frequeency. (The UPC has the same ability via DIP switches on the board.) Since the PPMC encoder board doesn't have these extra DIP switches, it needs to be selected via a command-line option. By default, the filter runs at 1 MHz, allowing encoders to be counted up to about 900 kHz (depending on noise and quadrature accuracy of the encoder.) The options are 1, 2.5, 5 and 10 MHz. These are set with a parameter of 1,2,5 and 10 (decimal) which is specified as the hex digit "A". These are specified in a manner similar to the above options, but with the frequency setting to the left of the bus/address digits. So, to set 5 MHz on the encoder board at address 3 on the first EPP bus, you would write: `enc_clock='0x503'`.

It was recently discovered that some parallel port chips would not work with the ppmc driver. Especially, the Oxford OXPCIe952 chip on the SIIG PCIe parallel port cards had this trouble. The ppmc driver in all LinuxCNC versions starting from 2.7.8 have been corrected for this problem by default. However, this possibly could cause problems with really old EPP parallel port hardware, so there is a command line option to go back to the previous behavior. The new behavior is set by default, or by adding the parameter epp_dir=0 on the command line. To get the old behavior, add epp_dir=1 to the command line. All parallel ports I have here work with the new default behavior. As on the other parameters, it is possible to give a list, like epp_dir=1,0,1 to set different settings for each of up to 3 parallel ports.

6.12.2. Pines

In the following pins, parameters, and functions, <port> is the parallel port ID. According to the naming conventions the first port should always have an ID of zero. All the boards have some method of

setting the address on the EPP bus. USC and UPC have simple provisions for only two addresses, but jumper foil cuts allow up to 4 boards to be addressed. The PPMC boards have 16 possible addresses. In all cases, the driver enumerates the boards by type and exports the appropriate HAL pins. For instance, the encoders will be enumerated from zero up, in the same order as the address switches on the board specify. So, the first board will have encoders 0 — 3, the second board would have encoders 4 — 7. The first column after the bullet tells which boards will have this HAL pin or parameter associated with it. All means that this pin is available on all three board types. Option means that this pin will only be exported when that option is enabled by an optional parameter in the loadrt HAL command. These options require the board to have a sufficient revision level to support the feature.

- *(All s32 output) ppmc.<port>.encoder.<channel>.count* - Encoder position, in counts.
- *(All s32 output) ppmc.<port>.encoder.<channel>.delta* - Change in counts since last read, in raw encoder count units.
- *(All float output) 'ppmc.<port>.encoder.<channel>.velocity* - Velocity scaled in user units per second. On PPMC and USC this is derived from raw encoder counts per servo period, and hence is affected by encoder granularity. On UPC boards with the 8/21/09 and later firmware, velocity estimation by timestamping encoder counts can be used to improve the smoothness of this velocity output. This can be fed to the PID HAL component to produce a more stable servo response. This function has to be enabled in the HAL command line that starts the PPMC driver, with the timestamp=0x00 option.
- *(All float output) ppmc.<port>.encoder.<channel>.position* - Encoder position, in user units.
- *(All bit bidir) ppmc.<port>.encoder.<channel>.index-enable* - Connect to joint.#.index-enable for home-to-index. This is a bidirectional HAL signal. Setting it to true causes the encoder hardware to reset the count to zero on the next encoder index pulse. The driver will detect this and set the signal back to false.
- *(PPMC float output) ppmc.<port>.DAC.<channel>.value* - sends a signed value to the 16-bit Digital to Analog Converter on the PPMC DAC16 board commanding the analog output voltage of that DAC channel.
- *(UPC bit input) ppmc.<port>.pwm.<channel>.enable* - Enables a PWM generator.
- *(UPC float input) ppmc.<port>.pwm.<channel>.value* - Value which determines the duty cycle of the PWM waveforms. The value is divided by *pwm.<channel>.scale*, and if the result is 0.6 the duty cycle will be 60 %, and so on. Negative values result in the duty cycle being based on the absolute value, and the direction pin is set to indicate negative.
- *(USC bit input) ppmc.<port>.stepgen.<channel>.enable* - Enables a step pulse generator.
- *(USC float input) ppmc.<port>.stepgen.<channel>.velocity* - Value which determines the step frequency. The value is multiplied by *stepgen.<channel>.scale* , and the result is the frequency in steps per second. Negative values result in the frequency being based on the absolute value, and the direction pin is set to indicate negative.
- *(All bit output) ppmc.<port>.din.<channel>.in* - State of digital input pin, see canonical digital input.
- *(All bit output) ppmc.<port>.din.<channel>.in-not* - Inverted state of digital input pin, see canonical digital input.
- *(All bit input) ppmc.<port>.dout.<channel>.out* - Value to be written to digital output, see canonical digital output.
- *(Option float input) ppmc.<port>.DAC8-<channel>.value* - Value to be written to analog output, range from 0 to 255. This sends 8 output bits to J8, which should have a Spindle DAC board connected to it. 0 corresponds to zero Volts, 255 corresponds to 10 Volts. The polarity of the output can be set for always minus, always plus, or can be controlled by the state of SSR1 (plus when on) and SSR2 (minus when on). You must specify extradac = 0x00 on the HAL command line that loads the PPMC driver to enable this function on the first USC ur UPC board.

- *(Option bit input) ppmc.<port>.dout.<channel>.out* - Value to be written to one of the 8 extra digital output pins on J8. You must specify extradout = 0x00 on the HAL command line that loads the ppmc driver to enable this function on the first USC or UPC board. extradac and extradout are mutually exclusive features as they use the same signal lines for different purposes. These output pins will be enumerated after the standard digital outputs of the board.

6.12.3. Parámetros

- *(All float) ppmc.<port>.encoder.<channel>.scale* - The number of counts / user unit (to convert from counts to units).
- *(UPC float) ppmc.<port>.pwm.<channel-range>.freq* - The PWM carrier frequency, in Hz. Applies to a group of four consecutive PWM generators, as indicated by <*channel-range*>. Minimum is 610 Hz, maximum is 500 kHz.
- *(PPMC float) ppmc.<port>.DAC.<channel>.scale* - Sets scale of DAC16 output channel such that an output value equal to the 1/scale value will produce an output of + or - value Volts. So, if the scale parameter is 0.1 and you send a value of 0.5, the output will be 5.0 Volts.
- *(UPC float) ppmc.<port>.pwm.<channel>.scale* - Scaling for PWM generator. If *scale* is X, then the duty cycle will be 100 % when the *value* pin is X (or -X).
- *(UPC float) ppmc.<port>.pwm.<channel>.max-dc* - Maximum duty cycle, from 0.0 to 1.0.
- *(UPC float) ppmc.<port>.pwm.<channel>.min-dc* - Minimum duty cycle, from 0.0 to 1.0.
- *(UPC float) ppmc.<port>.pwm.<channel>.duty-cycle* - Actual duty cycle (used mostly for troubleshooting.)
- *(UPC bit) ppmc.<port>.pwm.<channel>.bootstrap* - If true, the PWM generator will generate a short sequence of pulses of both polarities when E-stop goes false, to reset the shutdown latches on some PWM servo drives.
- *(USC u32) ppmc.<port>.stepgen.<channel-range>.setup-time* - Sets minimum time between direction change and step pulse, in units of 100 ns. Applies to a group of four consecutive step generators, as indicated by <*channel-range*>. Values between 200 ns and 25.5 µs can be specified.
- *(USC u32) ppmc.<port>.stepgen.<channel-range>.pulse-width* - Sets width of step pulses, in units of 100 ns. Applies to a group of four consecutive step generators, as indicated by <*channel-range*>. Values between 200 ns and 25.5 µs may be specified.
- *(USC u32) ppmc.<port>.stepgen.<channel-range>.pulse-space-min* - Sets minimum time between pulses, in units of 100 ns. Applies to a group of four consecutive step generators, as indicated by <*channel-range*>. Values between 200 ns and 25.5 µs can be specified. The maximum step rate is:

$$\frac{1}{100\text{ns} * (pulsewidth + pulsespacemin)}$$

- *(USC float) ppmc.<port>.stepgen.<channel>.scale* - Scaling for step pulse generator. The step frequency in Hz is the absolute value of *velocity* * *scale*.
- *(USC float) ppmc.<port>.stepgen.<channel>.max-vel* - The maximum value for *velocity*. Commands greater than *max-vel* will be clamped. Also applies to negative values. (The absolute value is clamped.)
- *(USC float) ppmc.<port>.stepgen.<channel>.frequency* - Actual step pulse frequency in Hz (used mostly for troubleshooting.)
- *(Option float) ppmc.<port>.DAC8.<channel>.scale* - Sets scale of extra DAC output such that an output value equal to scale gives a magnitude of 10.0 V output. (The sign of the output is set by jumpers and/or other digital outputs.)

- *(Option bit) ppmc.<port>.dout.<channel>.invert* - Inverts a digital output, see canonical digital output.
- *(Option bit) ppmc.<port>.dout.<channel>.invert* - Inverts a digital output pin of J8, see canonical digital output.

6.12.4. Funciones

- *(All funct) ppmc.<port>.read* - Reads all inputs (digital inputs and encoder counters) on one port. These reads are organized into blocks of contiguous registers to be read in a block to minimize CPU overhead.
- *(All funct) ppmc.<port>.write* - Writes all outputs (digital outputs, stepgens, PWMs) on one port. These writes are organized into blocks of contiguous registers to be written in a block to minimize CPU overhead.

6.13. Pluto P Driver

6.13.1. General Info

The Pluto-P is a FPGA board featuring the ACEX1K chip from Altera.

6.13.1.1. Requisitos

1. A Pluto-P board
2. An EPP-compatible parallel port, configured for EPP mode in the system BIOS or a PCI EPP compatible parallel port card.

---

nota
The Pluto P board requires EPP mode. Netmos98xx chips do not work in EPP mode. The Pluto P board will work on some computers and not on others. There is no known pattern to which computers work and which don't work.

For more information on PCI EPP compatible parallel port cards see the LinuxCNC Supported Hardware page on the wiki.

6.13.1.2. Connectors

- The Pluto-P board is shipped with the left connector presoldered, with the key in the indicated position. The other connectors are unpopulated. There does not seem to be a standard 12-pin IDC connector, but some of the pins of a 16P connector can hang off the board next to QA3/QZ3.
- The bottom and right connectors are on the same .1" grid, but the left connector is not. If OUT2…OUT9 are not required, a single IDC connector can span the bottom connector and the bottom two rows of the right connector.

6.13.1.3. Physical Pins

- Read the ACEX1K datasheet for information about input and output voltage thresholds. The pins are all configured in *LVTTL/LVCMOS* mode and are generally compatible with 5V TTL logic.
- Before configuration and after properly exiting LinuxCNC, all Pluto-P pins are tristated with weak pull-ups (20 kΩ min, 50 kΩ max). If the watchdog timer is enabled (the default), these pins are also tristated after an interruption of communication between LinuxCNC and the board. The watchdog timer takes approximately 6.5 ms to activate. However, software bugs in the pluto_servo firmware or LinuxCNC can leave the Pluto-P pins in an undefined state.
- In pwm+dir mode, by default dir is HIGH for negative values and LOW for positive values. To select HIGH for positive values and LOW for negative values, set the corresponding dout-NN-invert parameter TRUE to invert the signal.
- The index input is triggered on the rising edge. Initial testing has shown that the QZx inputs are particularly noise sensitive, due to being polled every 25 ns. Digital filtering has been added to filter pulses shorter than 175 ns (seven polling times). Additional external filtering on all input pins, such as a Schmitt buffer or inverter, RC filter, or differential receiver (if applicable) is recommended.
- The IN1…IN7 pins have 22 Ω series resistors to their associated FPGA pins. No other pins have any sort of protection for out-of-spec voltages or currents. It is up to the integrator to add appropriate isolation and protection. Traditional parallel port optoisolator boards do not work with pluto_servo due to the bidirectional nature of the EPP protocol.

6.13.1.4. LED

- When the device is unprogrammed, the LED glows faintly. When the device is programmed, the LED glows according to the duty cycle of PWM0 (*LED = UP0 xor DOWN0*) or STEPGEN0 (*LED = STEP0 xor DIR0*).

6.13.1.5. Power

- A small amount of current may be drawn from VCC. The available current depends on the unregulated DC input to the board. Alternately, regulated +3.3VDC may be supplied to the FPGA through these VCC pins. The required current is not yet known, but is probably around 50mA plus I/O current.
- The regulator on the Pluto-P board is a low-dropout type. Supplying 5V at the power jack will allow the regulator to work properly.

6.13.1.6. PC interface

- Only a single pluto_servo or pluto_step board is supported.

6.13.1.7. Rebuilding the FPGA firmware

The *src/hal/drivers/pluto_servo_firmware/* and *src/hal/drivers/pluto_step_firmware/* subdirectories contain the Verilog source code plus additional files used by Quartus for the FPGA firmwares. Altera's Quartus II software is required to rebuild the FPGA firmware. To rebuild the firmware from the .hdl and other source files, open the *.qpf* file and press CTRL-L. Then, recompile LinuxCNC.

Like the HAL hardware driver, the FPGA firmware is licensed under the terms of the GNU General Public License.

The gratis version of Quartus II runs only on Microsoft Windows, although there is apparently a paid version that runs on Linux.

6.13.1.8. For more information

Some additional information about it is available from KNJC LLC and from the developer's blog.

6.13.2. Pluto Servo

The pluto_servo system is suitable for control of a 4-axis CNC mill with servo motors, a 3-axis mill with PWM spindle control, a lathe with spindle encoder, etc. The large number of inputs allows a full set of limit switches.

This driver features:

- 4 quadrature channels with 40 MHz sample rate. The counters operate in *4x* mode. The maximum useful quadrature rate is 8191 counts per LinuxCNC servo cycle, or about 8 MHz for LinuxCNC's default 1 ms servo rate.
- 4 PWM channels, *up/down* or *pwm+dir* style. 4095 duty cycles from -100 % to +100 %, including 0 %. The PWM period is approximately 19.5 kHz (40 MHz / 2047). A PDM-like mode is also available.
- 18 digital outputs: 10 dedicated, 8 shared with PWM functions. (Example: A lathe with unidirectional PWM spindle control may use 13 total digital outputs)
- 20 digital inputs: 8 dedicated, 12 shared with Quadrature functions. (Example: A lathe with index pulse only on the spindle may use 13 total digital inputs.)
- EPP communication with the PC. The EPP communication typically takes around 100 µs on machines tested so far, enabling servo rates above 1 kHz.

6.13.2.1. Asignación de pines

- *UPx* - The *up* (up/down mode) or *pwm* (pwm+direction mode) signal from PWM generator X. May be used as a digital output if the corresponding PWM channel is unused, or the output on the channel is always negative. The corresponding digital output invert may be set to TRUE to make UPx active low rather than active high.
- *DNx* - The *down* (up/down mode) or *direction* (pwm+direction mode) signal from PWM generator X. May be used as a digital output if the corresponding PWM channel is unused, or the output on the channel is never negative. The corresponding digital output invert may be set to TRUE to make DNx active low rather than active high.
- *QAx, QBx* - The A and B signals for Quadrature counter X. May be used as a digital input if the corresponding quadrature channel is unused.
- *QZx* - The Z (index) signal for quadrature counter X. May be used as a digital input if the index feature of the corresponding quadrature channel is unused.
- *INx* - Dedicated digital input #x
- *OUTx* - Dedicated digital output #x
- *GND* - Ground
- *VCC* - +3.3V regulated DC

.Pluto-Servo Pinout Pluto-Servo Pinout

Cuadro 6.41: Pluto-Servo Alternate Pin Functions

Primary function	**Alternate Function**	**Behavior if both functions used**
UP0	PWM0	When pwm-0-pwmdir is TRUE, this pin is the PWM output
	OUT10	XOR'd with UP0 or PWM0
UP1	PWM1	When pwm-1-pwmdir is TRUE, this pin is the PWM output
	OUT12	XOR'd with UP1 or PWM1
UP2	PWM2	When pwm-2-pwmdir is TRUE, this pin is the PWM output
	OUT14	XOR'd with UP2 or PWM2
UP3	PWM3	When pwm-3-pwmdir is TRUE, this pin is the PWM output
	OUT16	XOR'd with UP3 or PWM3
DN0	DIR0	When pwm-0-pwmdir is TRUE, this pin is the DIR output
	OUT11	XOR'd with DN0 or DIR0
DN1	DIR1	When pwm-1-pwmdir is TRUE, this pin is the DIR output
	OUT13	XOR'd with DN1 or DIR1
DN2	DIR2	When pwm-2-pwmdir is TRUE, this pin is the DIR output
	OUT15	XOR'd with DN2 or DIR2
DN3	DIR3	When pwm-3-pwmdir is TRUE, this pin is the DIR output
	OUT17	XOR'd with DN3 or DIR3
QZ0	IN8	Read same value
QZ1	IN9	Read same value
QZ2	IN10	Read same value
QZ3	IN11	Read same value
QA0	IN12	Read same value
QA1	IN13	Read same value
QA2	IN14	Read same value
QA3	IN15	Read same value
QB0	IN16	Read same value
QB1	IN17	Read same value
QB2	IN18	Read same value
QB3	IN19	Read same value

6.13.2.2. Input latching and output updating

- PWM duty cycles for each channel are updated at different times.

- Digital outputs OUT0 through OUT9 are all updated at the same time. Digital outputs OUT10 through OUT17 are updated at the same time as the pwm function they are shared with.
- Digital inputs IN0 through IN19 are all latched at the same time.
- Quadrature positions for each channel are latched at different times.

6.13.2.3. HAL Functions, Pins and Parameters

A list of all *loadrt* arguments, HAL function names, pin names and parameter names is in the manual page, *pluto_servo.9*.

6.13.2.4. Compatible driver hardware

A schematic for a 2A, 2-axis PWM servo amplifier board is available from the (the software developer). The L298 H-Bridge can be used for motors up to 4A (one motor per L298) or up to 2A (two motors per L298) with the supply voltage up to 46V. However, the L298 does not have built-in current limiting, a problem for motors with high stall currents. For higher currents and voltages, some users have reported success with International Rectifier's integrated high-side/low-side drivers.

6.13.3. Pluto Step

Pluto-step is suitable for control of a 3- or 4-axis CNC mill with stepper motors. The large number of inputs allows for a full set of limit switches.

The board features:

- 4 *step+direction* channels with 312.5 kHz maximum step rate, programmable step length, space, and direction change times
- 14 dedicated digital outputs
- 16 dedicated digital inputs
- EPP communication with the PC

6.13.3.1. Asignación de pines

- *STEPx* - The *step* (clock) output of stepgen channel *x*
- *DIRx* - The *direction* output of stepgen channel *x*
- *INx* - Dedicated digital input #x
- *OUTx* - Dedicated digital output #x
- *GND* - Ground
- *VCC* - +3.3V regulated DC

While the *extended main connector* has a superset of signals usually found on a Step & Direction DB25 connector—4 step generators, 9 inputs, and 6 general-purpose outputs—the layout on this header is different than the layout of a standard 26-pin ribbon cable to DB25 connector.

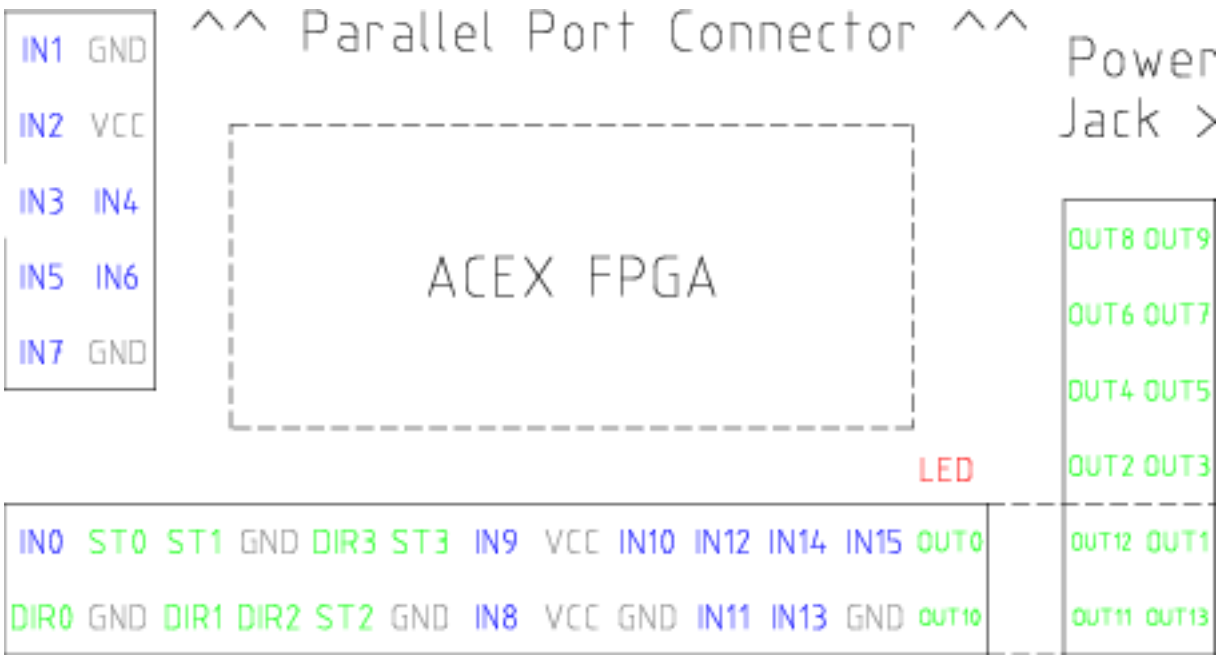


Figura 6.10: Pluto-Step Pinout

6.13.3.2. Input latching and output updating

- Step frequencies for each channel are updated at different times.
- Digital outputs are all updated at the same time.
- Digital inputs are all latched at the same time.
- Feedback positions for each channel are latched at different times.

6.13.3.3. Step Waveform Timings

The firmware and driver enforce step length, space, and direction change times. Timings are rounded up to the next multiple of 1.6µs, with a maximum of 49.6µs. The timings are the same as for the software stepgen component, except that *dirhold* and *dirsetup* have been merged into a single parameter *dirtime* which should be the maximum of the two, and that the same step timings are always applied to all channels.

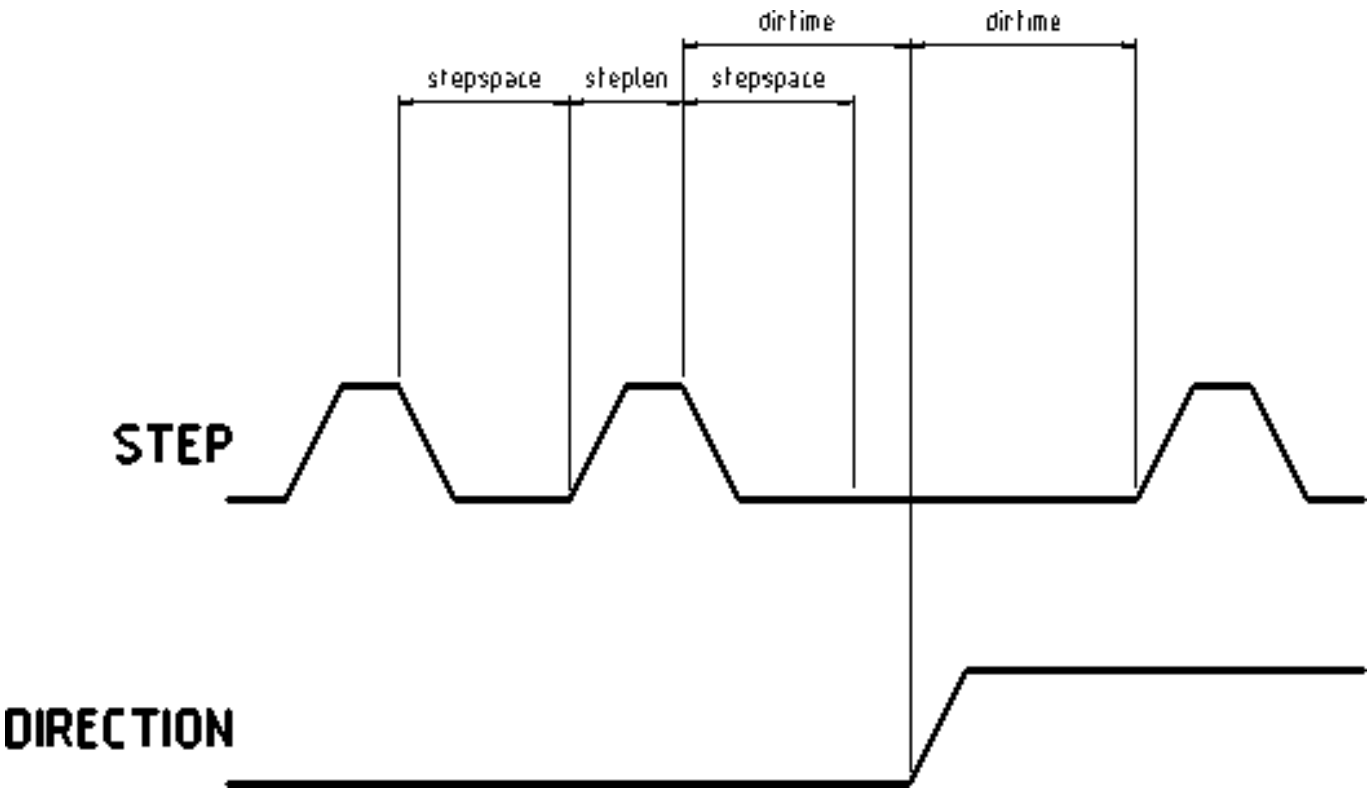


Figura 6.11: Pluto-Step Timings

6.13.3.4. HAL Functions, Pins and Parameters

A list of all *loadrt* arguments, HAL function names, pin names and parameter names is in the manual page, *pluto_step.9*.

6.14. Powermax Modbus Driver

This is a non-realtime HAL program, written in python, to control Hypetherm Powermax plasma cutters using the Modbus ASCII protocol over RS485.

---

nota
Since this is a non-realtime program it can be affected by computer loading and latency. It is possible to lose communications which will be indicated by a change in the status output. One should always have an Estop circuit that kills the power to the unit in case of emergency.

This component is loaded using the halcmd ”loadusr” command:

```
loadusr -Wn pmx485 pmx485 /dev/ttyUSB0
```

This will load the pmx485 component using the /dev/ttyUSB0 port and wait for it to become ready.

It is necessary to name the port to use for communications.

6.14.1. Pines

- **pmx485.mode-set** (bit, in) # set cutting mode
- **pmx485.current-set** (bit, in) # set cutting current
- **pmx485.pressure-set** (bit, in) # set gas pressure
- **pmx485.enable** (bit, in) # enable the component
- **pmx485.mode** (bit, out) # cut mode feedback
- **pmx485.current** (bit, out) # cutting current feedback
- **pmx485.pressure** (bit, out) # gas pressure feedback
- **pmx485.fault** (bit, out) # powermax fault code
- **pmx485.status** (bit, out) # connection status
- **pmx485.current-min** (bit, out) # minimum allowed current
- **pmx485.current-max** (bit, out) # maximum allowed current
- **pmx485.pressure-min** (bit, out) # minimum allowed gas pressure
- **pmx485.pressure-max** (bit, out) # maximum allowed gas pressure

6.14.2. Descripción

To communicate with a Powermax, the component must first be enabled via the **enable** pin and it may then initiate a request to the Powermax by writing a valid string to the following pins:

- **mode-set**
- **current-set**
- **pressure-set**

---

nota
A **pressure-set** value of zero is valid, the Powermax will then calculate the required pressure internally.

Communications may be validated from the Powermax display or the **status** pin. While in remote mode, the mode, current and pressure may be changed as needed.

To terminate the communications, do one of the following:

- Set all set pins to zero: **mode-set**, **current-set**, and **pressure-set**.
- Disconnect the Powermax power supply from its power source for approximately 30 seconds. When you power the system back ON, it will no longer be in remote mode.

6.14.3. Referencia:

- Hypertherm Application Note #807220
  ”Powermax45 XP/65/85/105/125® Serial Communication Protocol”

6.15. Servo To Go Driver

The Servo-To-Go (STG) is one of the first PC motion control cards supported by LinuxCNC. It is an ISA card and it exists in different flavors (all supported by this driver). The board includes up to 8 channels of quadrature encoder input, 8 channels of analog input and output, 32 bits digital I/O, an interval timer with interrupt and a watchdog.

nota
We have had reports that the opamps on the Servo To Go card do not work with newer ATX power supplies that use modern switch mode DC-DC converters. The failure mode is that STG card outputs a constant voltage regardless of what the driver is commanding it to do. Older ATX power supplies with linear voltage regulators do not have this problem, and work fine with the STG cards.

6.15.1. Instalación

```
loadrt hal_stg [base=<address>] [num_chan=<nr>] [dio="<dio-string>"] \
                  [model=<model>]
```

The base address field is optional; if it's not provided the driver attempts to autodetect the board. The num_chan field is used to specify the number of channels available on the card, if not used the 8 axis version is assumed. The digital inputs/outputs configuration is determined by a config string passed to insmod when loading the module. The format consists of a four character string that sets the direction of each group of pins. Each character of the direction string is either "I" or "O". The first character sets the direction of port A (Port A - DIO.0-7), the next sets port B (Port B - DIO.8-15), the next sets port C (Port C - DIO.16-23), and the fourth sets port D (Port D - DIO.24-31). The model field can be used in case the driver doesn't autodetect the right card version.

HINT: after starting up the driver, *dmesg* can be consulted for messages relevant to the driver (e.g. autodetected version number and base address). For example:

```
loadrt hal_stg base=0x300 num_chan=4 dio="IOIO"
```

This example installs the STG driver for a card found at the base address of 0x300, 4 channels of encoder feedback, DACs and ADCs, along with 32 bits of I/O configured like this: the first 8 (Port A) configured as Input, the next 8 (Port B) configured as Output, the next 8 (Port C) configured as Input, and the last 8 (Port D) configured as Output

```
loadrt hal_stg
```

This example installs the driver and attempts to autodetect the board address and board model, it installs 8 axes by default along with a standard I/O setup: Port A & B configured as Input, Port C & D configured as Output.

6.15.2. Pines

- *stg.<channel>.counts* - (s32) Tracks the counted encoder ticks.
- *stg.<channel>.position* - (float) Outputs a converted position.
- *stg.<channel>.dac-value* - (float) Drives the voltage for the corresponding DAC.
- *stg.<channel>.adc-value* - (float) Tracks the measured voltage from the corresponding ADC.
- *stg.in-<pinnum>* - (bit) Tracks a physical input pin.

- *stg.in-<pinnum>-not* - (bit) Tracks a physical input pin, but inverted.
- *stg.out-<número-de-pin>* - (bit) Maneja un pin de salida física

For each pin, <channel> is the axis number, and <pinnum> is the logic pin number of the STG if `II00` is defined, there are 16 input pins (in-00 .. in-15) and 16 output pins (out-00 .. out-15), and they correspond to PORTs ABCD (in-00 is PORTA.0, out-15 is PORTD.7).

El pin HAL in-<número-de-pin> es VERDADERO si el pin físico es alto, y FALSO si el pin físico es bajo. El pin HAL in-<número-de-pin>-not está invertido y es FALSO si el pin físico es alto. El usuario puede determinar el estado de la entrada conectando una señal a uno u otro.

6.15.3. Parámetros

- *stg.<channel>.position-scale* - (float) The number of counts / user unit (to convert from counts to units).
- *stg.<channel>.dac-offset* - (float) Sets the offset for the corresponding DAC.
- *stg.<channel>.dac-gain* - (float) Sets the gain of the corresponding DAC.
- *stg.<channel>.adc-offset* - (float) Sets the offset of the corresponding ADC.
- *stg.<channel>.adc-gain* - (float) Sets the gain of the corresponding ADC.
- *stg.out-<pinnum>-invert* - (bit) Inverts an output pin.

The -invert parameter determines whether an output pin is active high or active low. If -invert is FALSE, setting the HAL out- pin TRUE drives the physical pin high, and FALSE drives it low. If -invert is TRUE, then setting the HAL out- pin TRUE will drive the physical pin low.

6.15.4. Funciones

- *stg.capture-position* - Reads the encoder counters from the axis <channel>.
- *stg.write-dacs* - Writes the voltages to the DACs.
- *stg.read-adcs* - Reads the voltages from the ADCs.
- *stg.di-read* - Reads physical in- pins of all ports and updates all HAL in-<pinnum> and in-<pinnum>-not pins.
- *stg.do-write* - Reads all HAL out-<pinnum> pins and updates all physical output pins.

6.16. Shuttle

6.16.1. Descripción

Shuttle is a non-realtime HAL component that interfaces Contour Design’s ShuttleXpress, ShuttlePRO, and ShuttlePRO2 devices with LinuxCNC’s HAL.

If the driver is started without command-line arguments, it will probe all /dev/hidraw* device files for Shuttle devices, and use all devices found. If it is started with command-line arguments, it will only probe the devices specified.

The ShuttleXpress has five momentary buttons, a 10 counts/revolution jog wheel with detents, and a 15-position spring-loaded outer wheel that returns to center when released.

The ShuttlePRO has 13 momentary buttons, a 10 counts/revolution jog wheel with detents, and a 15-position spring-loaded outer wheel that returns to center when released.

The ShuttlePRO2 has 15 momentary buttons, a 10 counts/revolution jog wheel with detents, and a 15-position spring-loaded outer wheel that returns to center when released.



aviso

The Shuttle devices have an internal 8-bit counter for the current jog-wheel position. The shuttle driver can not know this value until the Shuttles device sends its first event. When the first event comes into the driver, the driver uses the device's reported jog-wheel position to initialize counts to 0.

This means that if the first event is generated by a jog-wheel move, that first move will be lost. Any user interaction with the Shuttle device will generate an event, informing the driver of the jog-wheel position. So if you (for example) push one of the buttons at startup, the jog-wheel will work fine and notice the first click.

6.16.2. Setup

The shuttle driver needs read permission to the /dev/hidraw* device files. This can be accomplished by adding a file /etc/udev/rules.d/99-shuttle.rules, with the following contents:

```
SUBSYSTEM==”hidraw”, ATTRS{idVendor}==”0b33”, ATTRS{idProduct}==”0020”, MODE=”0444”
SUBSYSTEM==”hidraw”, ATTRS{idVendor}==”05f3”, ATTRS{idProduct}==”0240”, MODE=”0444”
SUBSYSTEM==”hidraw”, ATTRS{idVendor}==”0b33”, ATTRS{idProduct}==”0030”, MODE=”0444”
```

The LinuxCNC Debian package installs an appropriate udev file automatically, but if you are building LinuxCNC from source and are not using the Debian packaging you'll need to install this file by hand. If you install the file by hand you'll need to tell udev to reload its rules files by running `udevadm control --reload-rules`.

6.16.3. Pines

All HAL pin names are prefixed with `shuttle` followed by the index of the device (the order in which the driver found them), for example `shuttle.0` or `shuttle.2`.

<*Prefix*>.button-<ButtonNumber> (bit out)
: These pins are True (1) when the button is pressed.

<*Prefix*>.button-<ButtonNumber>-not (bit out)
: These pins have the inverse of the button state, so they're True (1) when the button is not pressed.

<*Prefix*>.counts (s32 out)
: Accumulated counts from the jog wheel (the inner wheel).

<*Prefix*>.spring-wheel-s32 (s32 out)
: The current deflection of the spring-wheel (the outer wheel). It's 0 at rest, and ranges from -7 at the counter-clockwise extreme to +7 at the clockwise extreme.

<*Prefix*>.spring-wheel-f (float out)
: The current deflection of the spring-wheel (the outer wheel). It's 0.0 at rest, -1.0 at the counter-clockwise extreme, and +1.0 at the clockwise extreme. The Shuttle devices report the spring-wheel position as an integer from -7 to +7, so this pin reports only 15 discrete values in it's range.

6.17. VFS11 VFD Driver

This is a non-realtime HAL program to control the S11 series of VFDs from Toshiba.

vfs11_vfd supports serial and TCP connections. Serial connections may be RS232 or RS485. RS485 is supported in full- and half-duplex mode. TCP connections may be passive (wait for incoming connection), or active outgoing connections, which may be useful to connect to TCP-based devices or through a terminal server.

Regardless of the connection type, vfs11_vfd operates as a Modbus master.

This component is loaded using the halcmd ”loadusr” command:

```
loadusr -Wn spindle-vfd vfs11_vfd -n spindle-vfd
```

The above command says: loadusr, wait for named to load, component vfs11_vfd, named spindle-vfd

6.17.1. Command Line Options

vfs11_vfd is mostly configured through INI file options. The command line options are:

- *-n or --name <halname>* : set the HAL component name
- *-I or --ini <inifilename>* : take configuration from this INI file. Defaults to environment variable INI_FILE_NAME.
- *-S or --section <section name>* : take configuration from this section in the INI file. Defaults to *VFS11*.
- *-d or --debug* enable debug messages on console output.
- *-m or --modbus-debug* enable modbus messages on console output
- *-r or --report-device* report device properties on console at startup

Debugging can be toggled by sending a USR1 signal to the vfs11_vfd process. Modbus debugging can be toggled by sending a USR2 signal to vfs11_vfd process (example: `kill -USR1 ` `pidof vfs11_vfd`).

---

nota
That if there are serial configuration errors, turning on verbose may result in a flood of timeout errors.

6.17.2. Pines

Where <n> is `vfs11_vfd` or the name given during loading with the -n option.

- *<n>.acceleration-pattern* (bit, in) when true, set acceleration and deceleration times as defined in registers F500 and F501 respectively. Used in PID loops to choose shorter ramp times to avoid oscillation.
- *<n>.alarm-code* (s32, out) non-zero if drive is in alarmed state. Bitmap describing alarm information (see register FC91 description). Use err-reset (see below) to clear the alarm.
- *<n>.at-speed* (bit, out) when drive is at commanded speed (see speed-tolerance below)
- *<n>.current-load-percentage* (float, out) reported from the VFD

- *<n>.dc-brake* (bit, in) engage the DC brake. Also turns off spindle-on.
- *<n>.enable* (bit, in) enable the VFD. If false, all operating parameters are still read but control is released and panel control is enabled (subject to VFD setup).
- *<n>.err-reset* (bit, in) reset errors (alarms a.k.a Trip and e-stop status). Resetting the VFD may cause a 2-second delay until it's rebooted and Modbus is up again.
- *<n>.estop* (bit, in) put the VFD into emergency-stopped status. No operation possible until cleared with err-reset or powercycling.
- *<n>.frequency-command* (float, out) current target frequency in Hz as set through speed-command (which is in RPM), from the VFD
- *<n>.frequency-out* (float, out) current output frequency of the VFD
- *<n>.inverter-load-percentage* (float, out) current load report from VFD
- *<n>.is-e-stopped* (bit, out) the VFD is in emergency stop status (blinking "E" on panel). Use err-reset to reboot the VFD and clear the e- stop status.
- *<n>.is-stopped* (bit, out) true when the VFD reports 0 Hz output
- *<n>.max-rpm* (float, R) actual RPM limit based on maximum frequency the VFD may generate, and the motors nameplate values. For instance, if nameplate-HZ is 50, and nameplate-RPM_ is 1410, but the VFD may generate up to 80 Hz, then max-rpm would read as 2256 (80*1410/50). The frequency limit is read from the VFD at startup. To increase the upper frequency limit, the UL and FH parameters must be changed on the panel. See the VF-S11 manual for instructions how to set the maximum frequency.
- *<n>.modbus-ok* (bit, out) true when the Modbus session is successfully established and the last 10 transactions returned without error.
- *<n>.motor-RPM* (float, out) estimated current RPM value, from the VFD
- *<n>.output-current-percentage* (float, out) from the VFD
- *<n>.output-voltage-percentage* (float, out) from the VFD
- *<n>.output-voltage* (float, out) from the VFD
- *<n>.speed-command* (float, in) speed sent to VFD in RPM. It is an error to send a speed faster than the Motor Max RPM as set in the VFD
- *<n>.spindle-fwd* (bit, in) 1 for FWD and 0 for REV, sent to VFD
- *<n>.spindle-on* (bit, in) 1 for ON and 0 for OFF sent to VFD, only on when running
- *<n>.spindle-rev* (bit, in) 1 for ON and 0 for OFF, only on when running
- *<n>.jog-mode* (bit, in) 1 for ON and 0 for OFF, enables the VF-S11 *jog mode*. Speed control is disabled, and the output frequency is determined by register F262 (preset to 5 Hz). This might be useful for spindle orientation. In normal mode, the VFD shuts off if the frequency drops below 12 Hz.
- *<n>.status* (s32, out) Drive Status of the VFD (see the TOSVERT VF-S11 Communications Function Instruction Manual, register FD01). A bitmap.
- *<n>.trip-code* (s32, out) trip code if VF-S11 is in tripped state.
- *<n>.error-count* (s32, out) number of Modbus transactions which returned an error
- *<n>.max-speed* (bit, in) ignore the loop-time parameter and run Modbus at maximum speed, at the expense of higher CPU usage. Suggested use during spindle positioning.

6.17.3. Parámetros

Where <n> is `vfs11_vfd` or the name given during loading with the -n option.

- *<n>.frequency-limit* (float, RO) upper limit read from VFD setup.
- *<n>.loop-time* (float, RW) how often the Modbus is polled (default interval 0.1 seconds)
- *<n>.nameplate-HZ* (float, RW) Nameplate Hz of motor (default 50). Used to calculate target frequency (together with nameplate-RPM) for a target RPM value as given by speed-command.
- *<n>.nameplate-RPM* (float, RW) Nameplate RPM of motor (default 1410)
- *<n>.rpm-limit* (float, RW) do-not-exceed soft limit for motor RPM (defaults to nameplate-RPM).
- *<n>.tolerance* (float, RW) speed tolerance (default 0.01) for determining whether spindle is at speed (0.01 meaning: Output frequency is within 1 % of target frequency)

6.17.4. INI file configuration

This lists all options understood by vfs11_vfd. Typical setups for RS-232, RS-485 and TCP can be found in *src/hal/user_comps/vfs11_vfd/*.ini*.

```
[VFS11]
# serial connection
TYPE=rtu

# serial port
DEVICE=/dev/ttyS0

# TCP server - wait for incoming connection
TYPE=tcpserver

# tcp portnumber for TYPE=tcpserver or tcpclient
PORT=1502

# TCP client - active outgoing connection
TYPE=tcpclient

# destination to connect to if TYPE=tcpclient
TCPDEST=192.168.1.1

#---------- meaningful only if TYPE=rtu -------
# serial device detail
# 5 6 7 8
BITS= 5

# even odd none
PARITY=none

# 110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
BAUD=19200

# 1 2
STOPBITS=1

#rs232 rs485
SERIAL_MODE=rs485

# up down none
# this feature might not work with a stock Ubuntu
```

```
# libmodbus5/libmodbus-dev package, and generate a warning
# execution will continue as if RTS_MODE=up were given.
RTS_MODE=up
#---------------------

# modbus timers in seconds
# inter-character timer
BYTE_TIMEOUT=0.5
# packet timer
RESPONSE_TIMEOUT=0.5

# target modbus ID
TARGET=1

# on I/O failure, try to reconnect after sleeping
# for RECONNECT_DELAY seconds
RECONNECT_DELAY=1

# misc. parameters
DEBUG=10
MODBUS_DEBUG=0
POLLCYCLES=10
```

6.17.5. Ejemplo HAL

```
#
# example usage of the VF-S11 VFD driver
#
#
loadusr -Wn spindle-vfd vfs11_vfd -n spindle-vfd

# conecte el pin de dirección del husillo al VFD
net vfs11-fwd spindle-vfd.spindle-fwd <= spindle.0.forward
net vfs11-rev spindle-vfd.spindle-rev <= spindle.0.reverse

# connect the spindle on pin to the VF-S11
net vfs11-run spindle-vfd.spindle-on <= spindle.0.on

# connect the VF-S11 at speed to the motion at speed
net vfs11-at-speed spindle.0.at-speed <= spindle-vfd.at-speed

# connect the spindle RPM to the VF-S11
net vfs11-RPM spindle-vfd.speed-command <= spindle.0.speed-out

# connect the VF-S11 DC brake
# since this draws power during spindle off, the dc-brake pin would
# better be driven by a monoflop which triggers on spindle-on falling edge
#net vfs11-spindle-brake spindle.N.brake => spindle-vfd.dc-brake

# to use the VFS11 jog mode for spindle orient
# see orient.9 and motion.9
net spindle-orient spindle.0.orient spindle-vfd.max-speed spindle-vfd.jog-mode

# take precedence over control panel
setp spindle-vfd.enable 1
```

6.17.6. Panel operation

The vfs11_vfd driver takes precedence over panel control while it is enabled (see *enable* pin), effectively disabling the panel. Clearing the *enable* pin re-enables the panel. Pins and parameters can still be set, but will not be written to the VFD untile the *enable* pin is set. Operating parameters are still read while bus control is disabled. Exiting the vfs11_vfd driver in a controlled way will release the VFD from the bus and restore panel control.

See the LinuxCNC Integrators Manual for more information. For a detailed register description of the Toshiba VFDs, see the ”TOSVERT VF-S11 Communications Function Instruction Manual” (Toshiba document number E6581222) and the ”TOSVERT VF-S11 Instruction manual” (Toshiba document number E6581158).

6.17.7. Error Recovery

`vfs11_vfd` recovers from I/O errors as follows: First, all HAL pins are set to default values, and the driver will sleep for `RECONNECT_DELAY` seconds (default 1 second).

- Serial (`TYPE=rtu`) mode: on error, close and reopen the serial port.
- TCP server (`TYPE=tcpserver`) mode: on losing the TCP connection, the driver will go back to listen for incoming connections.
- TCP client (`TYPE=tcpclient`) mode: on losing the TCP connection, the driver will reconnect to *TCP-DEST:PORTNO*.

6.17.8. Configuring the VFS11 VFD for Modbus usage

6.17.8.1. Connecting the Serial Port

The VF-S11 has an RJ-45 jack for serial communication. Unfortunately, it does not have a standard RS-232 plug and logic levels. The Toshiba-recommended way is: connect the USB001Z USB-to-serial conversion unit to the drive, and plug the USB port into the PC. A cheaper alternative is a homebrew interface (hints from Toshiba support, circuit diagram).

Note: the 24V output from the VFD has no short-circuit protection.

Serial port factory defaults are 9600/8/1/even, the protocol defaults to the proprietary ”Toshiba Inverter Protocol”.

6.17.8.2. Modbus setup

Several parameters need setting before the VF-S11 will talk to this module. This can either be done manually with the control panel, or over the serial link - Toshiba supplies a Windows application called *PCM001Z* which can read/set parameters in the VFD. Note - PCM001Z only talks the Toshiba inverter protocol. So the last parameter which you’d want to change is the protocol - set from Toshiba Inverter Protocol to Modbus; thereafter, the Windows app is useless.

To increase the upper frequency limit, the UL and FH parameters must be changed on the panel. I increased them from 50 to 80.

See dump-params.mio for a description of non-standard VF-S11 parameters of my setup. This file is for the modio Modbus interactive utility.

6.17.9. Programming Note

The vfs11_vfd driver uses the libmodbus version 3 library which is more recent than the version 2 code used in `gs2_vfd`.

The Ubuntu `libmodbus5` and `libmodbus-dev` packages are only available starting from Ubuntu 12 (*Precise Pengolin*). Moreover, these packages lack support for the MODBUS_RTS_MODE_* flags. Therefore, building vfs11_vfd using this library might generate a warning if RTS_MODE= is specified in the INI file.

To use the full functionality on lucid and precise:

- remove the libmodbus packages: `sudo apt-get remove libmodbus5 libmodbus-dev`
- build and install libmodbus version 3 from source as outlined here.

Libmodbus does not build on Ubuntu Hardy, hence vfs11_vfd is not available on Hardy.

Capítulo 7

Ejemplos de Hardware

7.1. Puerto paralelo PCI

Cuando agregas un segundo puerto paralelo a tu bus PCI, debes encontrar la dirección antes de poder usarlo con LinuxCNC.

Para encontrar la dirección de tu tarjeta de puerto paralelo, abre una ventana de terminal y escribe

```
lspci -v
```

Verás algo similar a esto, así como información sobre todo lo demás en el bus PCI:

```
0000:00:10.0 Communication controller: \
        NetMos Technology PCI 1 port parallel adapter (rev 01)
        Subsystem: LSI Logic / Symbios Logic: Unknown device 0010
        Flags: medium devsel, IRQ 11
        I/O ports at a800 [size=8]
        I/O ports at ac00 [size=8]
        I/O ports at b000 [size=8]
        I/O ports at b400 [size=8]
        I/O ports at b800 [size=8]
        I/O ports at bc00 [size=16]
```

En mi caso, la dirección fue la primera, así que cambié mi archivo .hal de

```
loadrt hal_parport cfg=0x378
```

a

```
loadrt hal_parport cfg=”0x378 0xa800 in”
```

(Observa las comillas dobles que rodean las direcciones.)

y luego agrega las líneas siguientes para que se pueda leer y escribir en el puerto paralelo:

```
addf parport.1.read base-thread
addf parport.1.write base-thread
```

Después de hacer lo anterior, ejecuta tu configuración y verifica que el puerto paralelo se cargó en la ventana Máquina/Mostrar Configuración Hal.

7.2. Spindle Control

LinuxCNC can control up to 8 spindles. The number is set in the INI file. The examples below all refer to a single-spindle config with spindle control pins with names like *spindle.0...* In the case of a multiple spindle machine all that changes is that additional pins exist with names such as *spindle.6...*

7.2.1. 0-10 Volt Spindle Speed

If your spindle speed is controlled by an analog signal, (for example, by a VFD with a 0 V to 10 V signal) and you're using a DAC card like the m5i20 to output the control signal:

Primero debe calcular la escala de la velocidad del husillo para controlar la señal, p.ej. el voltaje. Para este ejemplo, la velocidad máxima del husillo de 5000 RPM es igual a 10 voltios.

$$\frac{10\,Volts}{5000\,RPM} = \frac{0.002\,Volts}{1\,RPM}$$

Tenemos que agregar un componente de escala al archivo HAL para escalar spindle.N.speed-out de 0 a 10 que necesita el VFD si su DAC de la tarjeta no hace escalado.

```
loadrt scale count=1
addf scale.0 servo-thread
setp scale.0.gain 0.002
net spindle-speed-scale spindle.0.speed-out => scale.0.in
net spindle-speed-DAC scale.0.out => <su nombre de pin DAC>
```

7.2.2. PWM Spindle Speed

Si el husillo puede ser controlado por una señal PWM, usar el componente *pwmgen* para crear la señal:

```
loadrt pwmgen output_type=0
addf pwmgen.update servo-thread
addf pwmgen.make-pulses base-thread
net spindle-speed-cmd spindle.0.speed-out => pwmgen.0.value
net spindle-on spindle.0.on => pwmgen.0.enable
net spindle-pwm pwmgen.0.pwm => parport.0.pin-09-out
# Establecer la velocidad máxima del husillo en RPM
setp pwmgen.0.scale 1800
```

This assumes that the spindle controller's response to PWM is simple: 0 % PWM gives 0 RPM, 10 % PWM gives 180 RPM, etc. If there is a minimum PWM required to get the spindle to turn, follow the example in the nist-lathe sample configuration to use a scale component.

7.2.3. Spindle Enable

If you need a spindle enable signal, link your output pin to *spindle.0.on*. To link these pins to a parallel port pin put something like the following in your .hal file, making sure you pick the pin that is connected to your control device.

```
net spindle-enable spindle.0.on => parport.0.pin-14-out
```

7.2.4. Spindle Direction

If you have direction control of your spindle, then the HAL pins *spindle*.N.*forward* and *spindle*.N.*reverse* are controlled by the G-codes M3 and M4. Spindle speed S*n* must be set to a positive non-zero value for M3/M4 to turn on spindle motion.

To link these pins to a parallel port pin, put something like the following in your .hal file making sure you pick the pin that is connected to your control device.

```
net spindle-fwd spindle.0.forward => parport.0.pin-16-out
net spindle-rev spindle.0.reverse => parport.0.pin-17-out
```

7.2.5. Spindle Soft Start

If you need to ramp your spindle speed command and your control does not have that feature it can be done in HAL. Basically you need to hijack the output of *spindle*.N.*speed-out* and run it through a *limit2* component with the scale set so it will ramp the rpm from *spindle*.N.*speed-out* to your device that receives the rpm. The second part is to let LinuxCNC know when the spindle is at speed so motion can begin.

In the 0-10 Volt example the line

```
net spindle-speed-scale spindle.0.speed-out => scale.0.in
```

is changed as shown in the following example:

Intro to HAL components limit2 and near In case you have not run across them before, here's a quick introduction to the two HAL components used in the following example.

- A *limit2* is a HAL component (floating point) that accepts an input value and provides an output that has been limited to a max/min range, and also limited to not exceed a specified rate of change.
- A *near* is a HAL component (floating point) with a binary output that says whether two inputs are approximately equal.

More info is available in the documentation for HAL components, or from the man pages, just say *man limit2* or *man near* in a terminal.

```
# load the real time modules limit2 and near with names so it is easier to follow their ↩
    connections
loadrt limit2 names=spindle-ramp
loadrt near names=spindle-at-speed

# add the functions to a thread
addf spindle-ramp servo-thread
addf spindle-at-speed servo-thread

# set the parameter for max rate-of-change
# (max spindle accel/decel in units per second)
setp spindle-ramp.maxv 60

# hijack the spindle speed out and send it to spindle ramp in
net spindle-cmd <= spindle.0.speed-out => spindle-ramp.in

# the output of spindle ramp is sent to the scale in
net spindle-ramped <= spindle-ramp.out => scale.0.in

# to know when to start the motion we send the near component
# (named spindle-at-speed) to the spindle commanded speed from
```

```
# the signal spindle-cmd and the actual spindle speed
# provided your spindle can accelerate at the maxv setting.
net spindle-cmd => spindle-at-speed.in1
net spindle-ramped => spindle-at-speed.in2

# the output from spindle-at-speed is sent to spindle.0.at-speed
# and when this is true motion will start
net spindle-ready <= spindle-at-speed.out => spindle.0.at-speed
```

7.2.6. Spindle Feedback

7.2.6.1. Movimiento sincronizado del husillo

Spindle feedback is needed by LinuxCNC to perform any spindle coordinated motions like threading and constant surface speed. LinuxCNC can perform synchronized motion and CSS with any of up to 8 spindles. Which spindles are used is controlled from G-code. CSS is possible with several spindles simultaneously.

The StepConf Wizard can perform the connections for a single-spindle configuration for you if you select Encoder Phase A and Encoder Index as inputs.

Hardware assumptions for this example:

- Un codificador está conectado al husillo y emite 100 pulsos por revolución en la fase A.
- La fase A del codificador está conectada al pin 10 del puerto paralelo.
- El pulso de índice del codificador está conectado al pin 11 del puerto paralelo.

Basic Steps to add the components and configure them: [1] [2] [3]

```
# Add the encoder to HAL and attach it to threads.
loadrt encoder num_chan=4
addf encoder.update-counters base-thread
addf encoder.capture-position servo-thread

# Set the HAL encoder to 100 pulses per revolution.
setp encoder.3.position-scale 100

# Set the HAL encoder to non-quadrature simple counting using A only.
setp encoder.3.counter-mode true

# Connect the HAL encoder outputs to LinuxCNC.
net spindle-position encoder.3.position => spindle.0.revs
net spindle-velocity encoder.3.velocity => spindle.0.speed-in
net spindle-index-enable encoder.3.index-enable <=> spindle.0.index-enable

# Connect the HAL encoder inputs to the real encoder.
net spindle-phase-a encoder.3.phase-A <= parport.0.pin-10-in
net spindle-phase-b encoder.3.phase-B
net spindle-index encoder.3.phase-Z <= parport.0.pin-11-in
```

[1] In this example, we will assume that some encoders have already been issued to axes/joints 0, 1, and 2. So the next encoder available for us to attach to the spindle would be number 3. Your situation may differ.

[2] The HAL encoder index-enable is an exception to the rule in that it behaves as both an input and an output, see the Encoder Section for details

[3] It is because we selected *non-quadrature simple counting...* above that we can get away with *quadrature* counting without having any B quadrature input.

7.2.6.2. Spindle At Speed

To enable LinuxCNC to wait for the spindle to be at speed before executing a series of moves, the spindle.*N*.at-speed needs to turn true at the moment the spindle is at the commanded speed. To achieve this you need spindle feedback from an encoder. Since the feedback and the commanded speed are not usually *exactly* the same you should to use the *near* component to determine that the two numbers are close enough.

The connections needed are from the spindle velocity command signal to near.n.in1 and from the spindle velocity from the encoder to near.n.in2. Then the near.n.out is connected to spindle.*N*.at-speed. The near.n.scale needs to be set to say how close the two numbers must be before turning on the output. Depending on your setup you may need to adjust the scale to work with your hardware.

Lo siguiente son las adiciones típicas necesarias para su archivo HAL para habilitar Spindle At Speed. Si ya tiene near en su archivo HAL, aumente count y ajuste el código para adaptarlo. Asegúrese de que los nombres de las señales sean los mismos en su archivo HAL.

```
# load a near component and attach it to a thread
loadrt near
addf near.0 servo-thread

# connect one input to the commanded spindle speed
net spindle-cmd => near.0.in1

# connect one input to the encoder-measured spindle speed
net spindle-velocity => near.0.in2

# connect the output to the spindle-at-speed input
net spindle-at-speed spindle.0.at-speed <= near.0.out

# set the spindle speed inputs to agree if within 1%
setp near.0.scale 1.01
```

7.3. MPG Pendant

This example is to explain how to hook up the common MPG pendants found on the market today. This example uses an MPG3 pendant and a C22 pendant interface card from CNC4PC connected to a second parallel port plugged into the PCI slot. This example gives you 3 axes with 3 step increments of 0.1, 0.01, 0.001

In your custom.hal file or jog.hal file add the following, making sure you don’t have mux4 or an encoder already in use. If you do just increase the counts and change the reference numbers. More information about mux4 and encoder can be found in the HAL manual or the man page.

See the INI HAL Section of the documentation for more information on adding a HAL file. Jog management pins are provided for each joint and all coordinate letters. This example uses the axis jog pins for jogging in world mode. Machines with non-identity kinematics may need use additional connections for jogging in joint mode.

jog.hal

```
# Jog Pendant
loadrt encoder num_chan=1
loadrt mux4 count=1
addf encoder.capture-position servo-thread
addf encoder.update-counters base-thread
addf mux4.0 servo-thread

# If your MPG outputs a quadrature signal per click set x4 to 1
```

```
# If your MPG puts out 1 pulse per click set x4 to 0
setp encoder.0.x4-mode 0

# For velocity mode, set to 1
# In velocity mode the axis stops when the dial is stopped
# even if that means the commanded motion is not completed,
# For position mode (the default), set to 0
# In position mode the axis will move exactly jog-scale
# units for each count, regardless of how long that might take,
setp axis.x.jog-vel-mode 0
setp axis.y.jog-vel-mode 0
setp axis.z.jog-vel-mode 0

# This sets the scale that will be used based on the input to the mux4
setp mux4.0.in0 0.1
setp mux4.0.in1 0.01
setp mux4.0.in2 0.001

# The inputs to the mux4 component
net scale1 mux4.0.sel0 <= parport.1.pin-09-in
net scale2 mux4.0.sel1 <= parport.1.pin-10-in

# The output from the mux4 is sent to each axis jog scale
net mpg-scale <= mux4.0.out
net mpg-scale => axis.x.jog-scale
net mpg-scale => axis.y.jog-scale
net mpg-scale => axis.z.jog-scale

# Las entradas MPG
net mpg-a encoder.0.phase-A <= parport.1.pin-02-in
net mpg-b encoder.0.phase-B <= parport.1.pin-03-in

# The Axis select inputs
net mpg-x axis.x.jog-enable <= parport.1.pin-04-in
net mpg-y axis.y.jog-enable <= parport.1.pin-05-in
net mpg-z axis.z.jog-enable <= parport.1.pin-06-in

# The encoder output counts to the axis. Only the selected axis will move.
net encoder-counts  <= encoder.0.counts
net encoder-counts => axis.x.jog-counts
net encoder-counts => axis.y.jog-counts
net encoder-counts => axis.z.jog-counts
```

If the machine is capable of high acceleration to smooth out the moves for each click of the MPG use the ilowpass component to limit the acceleration.

jog.hal with ilowpass

```
loadrt encoder num_chan=1
loadrt mux4 count=1
addf encoder.capture-position servo-thread
addf encoder.update-counters base-thread
addf mux4.0 servo-thread

loadrt ilowpass
addf ilowpass.0 servo-thread

setp ilowpass.0.scale 1000
setp ilowpass.0.gain 0.01

# If your MPG outputs a quadrature signal per click set x4 to 1
# If your MPG puts out 1 pulse per click set x4 to 0
```

```
setp encoder.0.x4-mode 0

# For velocity mode, set to 1
# In velocity mode the axis stops when the dial is stopped
# even if that means the commanded motion is not completed,
# For position mode (the default), set to 0
# In position mode the axis will move exactly jog-scale
# units for each count, regardless of how long that might take,
setp axis.x.jog-vel-mode 0
setp axis.y.jog-vel-mode 0
setp axis.z.jog-vel-mode 0

# This sets the scale that will be used based on the input to the mux4
# The scale used here has to be multiplied by the ilowpass scale
setp mux4.0.in0 0.0001
setp mux4.0.in1 0.00001
setp mux4.0.in2 0.000001

# The inputs to the mux4 component
net scale1 mux4.0.sel0 <= parport.1.pin-09-in
net scale2 mux4.0.sel1 <= parport.1.pin-10-in

# The output from encoder counts is sent to ilowpass
net mpg-out ilowpass.0.in <= encoder.0.counts

# The output from the mux4 is sent to each axis jog scale
net mpg-scale <= mux4.0.out
net mpg-scale => axis.x.jog-scale
net mpg-scale => axis.y.jog-scale
net mpg-scale => axis.z.jog-scale

# Las entradas MPG
net mpg-a encoder.0.phase-A <= parport.1.pin-02-in
net mpg-b encoder.0.phase-B <= parport.1.pin-03-in

# The Axis select inputs
net mpg-x axis.x.jog-enable <= parport.1.pin-04-in
net mpg-y axis.y.jog-enable <= parport.1.pin-05-in
net mpg-z axis.z.jog-enable <= parport.1.pin-06-in

# The output from the ilowpass is sent to each axis jog count
# Only the selected axis will move.
net encoder-counts  <= ilowpass.0.out
net encoder-counts => axis.x.jog-counts
net encoder-counts => axis.y.jog-counts
net encoder-counts => axis.z.jog-counts
```

7.4. GS2 Spindle

7.4.1. Ejemplo

This example shows the connections needed to use an Automation Direct GS2 VFD to drive a spindle. The spindle speed and direction is controlled by LinuxCNC.

Using the GS2 component involves very little to set up. We start with a StepConf Wizard generated config. Make sure the pins with ”Spindle CW” and ”Spindle PWM” are set to unused in the parallel port setup screen.

In the custom.hal file we place the following to connect LinuxCNC to the GS2 and have LinuxCNC control the drive.

GS2 Example

```
# load the non-realtime component for the Automation Direct GS2 VFDs
loadusr -Wn spindle-vfd gs2_vfd -r 9600 -p none -s 2 -n spindle-vfd

# connect the spindle direction pin to the GS2
net gs2-fwd spindle-vfd.spindle-fwd <= spindle.N.forward

# connect the spindle on pin to the GS2
net gs2-run spindle-vfd.spindle-on <= spindle.N.on

# connect the GS2 at speed to the motion at speed
net gs2-at-speed spindle.N.at-speed <= spindle-vfd.at-speed

# connect the spindle RPM to the GS2
net gs2-RPM spindle-vfd.speed-command <= spindle.N.speed-out
```

nota
The transmission speed might be able to be faster depending on the exact environment. Both the drive and the command line options must match. To check for transmission errors add the *-v* command line option and run from a terminal.

On the GS2 drive itself you need to set a couple of things before the modbus communications will work. Other parameters might need to be set based on your physical requirements but these are beyond the scope of this manual. Refer to the GS2 manual that came with the drive for more information on the drive parameters.

- Los interruptores de comunicación deben configurarse en RS-232C.
- Los parámetros del motor deben configurarse para que coincidan con el motor.
- P3.00 (Fuente del comando de operación) debe establecerse en Operación determinada por la interfaz RS-485, 03 o 04.
- P4.00 (Fuente del comando de frecuencia) debe establecerse en Frecuencia determinada por la interfaz de comunicación RS232C/RS485, 05.
- P9.01 (Velocidad de transmisión) debe establecerse en 9600 baudios, 01.
- P9.02 (Protocolo de comunicación) debe establecerse en ”Modo Modbus RTU, 8 bits de datos, sin paridad, 2 bits de parada ”, 03.

A PyVCP panel based on this example is here.

Capítulo 8

Ladder Clásico

8.1. ClassicLadder Introduction

8.1.1. History

ClassicLadder is a free implementation of a ladder interpreter, released under the LGPL. It was written by Marc Le Douarain.

He describes the beginning of the project on his website:

> I decided to program a ladder language only for test purposes at the start, in February 2001. It was planned, that I would have to participate to a new product after leaving the enterprise in which I was working at that time. And I was thinking that to have a ladder language in those products could be a nice option to considerate. And so I started to code the first lines for calculating a rung with minimal elements and displaying dynamically it under Gtk, to see if my first idea to realize all this works.
>
> And as quickly I've found that it advanced quite well, I've continued with more complex elements: timer, multiples rungs, etc…
>
> Voila, here is this work… and more: I've continued to add features since then.
>
> — Marc Le Douarain, from "Genesis" at the ClassicLadder website

ClassicLadder has been adapted to work with LinuxCNC's HAL, and is currently being distributed along with LinuxCNC. If there are issues/problems/bugs please report them to the LinuxCNC project.

8.1.2. Introducción

Ladder logic or the Ladder programming language is a method of drawing electrical logic schematics. It is now a graphical language very popular for programming Programmable Logic Controllers (PLCs). It was originally invented to describe logic made from relays. The name is based on the observation that programs in this language resemble ladders, with two vertical *rails* and a series of horizontal *rungs* between them. In Germany and elsewhere in Europe, the style is to draw the rails horizontally along the top and bottom of the page while the rungs are drawn vertically from left to right.

A program in ladder logic, also called a ladder diagram, is similar to a schematic for a set of relay circuits. Ladder logic is useful because a wide variety of engineers and technicians can understand and use it without much additional training because of the resemblance.

Ladder logic is widely used to program PLCs, where sequential control of a process or manufacturing operation is required. Ladder logic is useful for simple but critical control systems, or for reworking

old hardwired relay circuits. As programmable logic controllers became more sophisticated it has also been used in very complex automation systems.

Ladder logic can be thought of as a rule-based language, rather than a procedural language. A *rung* in the ladder represents a rule. When implemented with relays and other electromechanical devices, the various rules *execute* simultaneously and immediately. When implemented in a programmable logic controller, the rules are typically executed sequentially by software, in a loop. By executing the loop fast enough, typically many times per second, the effect of simultaneous and immediate execution is obtained.

Ladder logic follows these general steps for operation.

- Read Inputs
- Solve Logic
- Update Outputs

8.1.3. Ejemplo

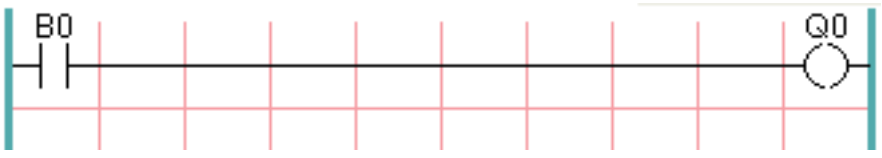
The most common components of ladder are contacts (inputs), these usually are either NC (normally closed) or NO (normally open), and coils (outputs).

- the NO contact 
- the NC contact 
- the coil (output) 

Of course there are many more components to a full ladder language, but understanding these will help you grasp the overall concept.

The ladder consists of one or more rungs. These rungs are horizontal traces (representing wires), with components on them (inputs, outputs and other), which get evaluated left to right.

This example is the simplest rung:



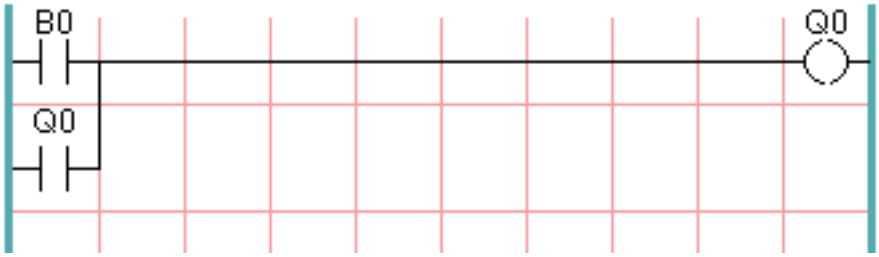
The input on the left, B0, a normally open contact, is connected to the coil (output) on the right, Q0. Now imagine a voltage gets applied to the leftmost end, because the input B0 turns true (e.g. the input is activated, or the user pushed the NO contact). The voltage has a direct path to reach the coil (output) on the right, Q0. As a consequence, the Q0 coil (output) will turn from 0/off/false to 1/on/true. If the user releases B0, the Q0 output quickly returns to 0/off/false.

8.1.4. Basic Latching On-Off Circuit

Building on the above example, suppose we add a switch that closes whenever the coil Q0 is active. This would be the case in a relay, where the coil can activate the switch contacts; or in a contactor, where there are often several small auxiliary contacts in addition to the large 3-phase contacts that are the primary feature of the contactor.

Since this auxiliary switch is driven from coil Q0 in our earlier example, we will give it the same number as the coil that drives it. This is the standard practice followed in all ladder programming, although it may seem strange at first to see a switch labeled the same as a coil. So let's call this auxiliary contact Q0 and connect it across the B0 *pushbutton* contact from our earlier example.

Let's take a look at it:

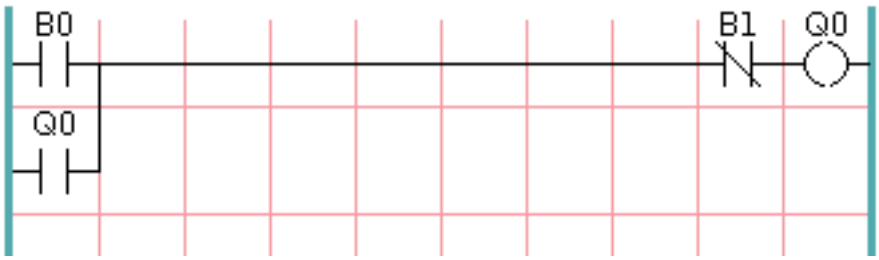


As before, when the user presses pushbutton B0, coil Q0 comes on. And when coil Q0 comes on, switch Q0 comes on. Now the interesting part happens. When the user releases pushbutton B0, coil Q0 does not stop as it did before. This is because switch Q0 of this circuit is effectively holding the user's pushbutton pressed. So we see that switch Q0 is still holding coil Q0 on after the *start* pushbutton has been released.

This type of contact on a coil or relay, used in this way, is often called a *holding contact*, because it *holds on* the coil that it is associated with. It is also occasionally called a *seal* contact, and when it is active it is said that the circuit is *sealed*.

Unfortunately, our circuit so far has little practical use, because, although we have an *on* or *start* button in the form of pushbutton B0, we have no way to shut this circuit off once it is started. But that's easy to fix. All we need is a way to interrupt the power to coil Q0. So let's add a normally-closed (NC) pushbutton just ahead of coil Q0.

Here's how that would look:



Now we have added *off* or *stop* pushbutton B1. If the user pushes it, contact from the rung to the coil is broken. When coil Q0 loses power, it drops to 0/off/false. When coil Q0 goes off, so does switch Q0, so the *holding contact* is broken, or the circuit is *unsealed*. When the user releases the *stop* pushbutton, contact is restored from the rung to coil Q0, but the rung has gone dead, so the coil doesn't come back on.

This circuit has been used for decades on virtually every machine that has a three-phase motor controlled by a contactor, so it was inevitable that it would be adopted by ladder/PLC programmers. It is also a very safe circuit, in that if *start* and *stop* are both pressed at the same time, the *stop* function always wins.

This is the basic building block of much of ladder programming, so if you are new to it, you would do well to make sure that you understand how this circuit operates.

8.2. ClassicLadder Programming

8.2.1. Ladder Concepts

ClassicLadder is a type of programming language originally implemented on industrial PLCs (it's called Ladder Programming). It is based on the concept of relay contacts and coils, and can be used to construct logic checks and functions in a manner that is familiar to many systems integrators. Ladder consists of rungs that may have branches and resembles an electrical circuit. It is important to know how ladder programs are evaluated when running.

It seems natural that each line would be evaluated left to right, then the next line down, etc., but it doesn't work this way in ladder logic. Ladder logic *scans* the ladder rungs 3 times to change the state of the outputs.

- the inputs are read and updated
- the logic is figured out
- the outputs are set

This can be confusing at first if the output of one line is read by the input of a another rung. There will be one scan before the second input becomes true after the output is set.

Another gotcha with ladder programming is the "Last One Wins" rule. If you have the same output in different locations of your ladder the state of the last one will be what the output is set to.

8.2.2. Languages

The most common language used when working with ClassicLadder is *ladder*. ClassicLadder also supports Sequential Function Chart (Grafcet).

8.2.3. Componentes

Hay dos componentes para ClassicLadder.

- The realtime module classicladder_rt
- The non-realtime module (including a GUI) classicladder

8.2.3.1. Archivos

Typically ClassicLadder components are placed in the custom.hal file if your working from a StepConf generated configuration. These must not be placed in the custom_postgui.hal file or the Ladder Editor menu will be grayed out.

---

nota
Ladder files (.clp) must not contain any blank spaces in the name.

8.2.3.2. Realtime Module

Loading the ClassicLadder real time module (classicladder_rt) is possible from a HAL file, or directly using a halcmd instruction. The first line loads real time the ClassicLadder module. The second line adds the function classicladder.0.refresh to the servo thread. This line makes ClassicLadder update at the servo thread rate.

```
loadrt classicladder_rt
addf classicladder.0.refresh servo-thread
```

The speed of the thread that ClassicLadder is running in directly affects the responsiveness to inputs and outputs. If you can turn a switch on and off faster than ClassicLadder can notice it then you may need to speed up the thread. The fastest that ClassicLadder can update the rungs is one millisecond. You can put it in a faster thread but it will not update any faster. If you put it in a slower than one millisecond thread then ClassicLadder will update the rungs slower. The current scan time will be displayed on the section display, it is rounded to microseconds. If the scan time is longer than one millisecond you may want to shorten the ladder or put it in a slower thread.

8.2.3.3. Variables

It is possible to configure the number of each type of ladder object while loading the ClassicLadder real time module. If you do not configure the number of ladder objects ClassicLadder will use the default values.

Cuadro 8.1: Conteo de variables predeterminado

Object Name	**Variable Name**	**Default Value**
Number of rungs	(numRungs)	100
Number of bits	(numBits)	20
Number of word variables	(numWords)	20
Number of timers	(numTimers)	10
Number of timers IEC	(numTimersIec)	10
Number of monostables	(numMonostables)	10
Number of counters	(numCounters)	10
Number of HAL inputs bit pins	(numPhysInputs)	15
Number of HAL output bit pins	(numPhysOutputs)	15
Number of arithmetic expressions	(numArithmExpr)	50
Number of Sections	(numSections)	10
Number of Symbols	(numSymbols)	Auto
Number of S32 inputs	(numS32in)	10
Number of S32 outputs	(numS32out)	10
Number of Float inputs	(numFloatIn)	10
Number of Float outputs	(numFloatOut)	10

Objects of most interest are numPhysInputs, numPhysOutputs, numS32in, and numS32out.

Changing these numbers will change the number of HAL bit pins available. numPhysInputs and numPhysOutputs control how many HAL bit (on/off) pins are available. numS32in and numS32out control how many HAL signed integers (+- integer range) pins are available.

For example (you don’t need all of these to change just a few):

```
loadrt classicladder_rt numRungs=12 numBits=100 numWords=10
numTimers=10 numMonostables=10 numCounters=10 numPhysInputs=10
numPhysOutputs=10 numArithmExpr=100 numSections=4 numSymbols=200
numS32in=5 numS32out=5
```

To load the default number of objects:

```
loadrt classicladder_rt
```

8.2.4. Loading the ClassicLadder non-realtime module

ClassicLadder HAL commands must executed before the GUI loads or the menu item Ladder Editor will not function. If you used the Stepper Config Wizard place any ClassicLadder HAL commands in the custom.hal file.

Para cargar el módulo en tiempo no-real:

```
loadusr classicladder
```

nota
Only one .clp file can be loaded. If you need to divide your ladder then use sections.

To load a ladder file:

```
loadusr classicladder myladder.clp
```

ClassicLadder Loading Options

- *--nogui* - (loads without the ladder editor) normally used after debugging is finished.
- *--modbus_port=port* - (loads the modbus port number)
- *--modmaster* - (initializes MODBUS master) should load the ladder program at the same time or the TCP is default port.
- *--modslave* - (initializes MODBUS slave) only TCP

To use ClassicLadder with HAL without EMC:

```
loadusr -w classicladder
```

The -w tells HAL not to close down the HAL environment until ClassicLadder is finished.

If you first load ladder program with the *--nogui* option then load ClassicLadder again with no options the GUI will display the last loaded ladder program.

In AXIS you can load the GUI from File/Ladder Editor...

8.2.5. ClassicLadder GUI

If you load ClassicLadder with the GUI it will display two windows: Section display, and section manager.

8.2.5.1. Sections Manager

When you first start up ClassicLadder you get an empty Sections Manager window.

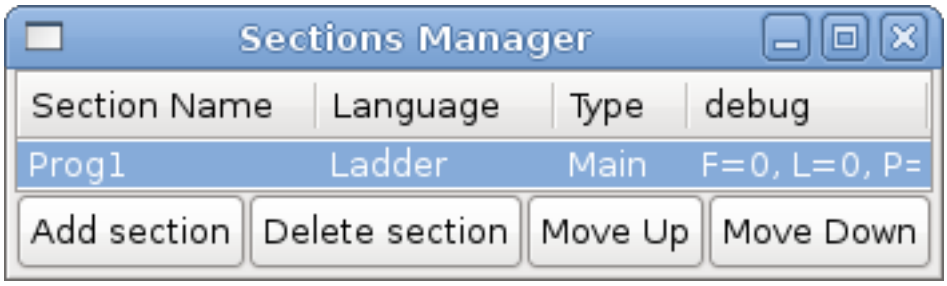


Figura 8.1: Sections Manager Default Window

This window allows you to name, create or delete sections and choose what language that section uses. This is also how you name a subroutine for call coils.

8.2.5.2. Section Display

When you first start up ClassicLadder you get an empty Section Display window. Displayed is one empty rung.

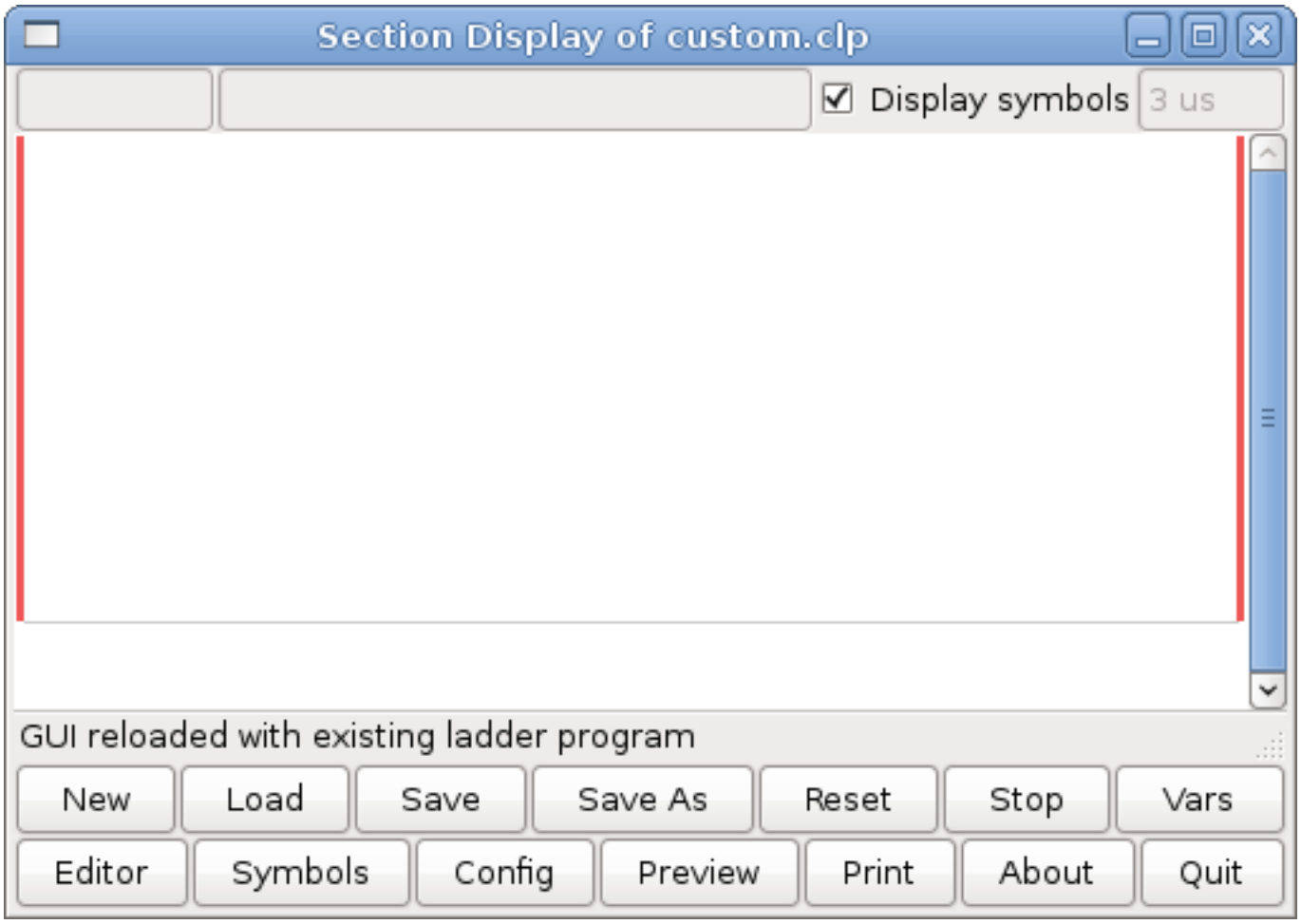


Figura 8.2: Section Display Default Window

Most of the buttons are self explanatory:

The Vars button is for looking at variables, toggle it to display one, the other, both, then none of the windows.

The Config button is used for modbus and shows the max number of ladder elements that was loaded with the real time module.

The Symbols button will display an editable list of symbols for the variables (hint you can name the inputs, outputs, coils etc).

The Quit button will shut down the non-realtime program, i.e. Modbus and the display. The realtime ladder program will still run in the background.

The check box at the top right allows you to select whether variable names or symbol names are displayed

You might notice that there is a line under the ladder program display that reads ”Project failed to load...”. That is the status bar that gives you info about elements of the ladder program that you click on in the display window. This status line will now display HAL signal names for variables %I, %Q and the first %W (in an equation). You might see some funny labels, such as (103) in the rungs. This is displayed (on purpose) because of an old bug- when erasing elements older versions sometimes didn’t erase the object with the right code. You might have noticed that the long horizontal connection button sometimes did not work in the older versions. This was because it looked for the *free* code but found something else. The number in the brackets is the unrecognized code. The ladder program will still work properly, to fix it erase the codes with the editor and save the program.

8.2.5.3. The Variable Windows

This are two variable windows: the Bit Status Window (boolean) and the Watch Window (signed integer). The Vars button is in the Section Display Window, toggle the Vars button to display one, the other, both, then none of the variable windows.

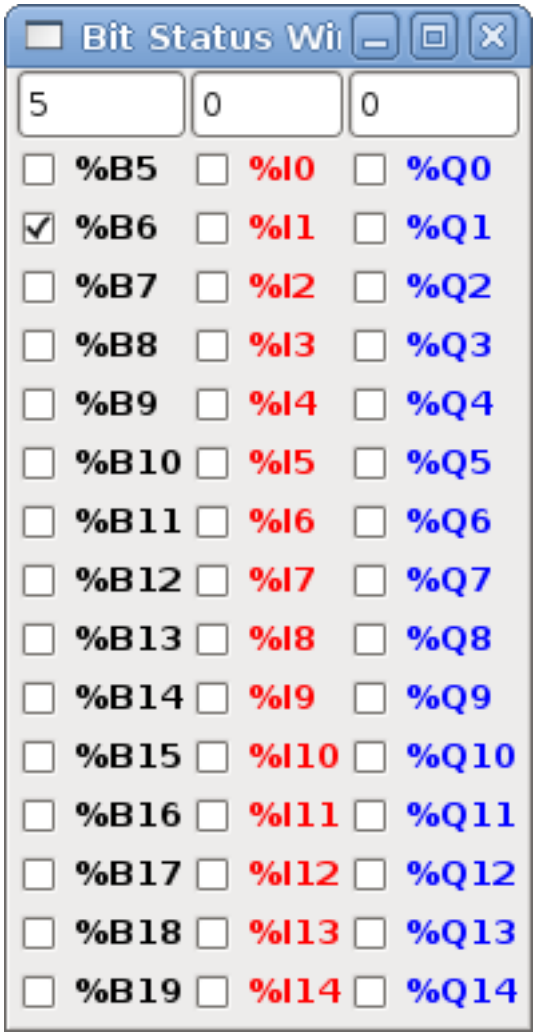


Figura 8.3: Bit Status Window

The Bit Status Window displays some of the boolean (on/off) variable data. Notice all variables start with the % sign. The %I variables represent HAL input bit pins. The %Q represents the relay coil and HAL output bit pins. The %B represents an internal relay coil or internal contact. The three edit areas at the top allow you to select what 15 variables will be displayed in each column. For instance, if the %B Variable column were 15 entries high, and you entered 5 at the top of the column, variables %B5 to %B19 would be displayed. The check boxes allow you to set and unset %B variables manually as long as the ladder program isn't setting them as outputs. Any Bits that are set as outputs by the program when ClassicLadder is running can not be changed and will be displayed as checked if on and unchecked if off.

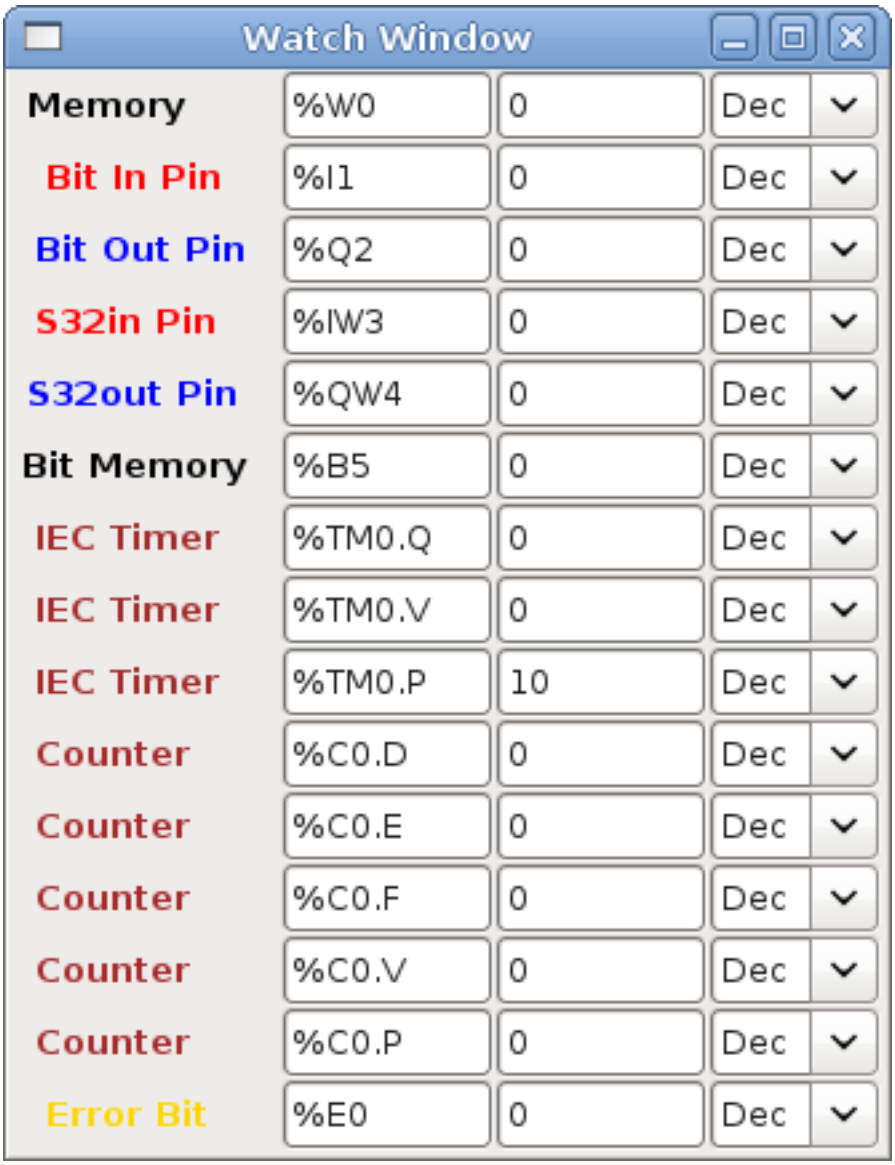


Figura 8.4: Watch Window

The Watch Window displays variable status. The edit box beside it is the number stored in the variable and the drop-down box beside that allow you to choose whether the number to be displayed in hex, decimal or binary. If there are symbol names defined in the symbols window for the word variables showing and the *display symbols* checkbox is checked in the section display window, symbol names will be displayed. To change the variable displayed, type the variable number, e.g. %W2 (if the display symbols check box is not checked) or type the symbol name (if the display symbols checkbox is checked) over an existing variable number/name and press the Enter Key.

8.2.5.4. Symbol Window

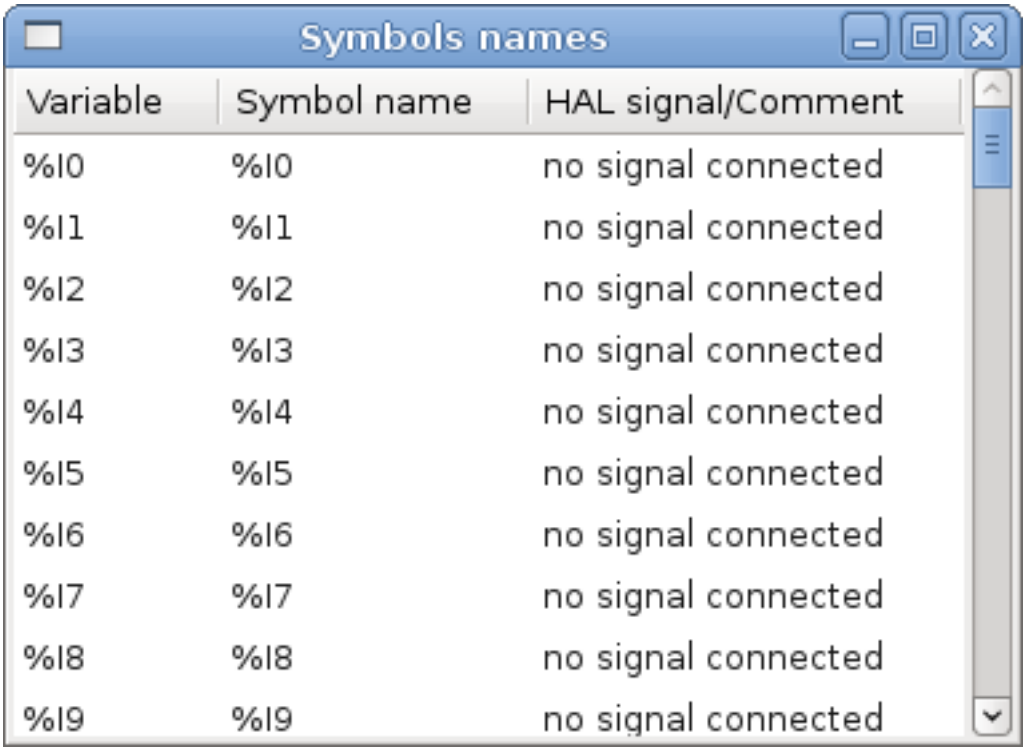


Figura 8.5: Symbol Names window

This is a list of *symbol* names to use instead of variable names to be displayed in the section window when the *display symbols* check box is checked. You add the variable name (remember the % symbol and capital letters), symbol name. If the variable can have a HAL signal connected to it (%I, %Q, and %W-if you have loaded s32 pin with the real time module) then the comment section will show the current HAL signal name or lack thereof. Symbol names should be kept short to display better. Keep in mind that you can display the longer HAL signal names of %I, %Q and %W variable by clicking on them in the section window. Between the two, one should be able to keep track of what the ladder program is connected to!

8.2.5.5. The Editor window



Figura 8.6: Editor Window

- *Add* - adds a rung after the selected rung
- *Insert* - inserts a rung before the selected rung
- *Delete* - deletes the selected rung
- *Modify* - opens the selected rung for editing

Starting from the top left image:

- Object Selector, Eraser
- N.O. Input, N.C. Input, Rising Edge Input, Falling Edge Input

- Horizontal Connection, Vertical Connection, Long Horizontal Connection
- Timer IEC Block, Counter Block, Compare Variable
- Old Timer Block, Old Monostable Block (These have been replaced by the IEC Timer)
- COILS - N.O. Output, N.C. Output, Set Output, Reset Output
- Jump Coil, Call Coil, Variable Assignment

A short description of each of the buttons:

- *Selector* - allows you to select existing objects and modify the information.
- *Eraser* - erases an object.
- *N.O. Contact* - creates a normally open contact. It can be an external HAL-pin (%I) input contact, an internal-bit coil (%B) contact or a external coil (%Q) contact. The HAL-pin input contact is closed when the HAL-pin is true. The coil contacts are closed when the corresponding coil is active (%Q2 contact closes when %Q2 coil is active).
- *N.C. Contact* - creates a normally closed contact. It is the same as the N.O. contact except that the contact is open when the HAL-pin is true or the coil is active.
- *Rising Edge Contact* - creates a contact that is closed when the HAL-pin goes from False to true, or the coil from not-active to active.
- *Falling Edge Contact* - creates a contact that is closed when the HAL-pin goes from true to false or the coil from active to not.
- *Horizontal Connection* - creates a horizontal connection to objects.
- *Vertical Connection* - creates a vertical connection to horizontal lines.
- *Horizontal Running Connection* - creates a horizontal connection between two objects and is a quick way to connect objects that are more than one block apart.
- *IEC Timer* - creates a timer and replaces the *Timer*.
- *Timer* - creates a Timer Module (depreciated use IEC Timer instead).
- *Monostable* - creates a one-shot monostable module
- *Counter* - creates a counter module.
- *Compare* - creates a compare block to compare variable to values or other variables, e.g. `%W1<=5` or `%W1=%W2`. Compare cannot be placed in the right most side of the section display.
- *Variable Assignment* - creates an assignment block so you to assign values to variables, e.g. `%W2=7` or `%W1=%W2`. ASSIGNMENT functions can only be placed at the right most side of the section display.

8.2.5.6. Config Window

The config window shows the current project status and has the Modbus setup tabs.

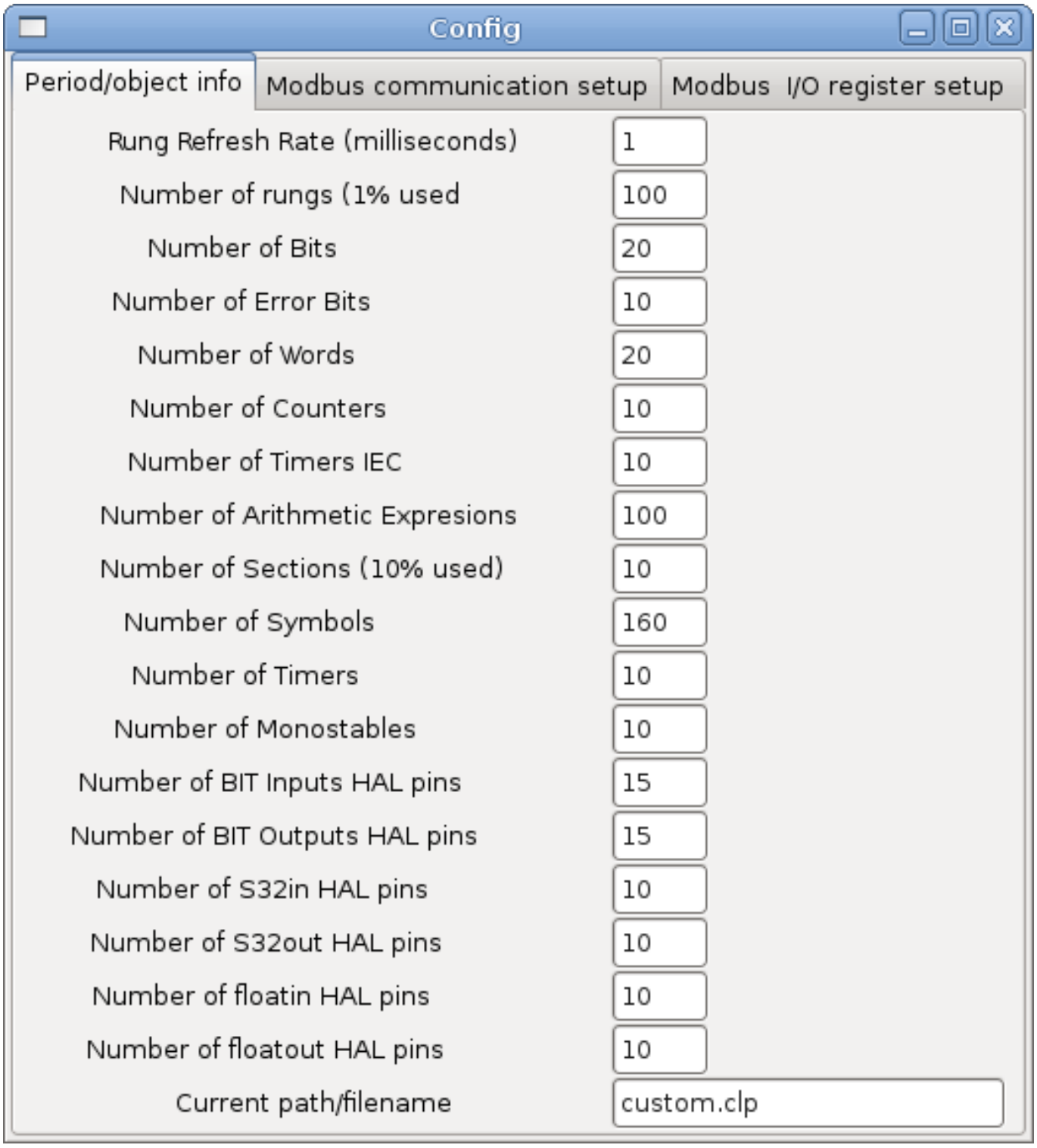


Figura 8.7: Config Window

8.2.6. Ladder objects

8.2.6.1. CONTACTS

Represent switches or relay contacts. They are controlled by the variable letter and number assigned to them.

The variable letter can be B, I, or Q and the number can be up to a three digit number, e.g. `%I2`, `%Q3`, or `%B123`. Variable I is controlled by a HAL input pin with a corresponding number. Variable B is for internal contacts, controlled by a B coil with a corresponding number. Variable Q is controlled by a Q coil with a corresponding number (like a relay with multiple contacts). E.g., if HAL pin `classicladder.0.in-00` is true then %I0 N.O. contact would be on (closed, true, whatever you like to call it). If %B7 coil is *energized* (on, true, etc) then %B7 N.O. contact would be on. If %Q1 coil is *energized* then %Q1 N.O. contact would be on (and HAL pin `classicladder.0.out-01` would be true).

- *N.O. Contact* -  (Normally Open) When the variable is false the switch is off.
- *N.C. Contact* -  (Normally Closed) When the variable is false the switch is on.
- *Rising Edge Contact* - When the variable changes from false to true, the switch is PULSED on.
- *Falling Edge Contact* - When the variable changes from true to false, the switch is PULSED on.

8.2.6.2. IEC TIMERS

Represent new count down timers. IEC Timers replace Timers and Monostables.

IEC Timers have 2 contacts.

- *I* - input contact
- *Q* - output contact

There are three modes - TON, TOF, TP.

- *TON* - When timer input is true countdown begins and continues as long as input remains true. After countdown is done and as long as timer input is still true the output will be true.
- *TOF* - When timer input is true, sets output true. When the input is false the timer counts down then sets output false.
- *TP* - When timer input is pulsed true or held true timer sets output true till timer counts down. (one-shot)

The time intervals can be set in multiples of 100&8239;ms, seconds, or minutes.

There are also Variables for IEC timers that can be read and/or written to in compare or operate blocks.

- *%TMxxx.Q* - timer done (Boolean, read write)
- *%TMxxx.P* - timer preset (read write)
- *%TMxxx.V* - timer value (read write)

8.2.6.3. TIMERS

Represent count down timers. This is deprecated and replaced by IEC Timers.

Timers have 4 contacts.

- *E* - enable (input) starts timer when true, resets when goes false
- *C* - control (input) must be on for the timer to run (usually connect to E)
- *D* - done (output) true when timer times out and as long as E remains true
- *R* - running (output) true when timer is running

The timer base can be multiples of milliseconds, seconds, or minutes.

There are also Variables for timers that can be read and/or written to in compare or operate blocks.

- *%Txx.R* - Timer xx running (Boolean, read only)
- *%Txx.D* - Timer xx done (Boolean, read only)
- *%Txx.V* - Timer xx current value (integer, read only)
- *%Txx.P* - Timer xx preset (integer, read or write)

8.2.6.4. MONOSTABLES

Represent the original one-shot timers. This is now deprecated and replaced by IEC Timers.

Monostables have 2 contacts, I and R.

- *I* - input (input) will start the mono timer running.
- *R* - running (output) will be true while timer is running.

The I contact is rising edge sensitive meaning it starts the timer only when changing from false to true (or off to on). While the timer is running the I contact can change with no effect to the running timer. R will be true and stay true till the timer finishes counting to zero. The timer base can be multiples of milliseconds, seconds, or minutes.

There are also Variables for monostables that can be read and/or written to in compare or operate blocks.

- *%Mxx.R* - Monostable xx running (Boolean, read only)
- *%Mxx.V* - Monostable xx current value (integer, read only)
- *%Mxx.P* - Monostable xx preset (integer, read or write)

8.2.6.5. COUNTERS

Represent up/down counters.

There are 7 contacts:

- *R* - reset (input) will reset the count to 0.
- *P* - preset (input) will set the count to the preset number assigned from the edit menu.
- *U* - up count (input) will add one to the count.

- *D* - down count (input) will subtract one from the count.
- *E* - under flow (output) will be true when the count rolls over from 0 to 9999.
- *D* - done (output) will be true when the count equals the preset.
- *F* - overflow (output) will be true when the count rolls over from 9999 to 0.

The up and down count contacts are edge sensitive meaning they only count when the contact changes from false to true (or off to on if you prefer).

The range is 0 to 9999.

There are also Variables for counters that can be read and/or written to in compare or operate blocks.

- '%C'*xx*.D - Counter *xx* done (*Boolean*, *read only*)
- '%C'*xx*.E - Counter *xx* empty overflow (*Boolean*, *read only*_)
- '%C'*xx*.F - Counter *xx* full overflow (*Boolean*, *read only*)
- '%C'*xx*.V - Counter *xx* current value (*integer*, *read* or *write*)
- '%C'*xx*.P - Counter *xx* preset (*integer*, *read* or *write*)

8.2.6.6. COMPARE

For arithmetic comparison. Is variable %XXX = to this number (or evaluated number)

The compare block will be true when comparison is true. You can use most math symbols:

- +, -, *, /, = (standard math symbols)
- < (less than), > (greater than), <= (less or equal), >= (greater or equal), <> (not equal)
- (,) separate into groups example %IF1=2,&%IF2<5 in pseudo code translates to if %IF1 is equal to 2 and %IF2 is less than 5 then the comparison is true. Note the comma separating the two groups of comparisons.
- ^ (exponent), % (modulus), & (and), | (or),. -
- ABS (absolute), MOY (French for average), AVG (average)

For example ABS(%W2)=1, MOY(%W1, %W2)<3.

No spaces are allowed in the comparison equation. For example %C0.V>%C0.P is a valid comparison expression while %C0.V >%CO.P is not a valid expression.

There is a list of Variables down the page that can be used for reading from and writing to ladder objects. When a new compare block is opened be sure and delete the # symbol when you enter a compare.

To find out if word variable #1 is less than 2 times the current value of counter #0 the syntax would be:

```
%W1<2*%C0.V
```

To find out if S32in bit 2 is equal to 10 the syntax would be:

```
%IW2=10
```

Note: Compare uses the arithmetic equals not the double equals that programmers are used to.

8.2.6.7. VARIABLE ASSIGNMENT

For variable assignment, e.g. assign this number (or evaluated number) to this variable %xxx, there are two math functions MINI and MAXI that check a variable for maximum (0x80000000) and minimum values (0x07FFFFFF) (think signed values) and keeps them from going beyond.

When a new variable assignment block is opened be sure to delete the # symbol when you enter an assignment.

To assign a value of 10 to the timer preset of IEC Timer 0 the syntax would be:

```
%TM0.P=10
```

To assign the value of 12 to s32out bit 3 the syntax would be:

```
%QW3=12
```

nota
When you assign a value to a variable with the variable assignment block the value is retained until you assign a new value using the variable assignment block. The last value assigned will be restored when LinuxCNC is started.

The following figure shows an Assignment and a Comparison Example. %QW0 is a S32out bit and %IW0 is a S32in bit. In this case the HAL pin `classicladder.0.s32out-00` will be set to a value of 5 and when the HAL pin `classicladder.0.s32in-00` is 0 the HAL pin `classicladder.0.out-00` will be set to True.

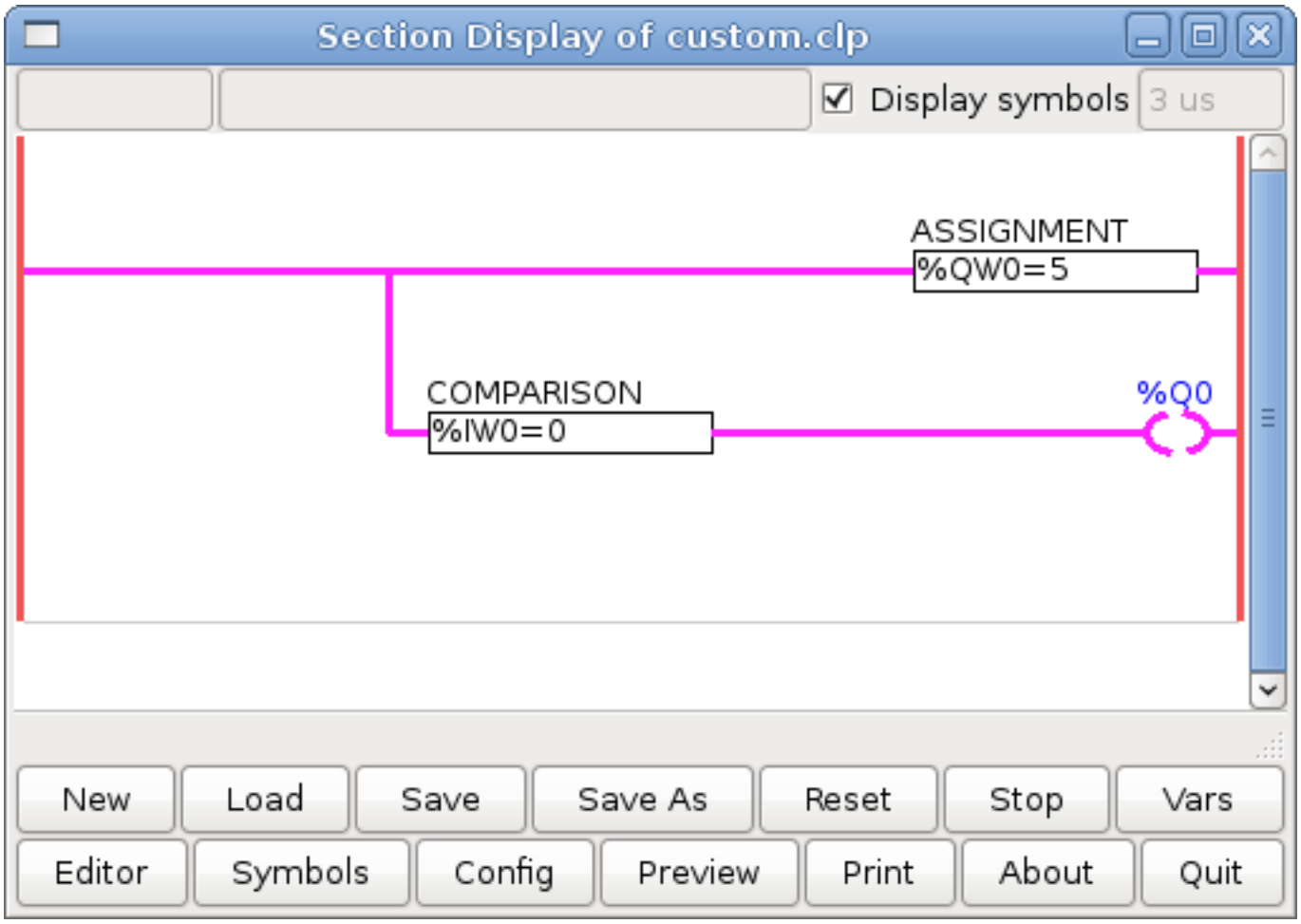


Figura 8.8: Assign/Compare Ladder Example



Figura 8.9: Assignment Expression Example



Figura 8.10: Comparison Expression Example

8.2.6.8. COILS

Coils represent relay coils. They are controlled by the variable letter and number assigned to them.

The variable letter can be B or Q and the number can be up to a three digit number, e.g., %Q3, or %B123. Q coils control HAL out pins, e.g. if %Q15 is energized then HAL pin classicladder.0.out-15 will be true. B coils are internal coils used to control program flow.

- *N.O. COIL* - A relay coil: When coil is energized, then its contact that is normally open (short: N.O.) will be closed (turned on, true, etc.) and the current may pass.
- *N.C. COIL* - A relay coil that inverses its contacts: When coil is energized, then its contact that is normally closed (short: N.C.) will be opened (turned off, false, etc) and the current flow is interrupted.
- *SET COIL* - A relay coil with latching contacts: When coil is energized then its N.O. contact will be latched closed.
- *RESET COIL* - A relay coil with latching contacts: When coil is energized then its N.O. contact will be latched open.
- *JUMP COIL* - A *goto* coil: When coil is energized then the ladder program jumps to a rung (in the CURRENT section) - jump points are designated by a rung label. (Add rung labels in the section display, top left label box.)
- *CALL COIL* - A *gosub* coil: When coil is energized then the program jumps to a subroutine section designated by a subroutine number - subroutines are designated SR0 to SR9 (designate them in the section manager).

---

aviso
If you use a N.C. contact with a N.C. coil the logic will work (when the coil is energized the contact will be closed) but that is really hard to follow!

A JUMP COIL is used to *JUMP* to another section, like a goto in BASIC programming language.

If you look at the top left of the sections display window you will see a small label box and a longer comment box beside it. Now go to Editor→Modify then go back to the little box, type in a name.

Go ahead and add a comment in the comment section. This label name is the name of this rung only and is used by the JUMP COIL to identify where to go.

When placing a JUMP COIL, add it in the rightmost position and change the label to the rung you want to JUMP to.

A CALL COIL is used to go to a subroutine section then return, like a gosub in BASIC programming language.

If you go to the sections manager window hit the add section button. You can name this section, select what language it will use (ladder or sequential), and select what type (main or subroutine).

Select a subroutine number (SR0 for example). An empty section will be displayed and you can build your subroutine.

When you've done that, go back to the section manager and click on the your main section (default name prog1).

Now you can add a CALL COIL to your program. CALL COILs are to be placed at the rightmost position in the rung.

Remember to change the label to the subroutine number you chose before.

8.2.7. ClassicLadder Variables

These Variables are used in COMPARE or OPERATE to get information about, or change specs of, ladder objects such as changing a counter preset, or seeing if a timer is done running.

Lista de variables :

- `%B`*xxx* - Bit memory *xxx* (Boolean)
- `%W`*xxx* - Word memory *xxx* (32 bits signed integer)
- `%IW`*xxx* - Word memory *xxx* (S32 in pin)
- `%QW`*xxx* - Word memory *xxx* (S32 out pin)
- `%IF`*xx* - Word memory *xx* (Float in pin) (**converted to S32 in ClassicLadder**)
- `%QF`*xx* - Word memory *xx* (Float out pin) (**converted to S32 in ClassicLadder**)
- `%T _xx_.R` - Timer *xx* running (Boolean, user read only)
- `%T _xx_.D` - Timer *xx* done (Boolean, user read only)
- `%T _xx_.V` - Timer *xx* current value (integer, user read only)
- `%T _xx_.P` - Timer *xx* preset (integer)
- `%TM _xxx_.Q` - Timer *xxx* done (Boolean, read write)
- `%TM _xxx_.P` - Timer *xxx* preset (integer, read write)
- `%TM _xxx_.V` - Timer *xxx* value (integer, read write)
- `%M _xx_.R` - Monostable *xx* running (Boolean)
- `%M _xx_.V` - Monostable *xx* current value (integer, user read only)
- `%M _xx_.P` - Monostable *xx* preset (integer)
- `%C _xx_.D` - Counter *xx* done (Boolean, user read only)
- `%C _xx_.E` - Counter *xx* empty overflow (Boolean, user read only)

- %C `__xx__.F` - Counter *xx* full overflow (Boolean, user read only)
- %C `__xx__.V` - Counter *xx* current value (integer)
- %C `__xx__.P` - Counter *xx* preset (integer)
- %I*xxx* - Physical input *xxx* (Boolean) (HAL input bit)
- %Q*xxx* - Physical output *xxx* (Boolean) (HAL output bit)
- %X*xxx* - Activity of step *xxx* (sequential language)
- %X `__xxx__.V` - Time of activity in seconds of step *xxx* (sequential language)
- %E*xx* - Errors (Boolean, read write(will be overwritten))
- *Indexed or vectored variables* - These are variables indexed by another variable. Some might call this vectored variables. Example: %W0[%W4] => if %W4 equals 23 it corresponds to %W23

8.2.8. GRAFCET (State Machine) Programming



aviso
This is probably the least used and most poorly understood feature of ClassicLadder. Sequential programming is used to make sure a series of ladder events always happen in a prescribed order. Sequential programs do not work alone. There is always a ladder program as well that controls the variables. Here are the basic rules governing sequential programs:

- Rule 1 : Initial situation - The initial situation is characterized by the initial steps which are by definition in the active state at the beginning of the operation. There shall be at least one initial step.
- Rule 2 : R2, Clearing of a transition - A transition is either enabled or disabled. It is said to be enabled when all immediately preceding steps linked to its corresponding transition symbol are active, otherwise it is disabled. A transition cannot be cleared unless it is enabled, and its associated transition condition is true.
- Rule 3 : R3, Evolution of active steps - The clearing of a transition simultaneously leads to the active state of the immediately following step(s) and to the inactive state of the immediately preceding step(s).
- Rule 4 : R4, Simultaneous clearing of transitions - All simultaneous cleared transitions are simultaneously cleared.
- Rule 5 : R5, Simultaneous activation and deactivation of a step - If during operation, a step is simultaneously activated and deactivated, priority is given to the activation.

This is the SEQUENTIAL editor window. (Starting from the top left):
Selector arrow, Eraser
Ordinary step, Initial (Starting) step
Transition, Step and Transition
Transition Link-Downside, Transition Link-Upside
Pass-through Link-Downside, Pass-through Link-Upside Jump
Link, Comment Box

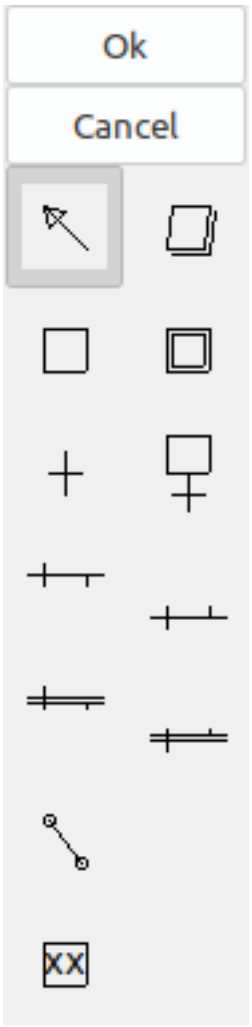


Figura 8.11: Sequence Editor Window

- *ORDINARY STEP* - has a unique number for each one
- *STARTING STEP* - a sequential program must have one. This is where the program will start.
- *TRANSITION* - shows the variable that must be true for control to pass through to the next step.
- *STEP AND TRANSITION* - combined for convenience
- *TRANSITION LINK-DOWNSIDE* - splits the logic flow to one of two possible lines based on which of the next steps is true first (Think OR logic)
- *TRANSITION LINK=UPSIDE* - combines two (OR) logic lines back in to one
- *PASS-THROUGH LINK-DOWNSIDE* - splits the logic flow to two lines that BOTH must be true to continue (Think AND logic)
- *PASS-THROUGH LINK-UPSIDE* - combines two concurrent (AND logic) logic lines back together
- *JUMP LINK* - connects steps that are not underneath each other such as connecting the last step to the first
- *COMMENT BOX* - used to add comments

To use links, you must have steps already placed. Select the type of link, then select the two steps or transactions one at a time. It takes practice!

With sequential programming: The variable `%X __xxx__ (e.g., %X5)` is used to see if a step is active. The variable `%X __xxx__.V` (e.g., `%X5.V`) is used to see how long the step has been active. The %X and %X.v variables are use in LADDER logic. The variables assigned to the transitions (e.g., %B) control whether the logic will pass to the next step. After a step has become active the transition variable that caused it to become active has no control of it anymore. The last step has to JUMP LINK back only to the beginning step.

8.2.9. Modbus

Things to consider:

- Modbus is a non-realtime program so it might have latency issues on a heavily laden computer.
- Modbus is not really suited to hard real time events such as position control of motors or to control E-stop.
- The ClassicLadder GUI must be running for Modbus to be running.
- Modbus is not fully finished so it does not do all modbus functions.

To get MODBUS to initialize you must specify that when loading the ClassicLadder non-realtime program.

Loading Modbus

```
loadusr -w classicladder --modmaster myprogram.clp
```

The `-w` makes HAL wait until you close ClassicLadder before closing realtime session. ClassicLadder also loads a TCP modbus slave if you add `--modserver` on command line.

Modbus Functions

- *1* - read coils
- *2* - read inputs
- *3* - read holding registers
- *4* - read input registers
- *5* - write single coils
- *6* - write single register
- *8* - echo test
- *15* - write multiple coils
- *16* - write multiple registers

If you do not specify a *--modmaster* when loading the ClassicLadder non-realtime program this page will not be displayed.

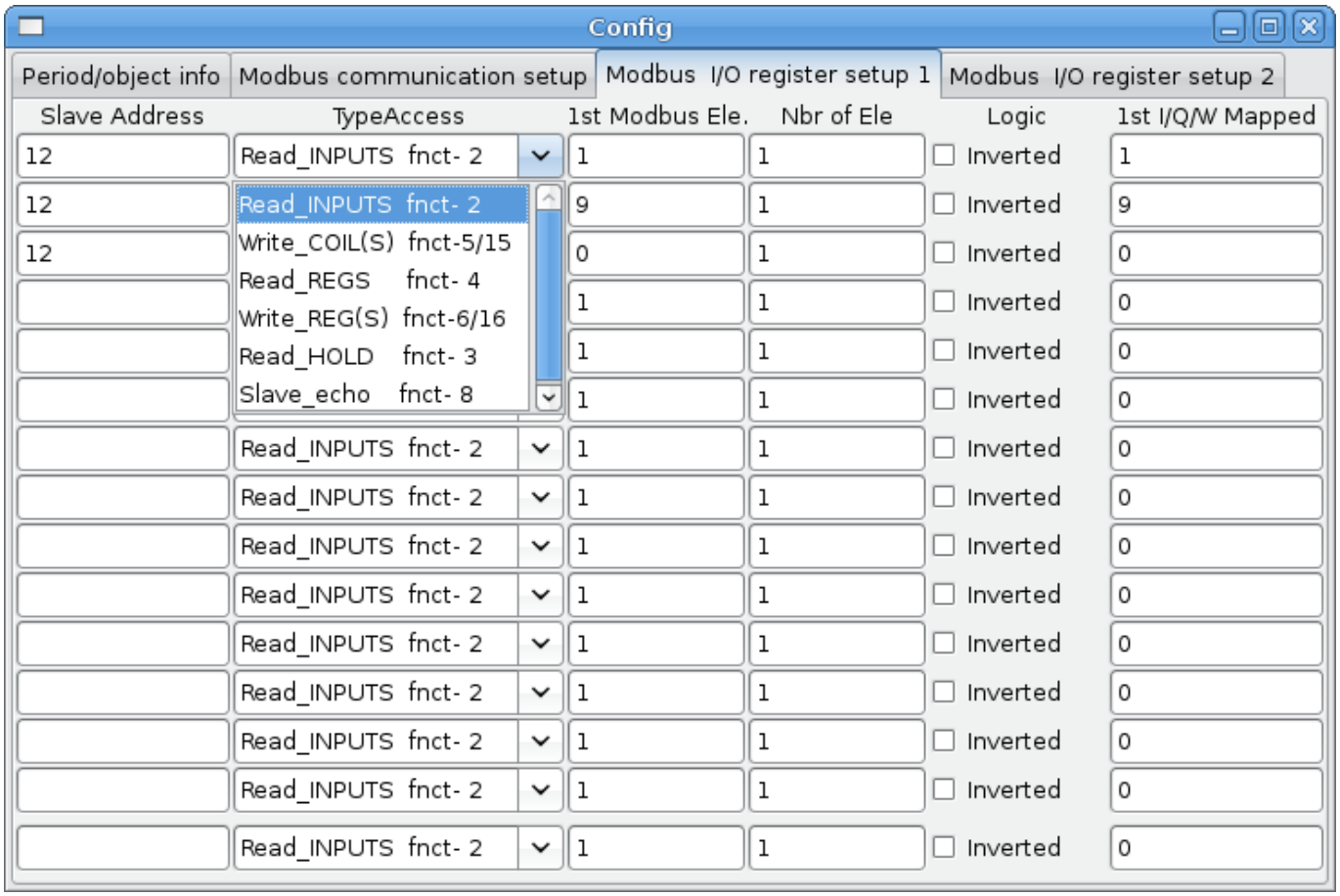


Figura 8.12: Modbus I/O Config

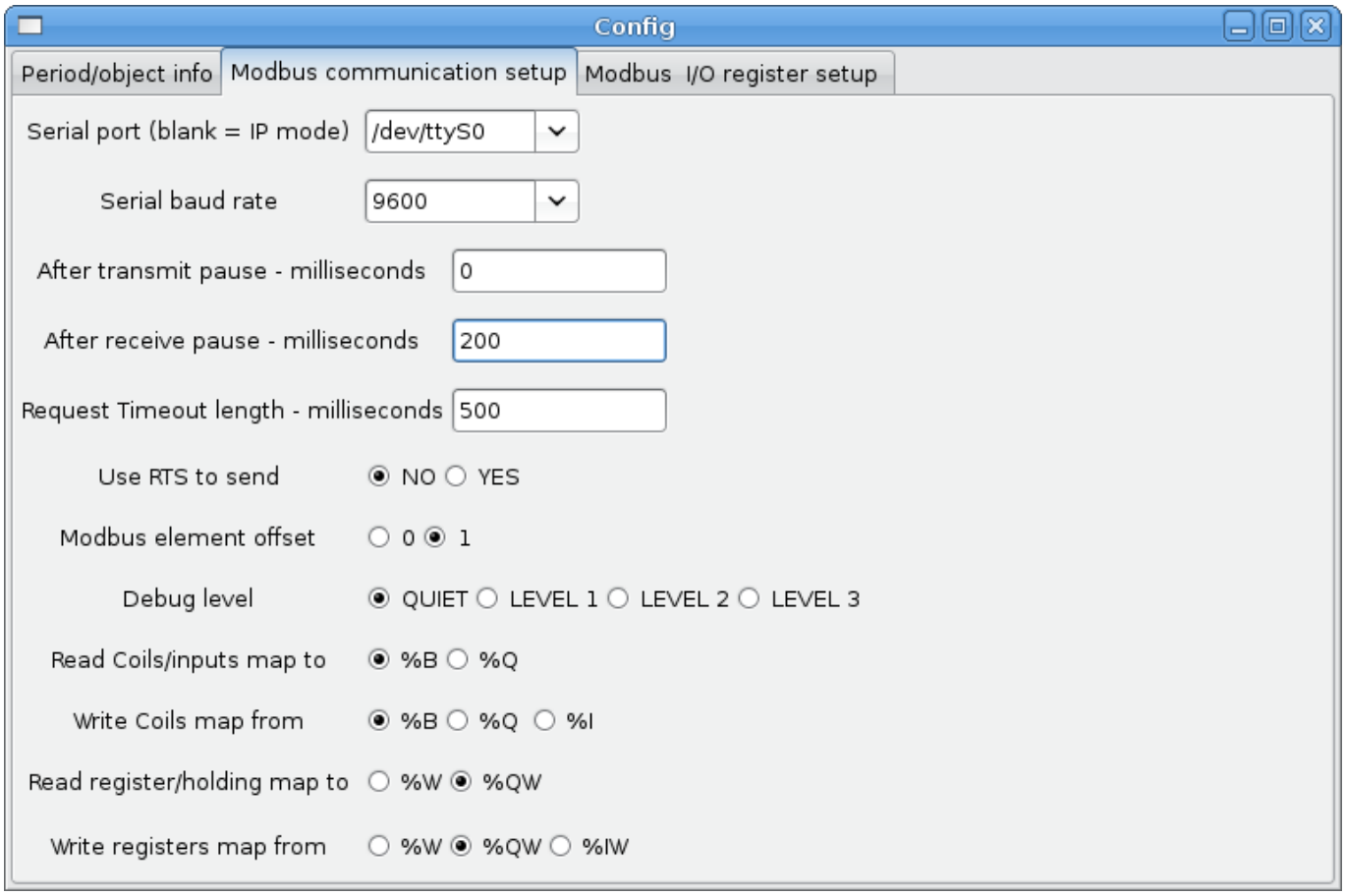


Figura 8.13: Modbus Communication Config

- *SERIAL PORT* - For IP blank. For serial the location/name of serial driver, e.g., /dev/ttyS0 (or /dev/ttyUSB0 for a USB-to-serial converter).
- *SERIAL SPEED* - Should be set to speed the slave is set for - 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 are supported.
- *PAUSE AFTER TRANSMIT* - Pause (milliseconds) after transmit and before receiving answer, some devices need more time (e.g., USB-to-serial converters).
- *PAUSE INTER-FRAME* - Pause (milliseconds) after receiving answer from slave. This sets the duty cycle of requests (it's a pause for EACH request).
- *REQUEST TIMEOUT LENGTH* - Length (milliseconds) of time before we decide that the slave didn't answer.
- *MODBUS ELEMENT OFFSET* - used to offset the element numbers by 1 (for manufacturers numbering differences).
- *DEBUG LEVEL* - Set this to 0-3 (0 to stop printing debug info besides no-response errors).
- *READ COILS/INPUTS MAP TO* - Select what variables that read coils/inputs will update. (B or Q).
- *WRITE COILS MAP TO* - Select what variables that write coils will updated from (B,Q,or I).
- *READ REGISTERS/HOLDING* - Select what variables that read registers will update (W or QW).
- *WRITE REGISTERS MAP TO* - Select what variables that read registers will updated from (W, QW, or IW).

- *SLAVE ADDRESS* - For serial the slaves ID number usually settable on the slave device (usually 1-256). For IP the slave IP address plus optionally the port number.
- *TYPE ACCESS* - This selects the MODBUS function code to send to the slave (eg what type of request).
- *COILS / INPUTS* - Inputs and Coils (bits) are read from/written to I, B, or Q variables (user selects).
- *REGISTERS (WORDS)* - Registers (Words/Numbers) map to IW, W, or QW variables (user selects).
- *1st MODBUS ELEMENT* - The address (or register number) of the first element in a group (remember to set MODBUS ELEMENT OFFSET properly).
- *NUMBER OF ELEMENTS* - The number of elements in this group.
- *LOGIC* - You can invert the logic here.
- *1st %I %Q IQ WQ MAPPED* - This is the starting number of %B, %I, %Q, %W, %IW, or %QW variables that are mapped onto/from the modbus element group (starting at the first modbus element number).

In the example above: Port number - for my computer /dev/ttyS0 was my serial port.

The serial speed is set to 9600 baud.

Slave address is set to 12 (on my VFD I can set this from 1-31, meaning I can talk to 31 VFDs maximum on one system).

The first line is set up for 8 input bits starting at the first register number (register 1). So register numbers 1-8 are mapped onto ClassicLadder's %B variables starting at %B1 and ending at %B8.

The second line is set for 2 output bits starting at the ninth register number (register 9) so register numbers 9-10 are mapped onto ClassicLadder's %Q variables starting at %Q9 ending at %Q10.

The third line is set to write 2 registers (16 bits each) starting at the 0^{th} register number (register 0), so register numbers 0-1 are mapped onto ClassicLadder's %W variables starting at %W0 ending at %W1.

It's easy to make an off-by-one error as sometimes the modbus elements are referenced starting at one rather then 0 (actually by the standard that is the way it's supposed to be!). You can use the modbus element offset radio button to help with this.

The documents for your modbus slave device will tell you how the registers are set up- there is no standard way.

The SERIAL PORT, PORT SPEED, PAUSE, and DEBUG level are editable for changes (when you close the config window values are applied, though Radio buttons apply immediately).

To use the echo function select the echo function and add the slave number you wish to test. You don't need to specify any variables.

The number 257 will be sent to the slave number you specified and the slave should send it back. You will need to have ClassicLadder running in a terminal to see the message.

8.2.10. MODBUS Settings

Serial:

- ClassicLadder uses RTU protocol (not ASCII).
- 8 data bits, No parity is used, and 1 stop bit is also known as 8-N-1.
- Baud rate must be the same for slave and master. ClassicLadder can only have one baud rate so all the slaves must be set to the same rate.
- Pause inter frame is the time to pause after receiving an answer.
- MODBUS_TIME_AFTER_TRANSMIT is the length of pause after sending a request and before receiving an answer (this apparently helps with USB converters which are slow).

8.2.10.1. MODBUS Info

- ClassicLadder can use distributed inputs/outputs on modules using the Modbus protocol (”master”: polling slaves).
- The slaves and theirs I/O can be configured in the config window.
- 2 exclusive modes are available : ethernet using Modbus/TCP and serial using Modbus/RTU.
- No se usa paridad.
- If no port name for serial is set, TCP/IP mode will be used…
- The slave address is the slave address (Modbus/RTU) or the IP address.
- The IP address can be followed per the port number to use (xx.xx.xx.xx:pppp) else the port 9502 will be used per default.
- 2 products have been used for tests: a Modbus/TCP one (Adam-6051, https://www.advantech.com) and a serial Modbus/RTU one (https://www.ipac.ws).
- See examples: adam-6051 and modbus_rtu_serial.
- Web links: https://www.modbus.org and this interesting one: https://www.iatips.com/modbus.html
- MODBUS TCP SERVER INCLUDED
- ClassicLadder has a Modbus/TCP server integrated. Default port is 9502. (the previous standard 502 requires that the application must be launched with root privileges).
- List of Modbus functions code supported are: 1, 2, 3, 4, 5, 6, 15 and 16.
- Modbus bits and words correspondence table is actually not parametric and correspond directly to the %B and %W variables.

Puede encontrar más información sobre modbus en internet.

https://www.modbus.org/

8.2.10.2. Communication Errors

If there is a communication error, a warning window will pop up (if the GUI is running) and %E0 will be true. Modbus will continue to try to communicate. The %E0 could be used to make a decision based on the error. A timer could be used to stop the machine if timed out, etc.

8.2.11. Debugging modbus problems

A good reference for the protocol: https://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf. If you run linuxcnc/classicladder from a terminal, it will print the Modbus commands and slave responses.

Here we set ClassicLadder to request slave 1, to read holding registers (function code 3) starting at address 8448 (0x2100). We ask for 1 (2 byte wide) data element to be returned. We map it to a ClassicLadder variable starting at 2.

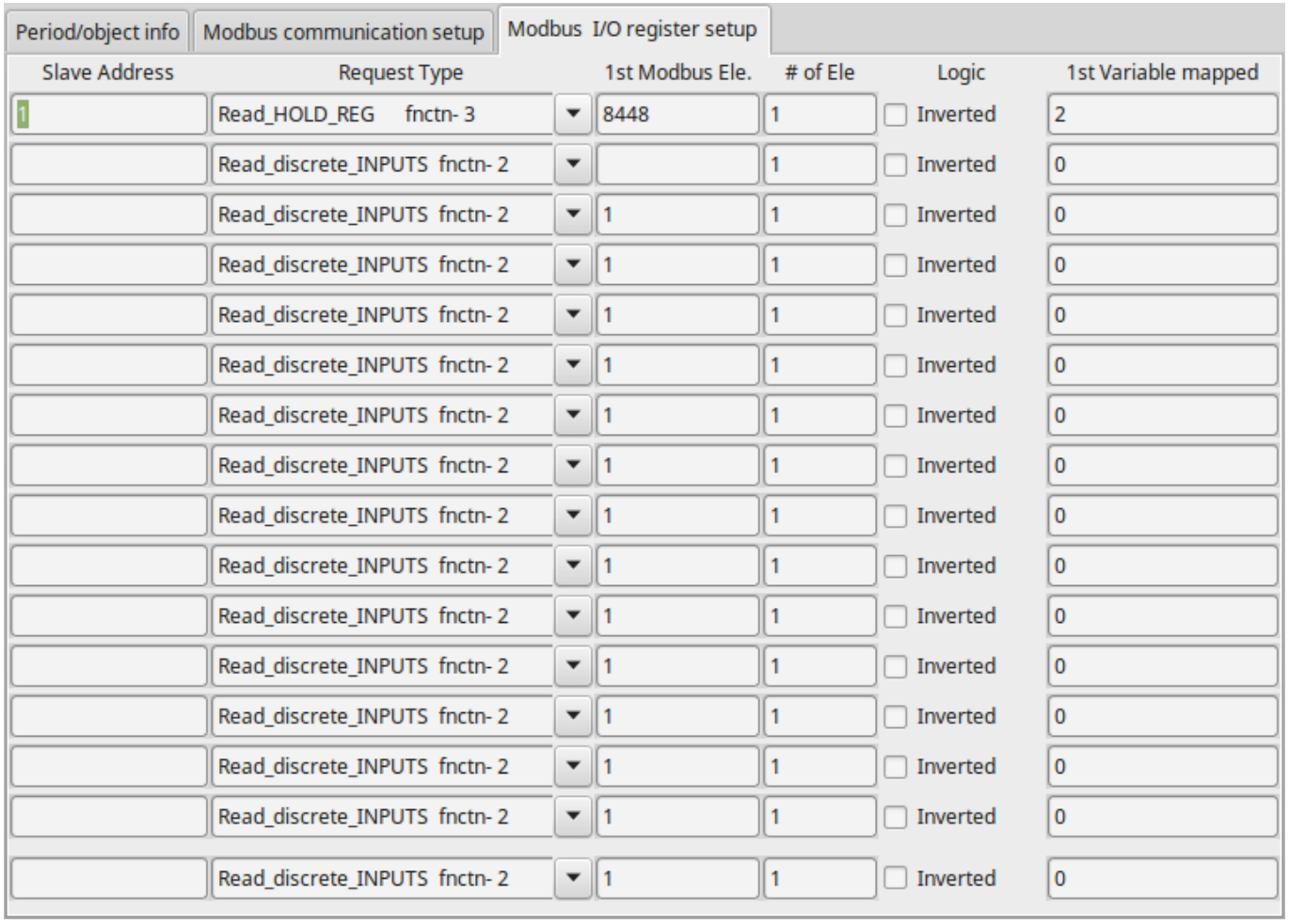


Figura 8.14: Modbus I/O Register Setup

Note in this image we have set the debug level to 1 so modbus messages are printed to the terminal. We have mapped our read and written holding registers to ClassicLadder's %W variables so our returned data will be in %W2 as in the other image we mapped the data starting at the 2nd element.

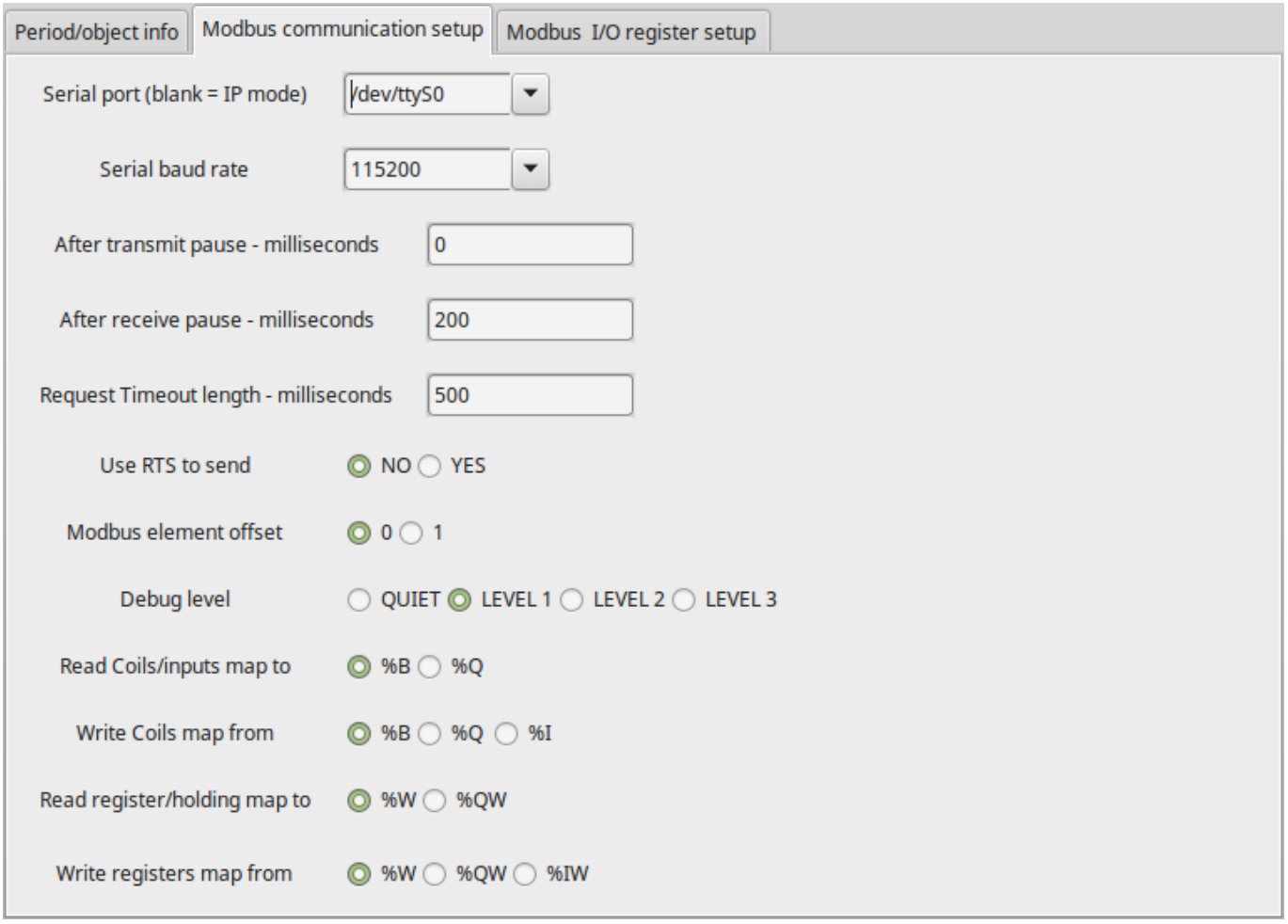


Figura 8.15: Modbus Communication Setup

8.2.11.1. Request

Lets look at an example of reading one hold register at 8448 Decimal (0x2100 Hex).
Looking in the Modbus protocol reference:

Cuadro 8.2: Read holding register request

Name	**número de bytes**	**Value (hex)**
Function code	(1 Byte)	3 (0x03)
Starting Address	(2 Bytes)	0 - 65535 (0x0000 to 0xFFFF)
Number of Registers	(2 Bytes)	1 to 125 (0x7D)
Checksum	(2 bytes)	Calculated automatically

Here is an example sent command as printed in the terminal (all Hex):

```
INFO CLASSICLADDER-   Modbus I/O module to send: Lgt=8 <-  Slave address-1  Function code-3 ↩
     Data-21 0 0 1 8E 36
```

Meaning (Hex):

- Lgt = 8 = message is 8 bytes long including slave number and checksum number
- Slave number = 1 (0x1) = Slave address 1
- Function code = 3 (0x3) = read holding register
- Start at address = highbyte 33 (0x21) lowbyte 0 (0x00) = combined address = 8448 (0x2100)
- Number of Registers = 1 (0x1) = return 1 2-byte register (holding and reading registers are always 2 bytes wide)
- Checksum = high byte 0x8E lowbyte 0x36 = (0x8E36)

8.2.11.2. Error response

If there is an error response, it sends the function code plus 0x80, an error code, and a checksum. Getting an error response means the slave is seeing the request command but can not give valid data. Looking in the Modbus protocol reference:

Cuadro 8.3: Error returned for function code 3 (read holding register)

Name	**Number of bytes**	**Value (hex)**
Error code	1 Byte	131 (0x83)
Exception code	1 Byte	1-4 (0x01 to 0x04)
Checksum	(2 bytes)	Calculated automatically

Significado del código de excepción:

- 1 - illegal Function
- 2 - illegal data address
- 3 - illegal data value
- 4 - slave device failure

Here is an example received command as printed in the terminal (all Hex):

```
INFO CLASSICLADDER-   Modbus I/O module received: Lgt=5 ->   (Slave address-1  Function  ↩
    code-83 ) 2 C0 F1
```

Meaning (Hex):

- Slave number = 1 (0x1) = Slave address 1
- Function code = 131 (0x83) = error while reading holding register
- Error code = 2 (0x2) = illegal data address requested
- Checksum = (0x8E36)

8.2.11.3. Data response

Looking in the Modbus protocol reference for Response:

Cuadro 8.4: Data response for function code 3 (read holding register)

Name	**número de bytes**	**Value (hex)**
Function code	1 Byte	3 (0x03)
Byte count	1 Byte	2 x N*
Valor del registro	N* x 2 Bytes	returned value of requested address
Checksum	(2 bytes)	calculated automatically

*N = Number of registers

Here is an example received command as printed in the terminal (all Hex):

```
INFO CLASSICLADDER-   Modbus I/O module received: Lgt=7 ->   (Slave address-1  Function  ↩
    code-3 2 0 0 B8 44)
```

meaning (Hex):

- Slave number = 1 (0x1) = Slave address 1
- Requested function code = 3 (0x3) = read holding register requested
- count of byte registers = 2 (0x1) = return 2 bytes (each register value is 2 bytes wide)
- value of highbyte = 0 (0x0) = high byte value of address 8448 (0x2100)
- value of lowbyte = 0 (0x0) = high byte value of address 8448 (0x2100)
- Checksum = (0xB844)

(high and low bytes are combined to create a 16 bit value and then transferred to ClassicLadder's variable.) Read Registers can be mapped to %W or %QW (internal memory or HAL out pins). Write registers can be mapped from %W, %QW or %IW (internal memory, HAL out pins or HAL in pins). The variable number will start at the number entered in the modbus I/O registry setup page's column: *First variable mapped*. If multiple registers are requested in one read/write then the variable number are sequential after the first one.

8.2.11.4. MODBUS Bugs

- In compare blocks the function %W=ABS(%W1- %W2) is accepted but does not compute properly. only %W0=ABS(%W1) is currently legal.
- When loading a ladder program it will load Modbus info but will not tell ClassicLadder to initialize Modbus. You must initialize Modbus when you first load the GUI by adding *--modmaster*.
- If the section manager is placed on top of the section display, across the scroll bar and exit is clicked the non-realtime program crashes.
- When using *--modmaster* you must load the ladder program at the same time or else only TCP will work.
- reading/writing multiple registers in Modbus has checksum errors.

8.2.12. Setting up ClassicLadder

In this section we will cover the steps needed to add ClassicLadder to a StepConf Wizard generated config. On the advanced Configuration Options page of StepConf Wizard check off "Include ClassicLadder PLC".

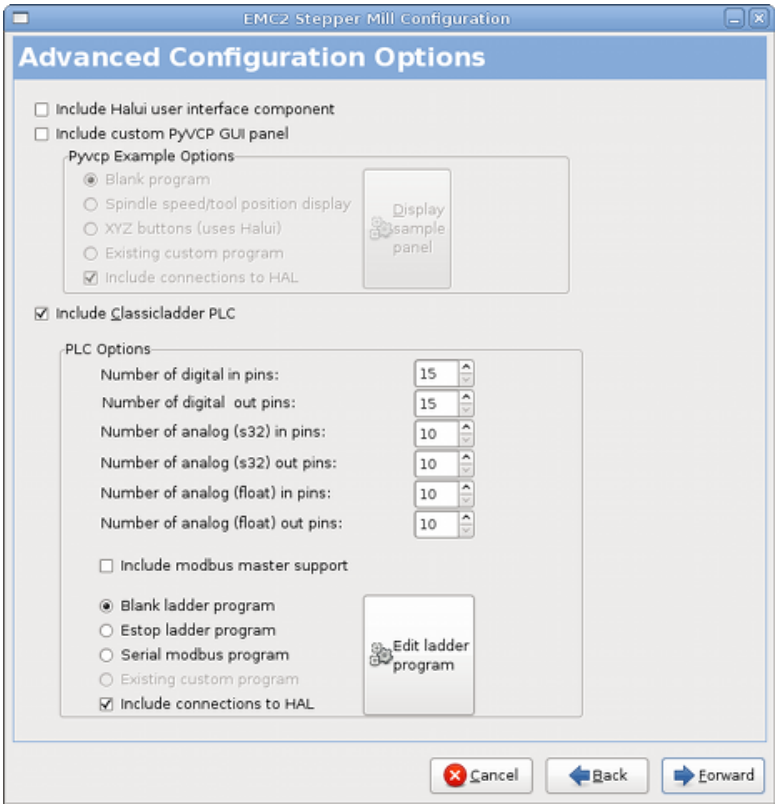


Figura 8.16: StepConf ClassicLadder

8.2.12.1. Add the Modules

If you used the StepConf Wizard to add ClassicLadder you can skip this step.

To manually add ClassicLadder you must first add the modules. This is done by adding a couple of lines to the custom.hal file.

Esta línea carga el módulo de tiempo real:

```
loadrt classicladder_rt
```

This line adds the ClassicLadder function to the servo thread:

```
addf classicladder.0.refresh servo-thread
```

8.2.12.2. Adding Ladder Logic

Now start up your config and select ”File/Ladder Editor” to open up the ClassicLadder GUI. You should see a blank Section Display and Sections Manager window as shown above. In the Section Display window open the Editor. In the Editor window select Modify. Now a Properties window pops up and the Section Display shows a grid. The grid is one rung of ladder. The rung can contain branches. A simple rung has one input, a connector line and one output. A rung can have up to six horizontal branches. While it is possible to have more than one circuit in a run the results are not predictable.

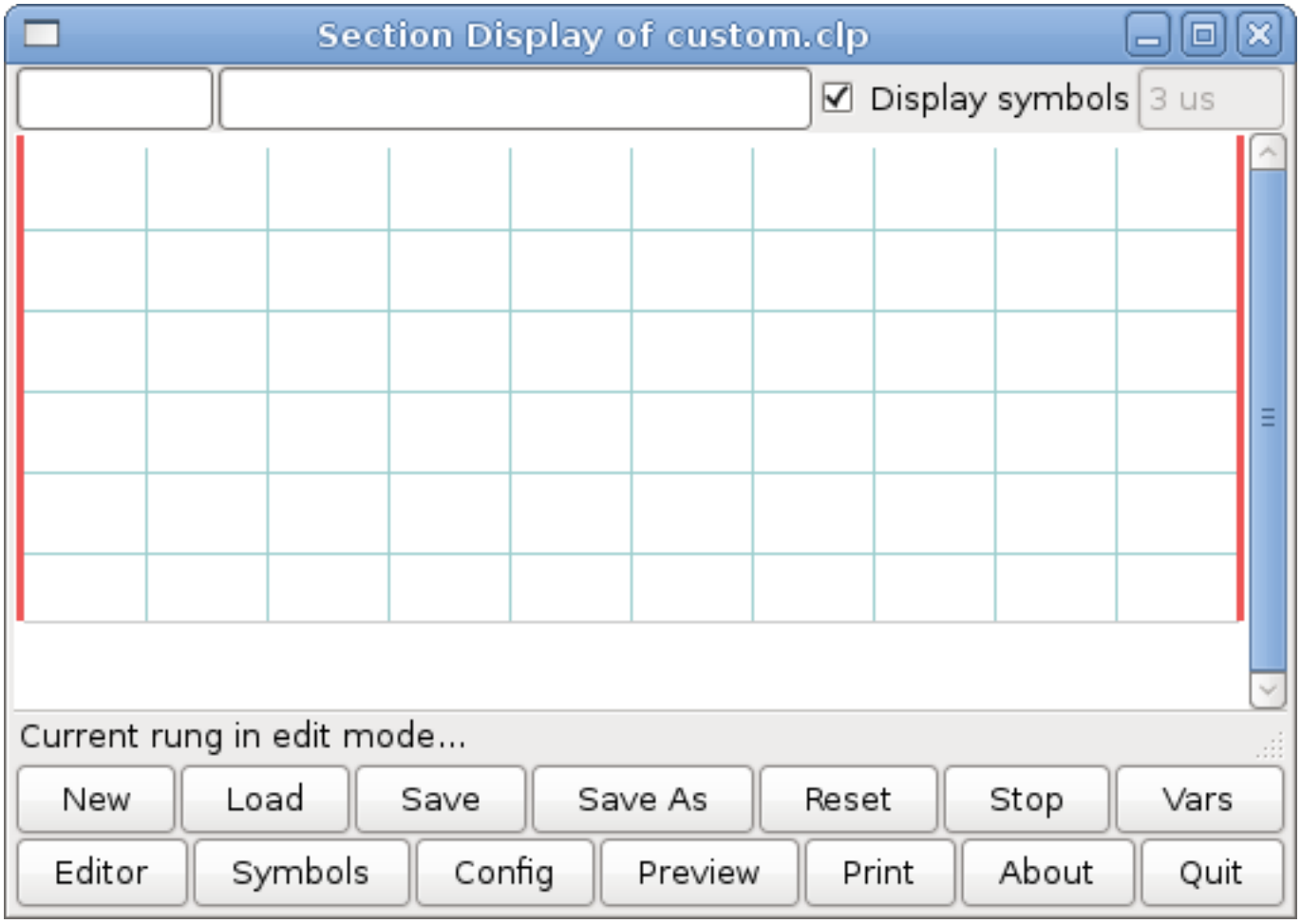


Figura 8.17: Section Display with Grid

Now click on the N.O. input in the Editor Window.

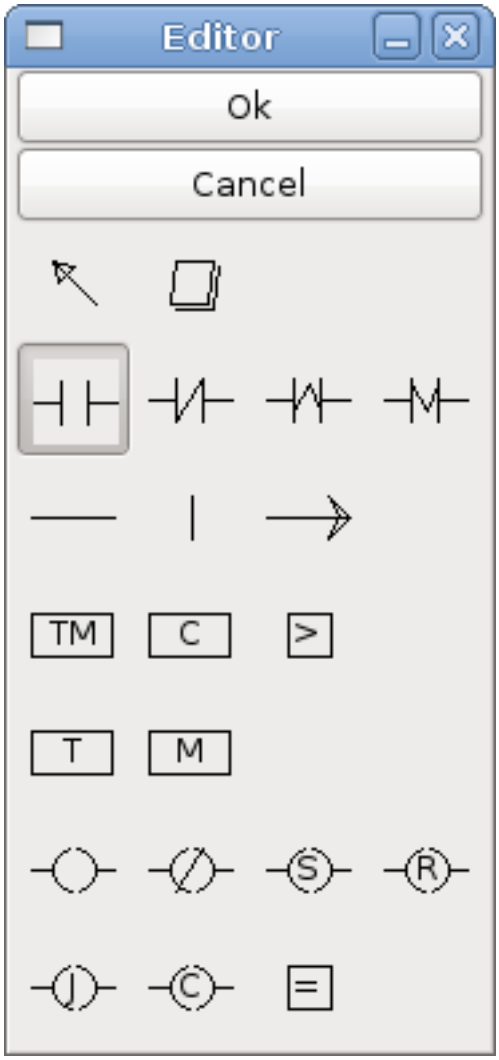


Figura 8.18: Editor Window

Now click in the upper left grid to place the N.O. Input into the ladder.

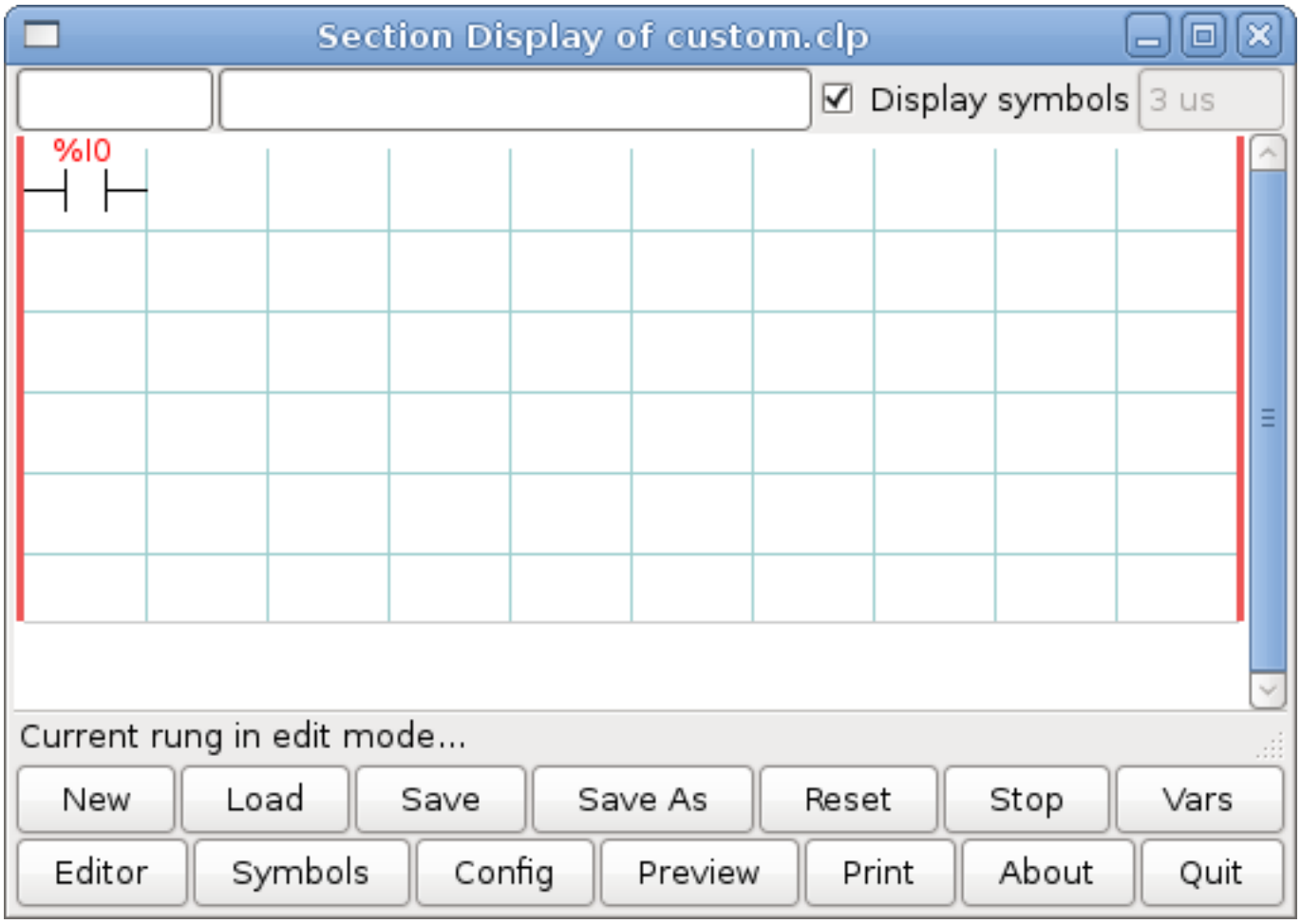


Figura 8.19: Section Display with Input

Repeat the above steps to add a N.O. output to the upper right grid and use the Horizontal Connection to connect the two. It should look like the following. If not, use the Eraser to remove unwanted sections.

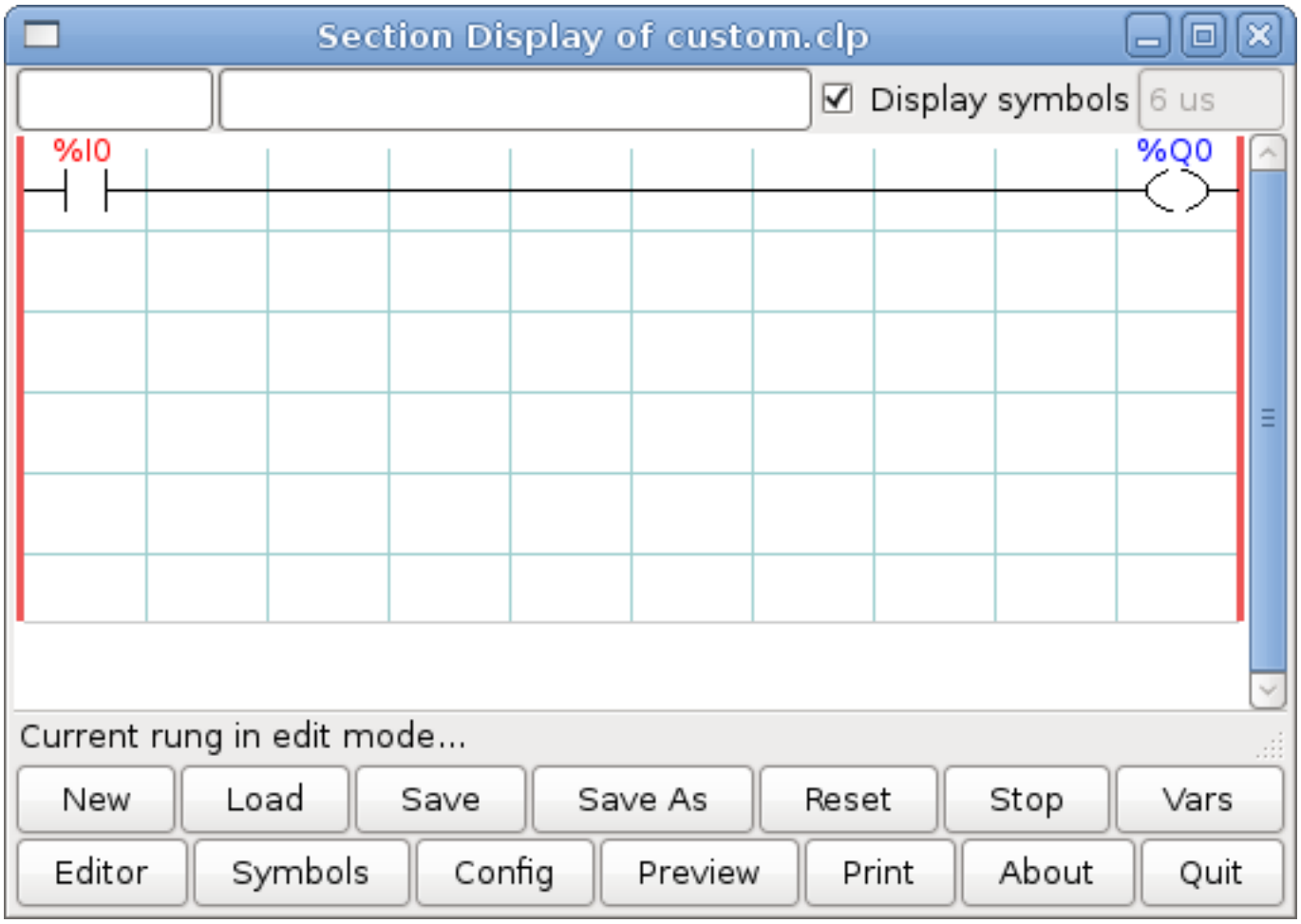


Figura 8.20: Section Display with Rung

Now click on the OK button in the Editor window. Now your Section Display should look like this:

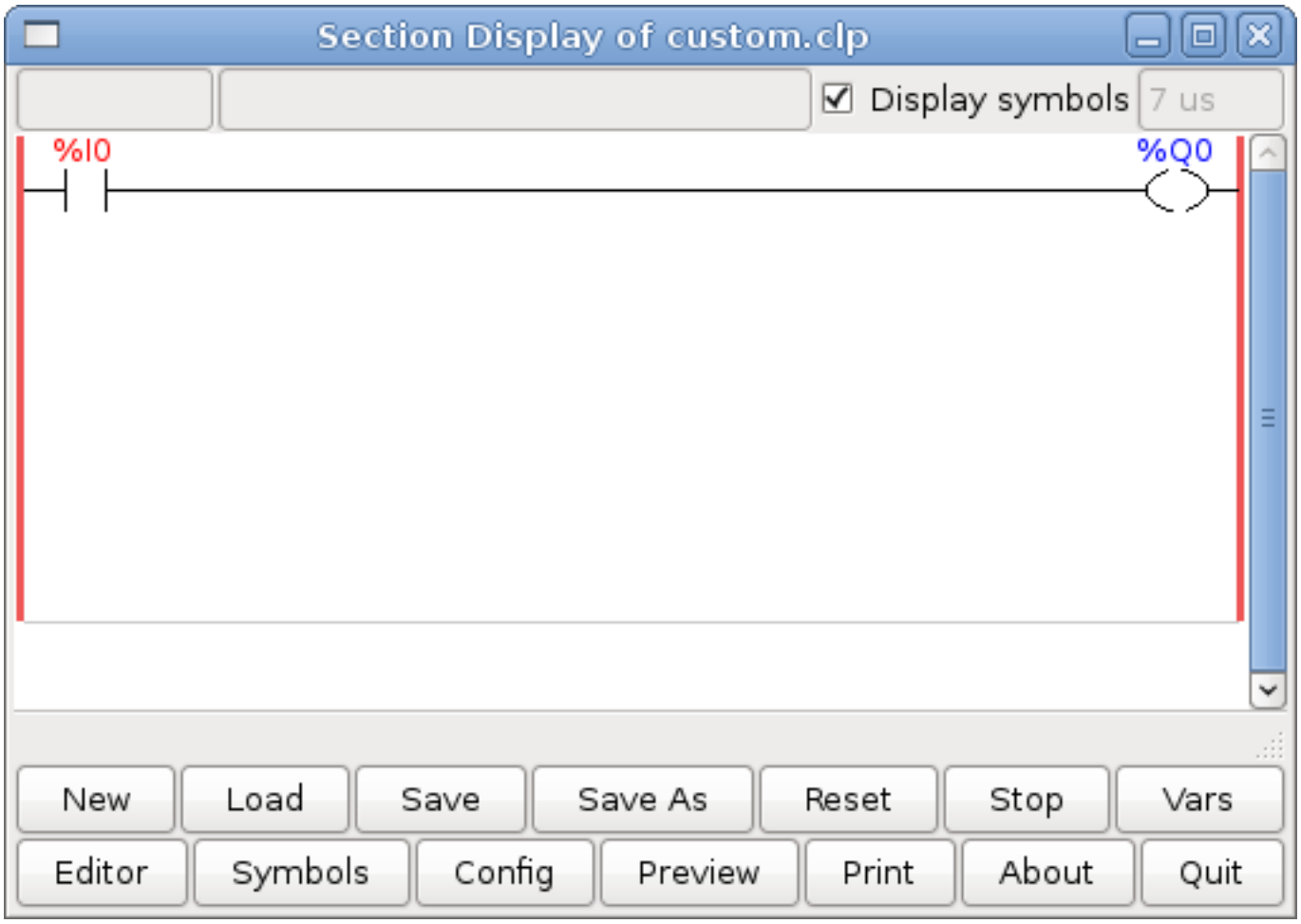


Figura 8.21: Section Display Finished

To save the new file select *Save As* and give it a name. The .clp extension will be added automatically. It should default to the running config directory as the place to save it.

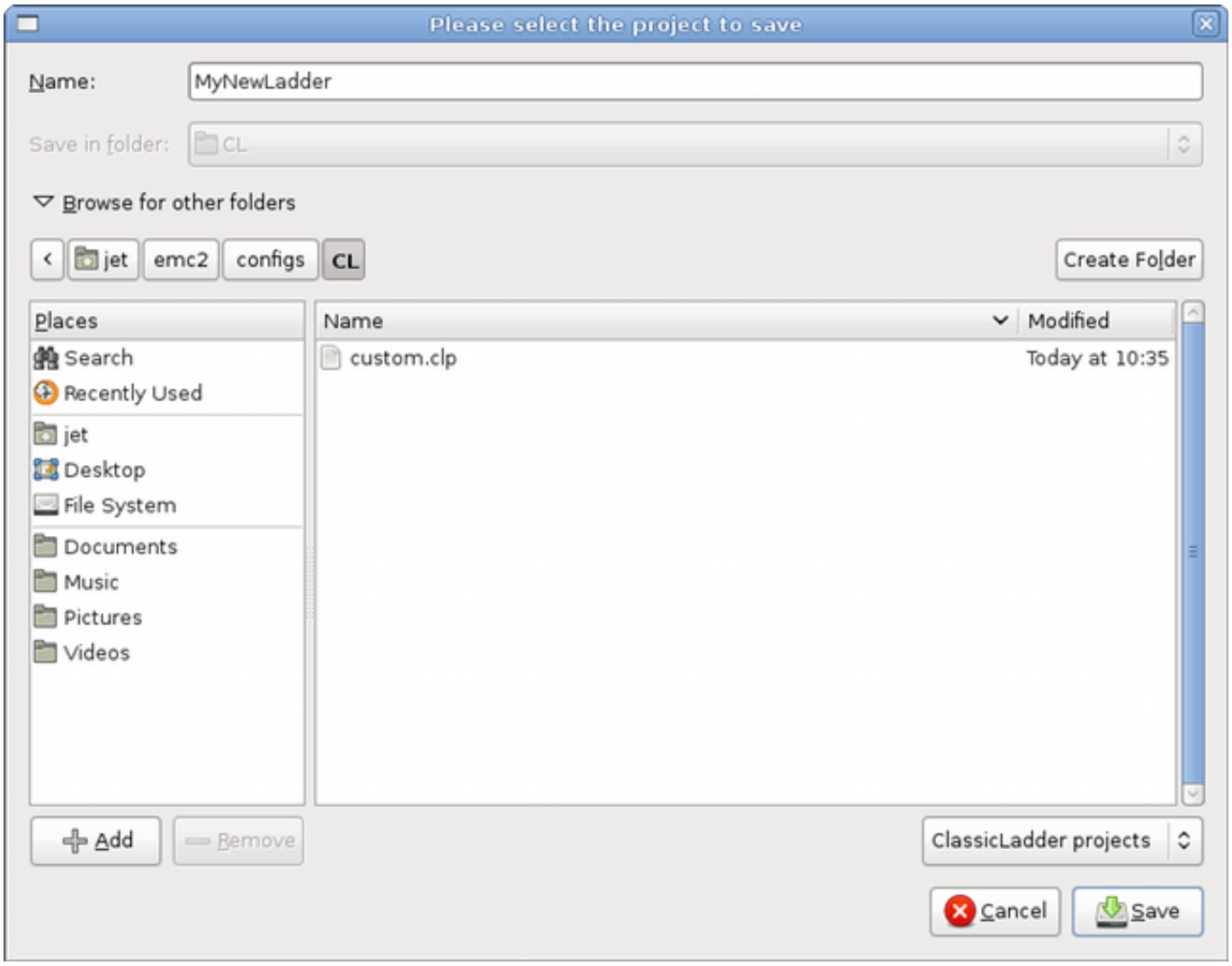


Figura 8.22: Save As Dialog

Again if you used the StepConf Wizard to add ClassicLadder you can skip this step.

To manually add a ladder you need to add add a line to your custom.hal file that will load your ladder file. Close your LinuxCNC session and add this line to your custom.hal file.

```
loadusr -w classicladder --nogui MyLadder.clp
```

Now if you start up your LinuxCNC config your ladder program will be running as well. If you select ”File/Ladder Editor”, the program you created will show up in the Section Display window.

8.3. ClassicLadder Examples

8.3.1. Wrapping Counter

To have a counter that *wraps around* you have to use the preset pin and the reset pin. When you create the counter set the preset at the number you wish to reach before wrapping around to 0. The logic is if the counter value is over the preset then reset the counter and if the underflow is on then

set the counter value to the preset value. As you can see in the example when the counter value is greater than the counter preset the counter reset is triggered and the value is now 0. The underflow output %Q2 will set the counter value at the preset when counting backwards.

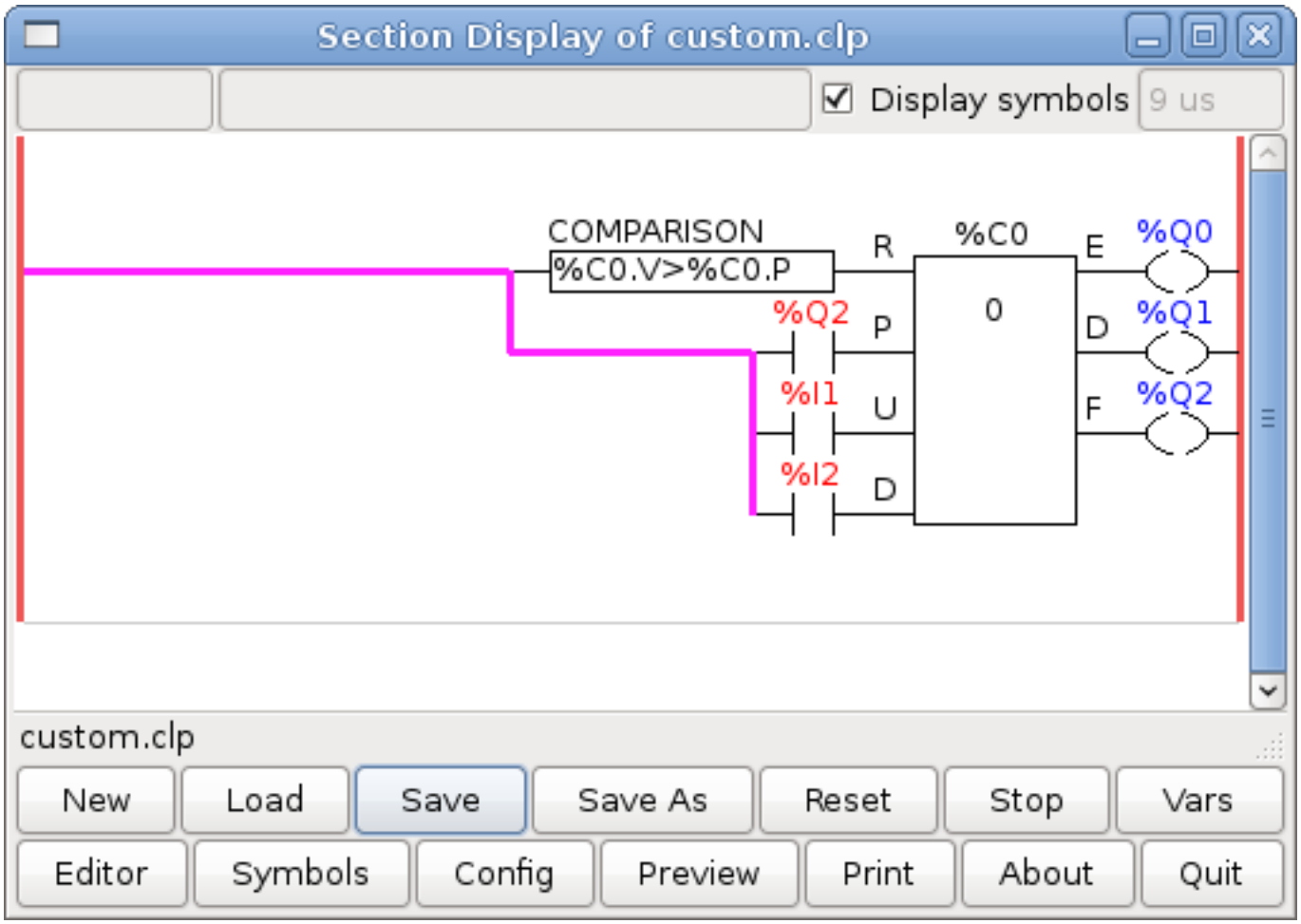


Figura 8.23: Wrapping Counter

8.3.2. Reject Extra Pulses

This example shows you how to reject extra pulses from an input. Suppose the input pulse %I0 has an annoying habit of giving an extra pulse that spoils our logic. The TOF (Timer Off Delay) prevents the extra pulse from reaching our cleaned up output %Q0. How this works is when the timer gets an input the output of the timer is on for the duration of the time setting. Using a normally closed contact %TM0.Q the output of the timer blocks any further inputs from reaching our output until it times out.

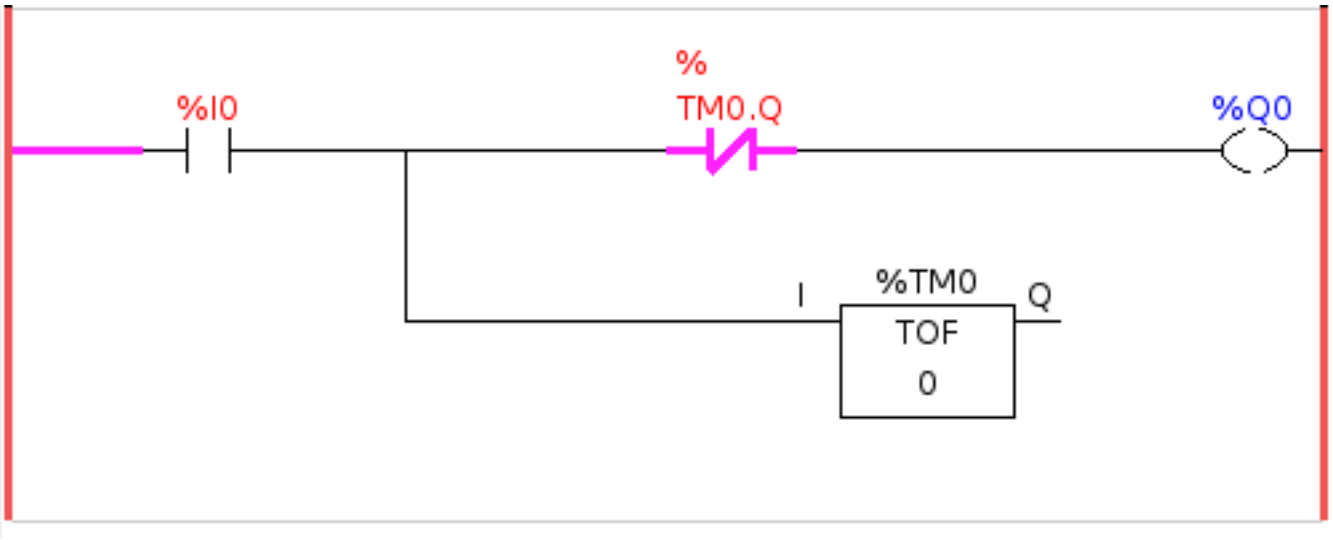


Figura 8.24: Reject Extra Pulse

8.3.3. External E-Stop

The External E-Stop example is in the /config/classicladder/cl-estop folder. It uses a PyVCP panel to simulate the external components.

To interface an external E-Stop to LinuxCNC and have the external E-Stop work together with the internal E-Stop requires a couple of connections through ClassicLadder.

First we have to open the E-Stop loop in the main HAL file by commenting out by adding the pound sign as shown or removing the following lines.

```
# net estop-out <= iocontrol.0.user-enable-out
# net estop-out => iocontrol.0.emc-enable-in
```

Next we add ClassicLadder to our custom.hal file by adding these two lines:

```
loadrt classicladder_rt
addf classicladder.0.refresh servo-thread
```

Next we run our config and build the ladder as shown here.

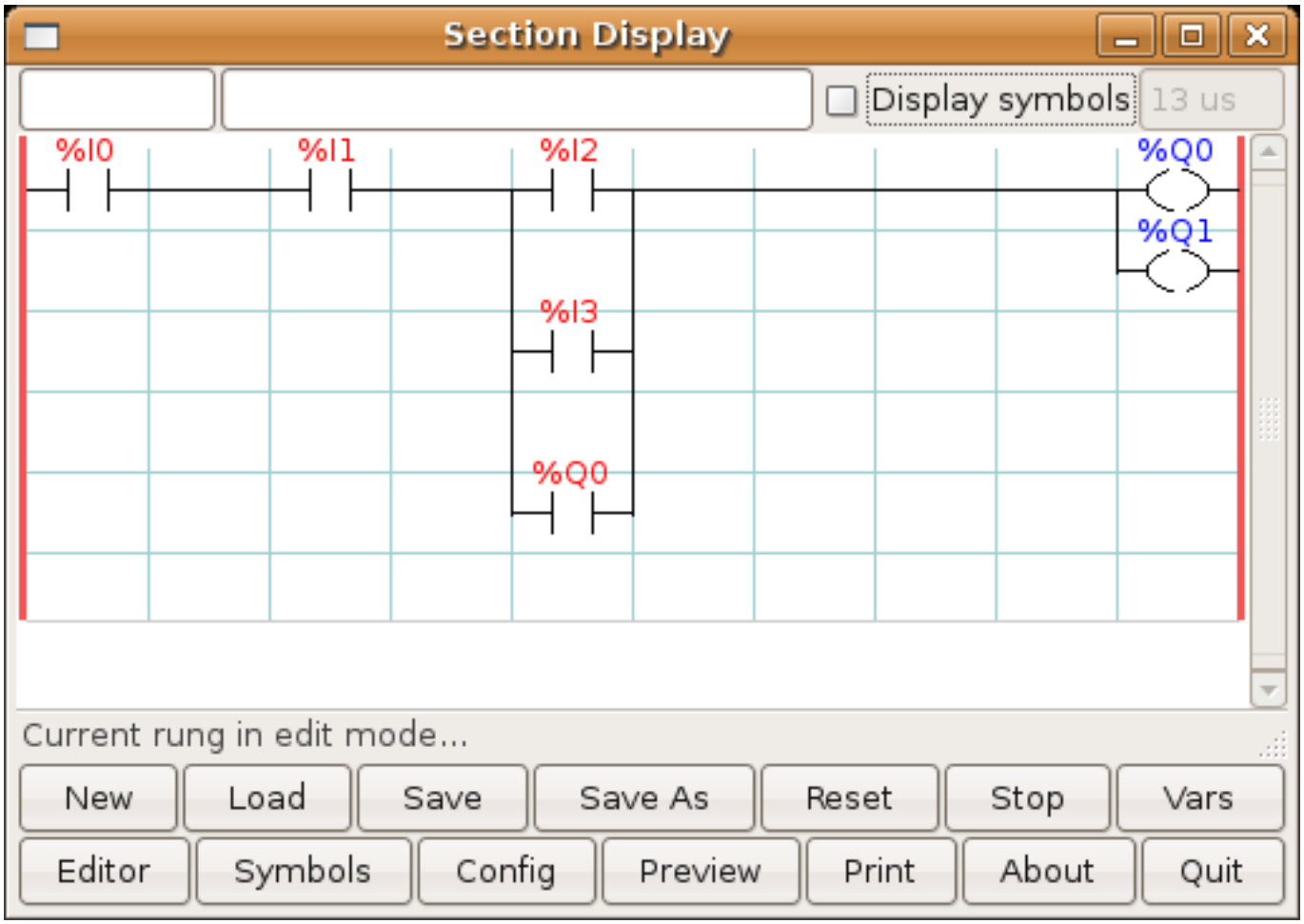


Figura 8.25: E-Stop Section Display

After building the ladder select Save As and save the ladder as estop.clp

Ahora agregue la siguiente línea a su archivo custom.hal.

```
# Load the ladder
loadusr classicladder --nogui estop.clp
```

I/O assignments

- %I0 = Input from the PyVCP panel simulated E-Stop (the checkbox)
- %I1 = Input from LinuxCNC's E-Stop
- %I2 = Input from LinuxCNC's E-Stop Reset Pulse
- %I3 = Input from the PyVCP panel reset button
- %Q0 = Output to LinuxCNC to enable
- %Q1 = Output to external driver board enable pin (use a N/C output if your board had a disable pin)

Next we add the following lines to the custom_postgui.hal file

```
# E-Stop example using PyVCP buttons to simulate external components

# The PyVCP checkbutton simulates a normally closed external E-Stop
net ext-estop classicladder.0.in-00 <= pyvcp.py-estop

# Request E-Stop Enable from LinuxCNC
net estop-all-ok iocontrol.0.emc-enable-in <= classicladder.0.out-00

# Request E-Stop Enable from PyVCP or external source
net ext-estop-reset classicladder.0.in-03 <= pyvcp.py-reset

# This line resets the E-Stop from LinuxCNC
net emc-reset-estop iocontrol.0.user-request-enable => classicladder.0.in-02

# This line enables LinuxCNC to unlatch the E-Stop in ClassicLadder
net emc-estop iocontrol.0.user-enable-out => classicladder.0.in-01

# This line turns on the green indicator when out of E-Stop
net estop-all-ok => pyvcp.py-es-status
```

Next we add the following lines to the panel.xml file. Note you have to open it with the text editor not the default html viewer.

```
<pyvcp>
<vbox>
<label><text>”E-Stop Demo”</text></label>
<led>
<halpin>”py-es-status”</halpin>
<size>50</size>
<on_color>”green”</on_color>
<off_color>”red”</off_color>
</led>
<checkbutton>
<halpin>”py-estop”</halpin>
<text>”E-Stop”</text>
</checkbutton>
</vbox>
<button>
<halpin>”py-reset”</halpin>
<text>”Reset”</text>
</button>
</pyvcp>
```

Now start up your config and it should look like this.

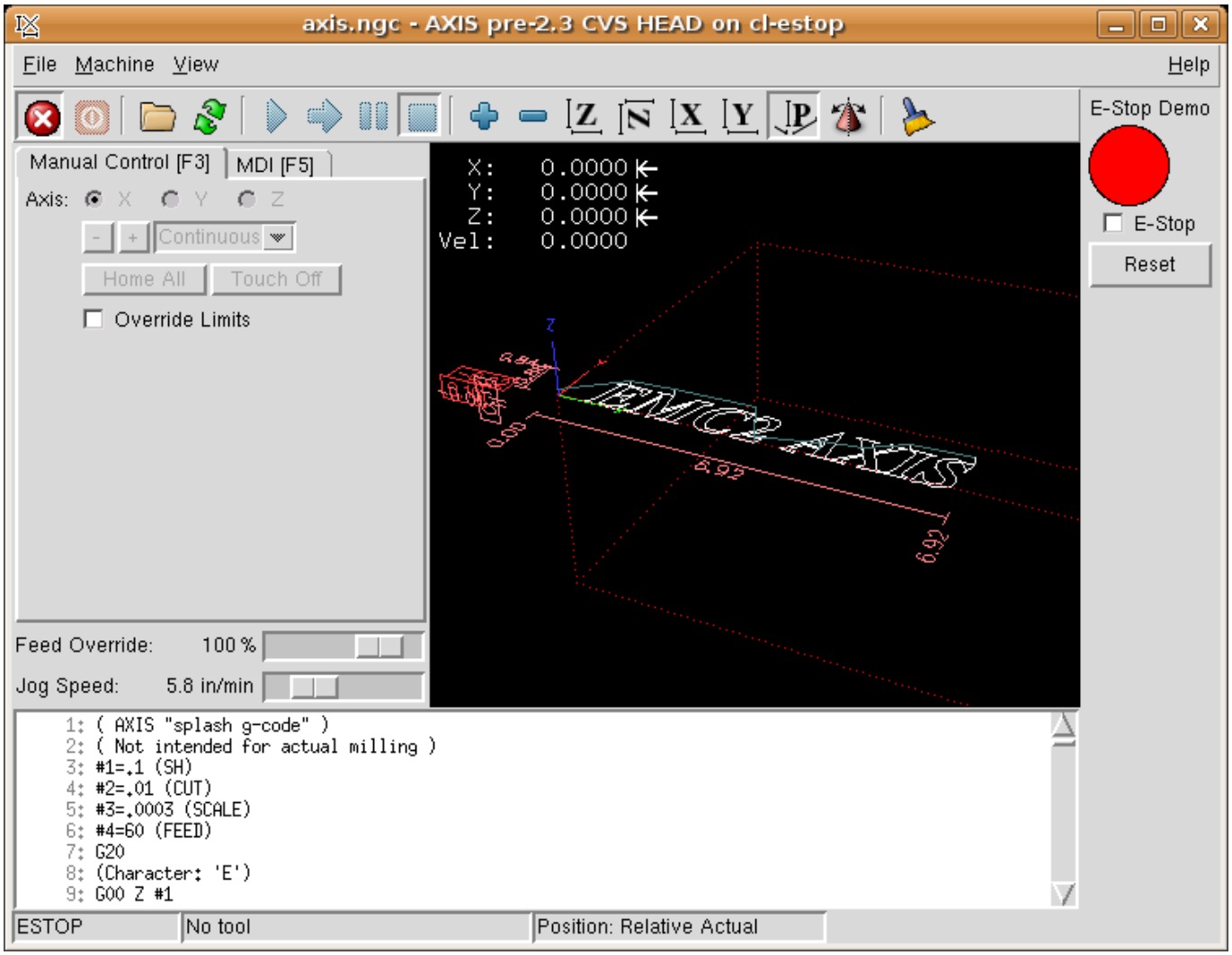


Figura 8.26: AXIS E-Stop

Note that in this example like in real life you must clear the remote E-Stop (simulated by the checkbox) before the AXIS E-Stop or the external Reset will put you in OFF mode. If the E-Stop in the AXIS screen was pressed, you must press it again to clear it. You cannot reset from the external after you do an E-Stop in AXIS.

8.3.4. Timer/Operate Example

In this example we are using the Operate block to assign a value to the timer preset based on if an input is on or off.

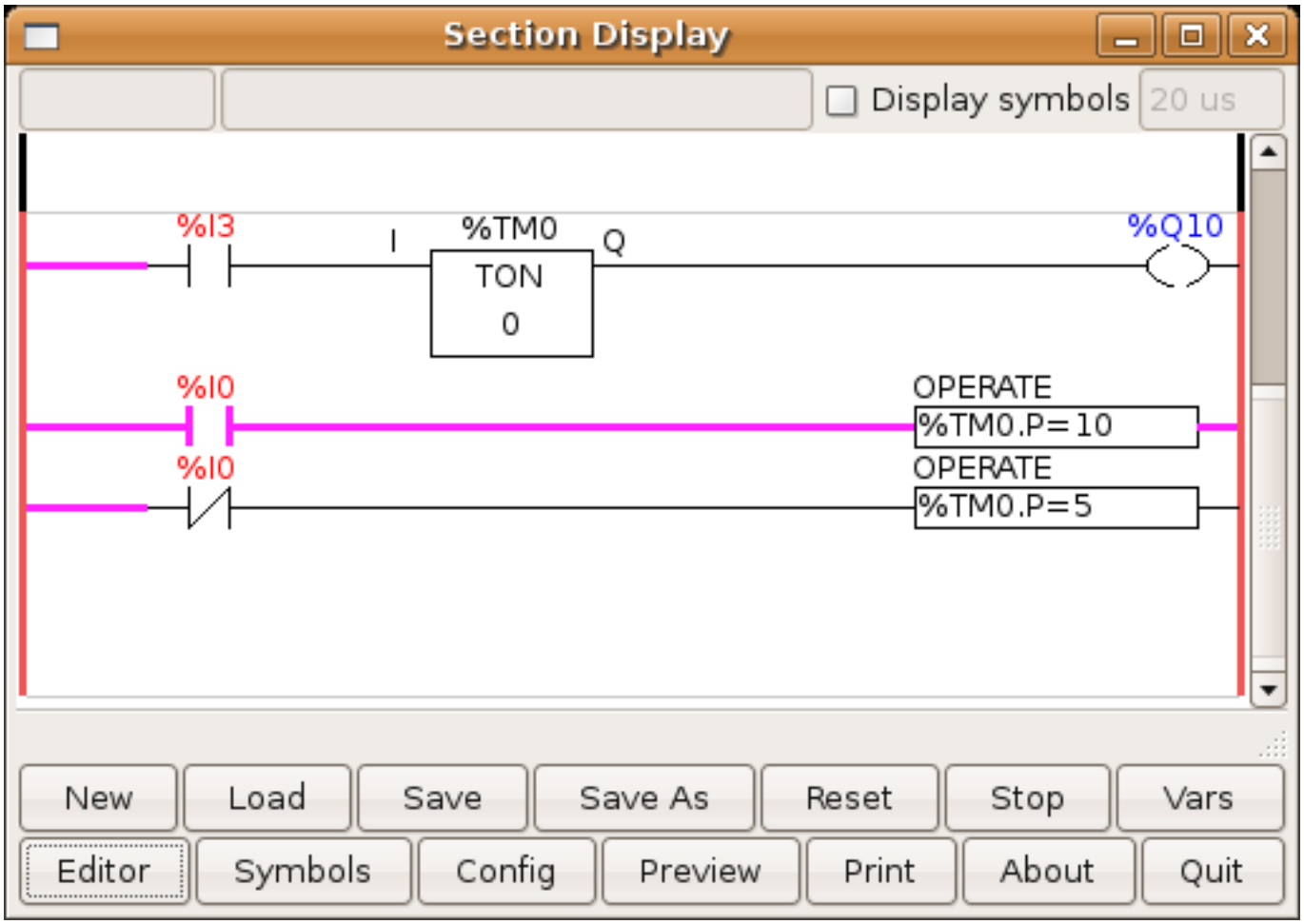


Figura 8.27: Timer/Operate Example

In this case %I0 is true so the timer preset value is 10. If %I0 was false the timer preset would be 5.

Capítulo 9

Temas avanzados

9.1. Cinemática

9.1.1. Introducción

When we talk about CNC machines, we usually think about machines that are commanded to move to certain locations and perform various tasks. In order to have an unified view of the machine space, and to make it fit the human point of view over 3D space, most of the machines (if not all) use a common coordinate system called the Cartesian Coordinate System.

The Cartesian Coordinate system is composed of three axes (X, Y, Z) each perpendicular to the other two [1].

Cuando hablamos de un programa de código G (RS274/NGC) hablamos de un conjunto de comandos (G0, G1, etc.) que tienen posiciones cartesianas como parámetros (X- Y- Z-). Estas posiciones se refieren exactamente a las posiciones cartesianas. Parte del controlador de movimiento LinuxCNC es responsable de traducir esas posiciones en posiciones que corresponden a la cinemática de la máquina [2].

9.1.1.1. Joints vs Axes

A joint of a CNC machine is a one of the physical degrees of freedom of the machine. This might be linear (leadscrews) or rotary (rotary tables, robot arm joints). There can be any number of joints on a given machine. For example, one popular robot has 6 joints, and a typical simple milling machine has only 3.

There are certain machines where the joints are laid out to match kinematics axes (joint 0 along axis X, joint 1 along axis Y, joint 2 along axis Z), and these machines are called Cartesian machines (or machines with Trivial Kinematics). These are the most common machines used in milling, but are not very common in other domains of machine control (e.g. welding: puma-typed robots).

LinuxCNC supports axes with names: X Y Z A B C U V W. The X Y Z axes typically refer to the usual Cartesian coordinates. The A B C axes refer to rotational coordinates about the X Y Z axes respectively. The U V W axes refer to additional coordinates that are commonly made colinear to the X Y Z axes respectively.

[1] The word ”axes” is also commonly (and wrongly) used when talking about CNC machines, and referring to the moving directions of the machine.

[2] Cinemática: una función bidireccional para transformar del espacio cartesiano al espacio de articulaciones.

9.1.2. Trivial Kinematics

Las máquinas más simples son aquellas en las que cada articulación se coloca a lo largo de uno de los ejes cartesianos. En estas máquinas, el mapeo de espacio cartesiano (el programa de código G) al espacio de articulaciones (los actuadores de la máquina) es trivial; es un mapeo simple 1:1:

```
pos->tran.x = joints[0];
pos->tran.y = joints[1];
pos->tran.z = joints[2];
```

In the above code snippet one can see how the mapping is done: the X position is identical with the joint 0, the Y position with joint 1, etc. The above refers to the direct kinematics (one direction of the transformation). The next code snippet refers to the inverse kinematics (or the inverse direction of the transformation):

```
joints[0] = pos->tran.x;
joints[1] = pos->tran.y;
joints[2] = pos->tran.z;
```

In LinuxCNC, the identity kinematics are implemented with the *trivkins* kinematics module and extended to 9 axes. The default relationships between axis coordinates and joint numbers are: [3] [4]

```
pos->tran.x = joints[0];
pos->tran.y = joints[1];
pos->tran.z = joints[2];
pos->a      = joints[3];
pos->b      = joints[4];
pos->c      = joints[5];
pos->u      = joints[6];
pos->v      = joints[7];
pos->w      = joints[8];
```

Similarly, the default relationships for inverse kinematics for trivkins are:

```
joints[0] = pos->tran.x;
joints[1] = pos->tran.y;
joints[2] = pos->tran.z;
joints[3] = pos->a;
joints[4] = pos->b;
joints[5] = pos->c;
joints[6] = pos->u;
joints[7] = pos->v;
joints[8] = pos->w;
```

It is straightforward to do the transformation for a trivial ”kins” (*trivkins* kinematics) or Cartesian machine provided that there are no omissions in the axis letters used.

It gets a bit more complicated if the machine is missing one or more of the axis letters. The problems of omitted axis letters is addressed by using the *coordinates*= module parameter with the trivkins module. Joint numbers are assigned consecutively to each coordinate specified. A lathe can be described with *coordinates=xz* The joint assignments will then be:

```
joints[0] = pos->tran.x
joints[1] = pos->tran.z
```

[3] If the machine (for example a lathe) is mounted with only the X, Z and A axes and the INI file of LinuxCNC contains only the definition of these 3 joints, then the previous assertion is false. Because we currently have (joint0=X, joint1=Z, joint2=A) which assumes that joint1=Y. To make this work in LinuxCNC just define all the axes (XYZA), LinuxCNC will then use a simple loop in HAL for unused Y axis.

[4] Another way to make it work is to change the corresponding code and recompile the software.

Use of the *coordinates*= parameter is recommended for configurations that omit axis letters. [5]

The *trivkins* kinematics module also allows the same coordinate to be specified for more than one joint. This feature can be useful on machines like a gantry having two independent motors for the y coordinate. Such a machine could use *coordinates*=*xyyz* resulting in joint assignments:

```
joints[0] = pos->tran.x
joints[1] = pos->tran.y
joints[2] = pos->tran.y
joints[3] = pos->tran.z
```

See the trivkins man pages for more information.

9.1.3. Non-trivial kinematics

There can be quite a few types of machine setups (robots: puma, scara; hexapods etc.). Each of them is set up using linear and rotary joints. These joints don't usually match with the Cartesian coordinates, therefore we need a kinematics function which does the conversion (actually 2 functions: forward and inverse kinematics function).

To illustrate the above, we will analyze a simple kinematics called bipod (a simplified version of the tripod, which is a simplified version of the hexapod).

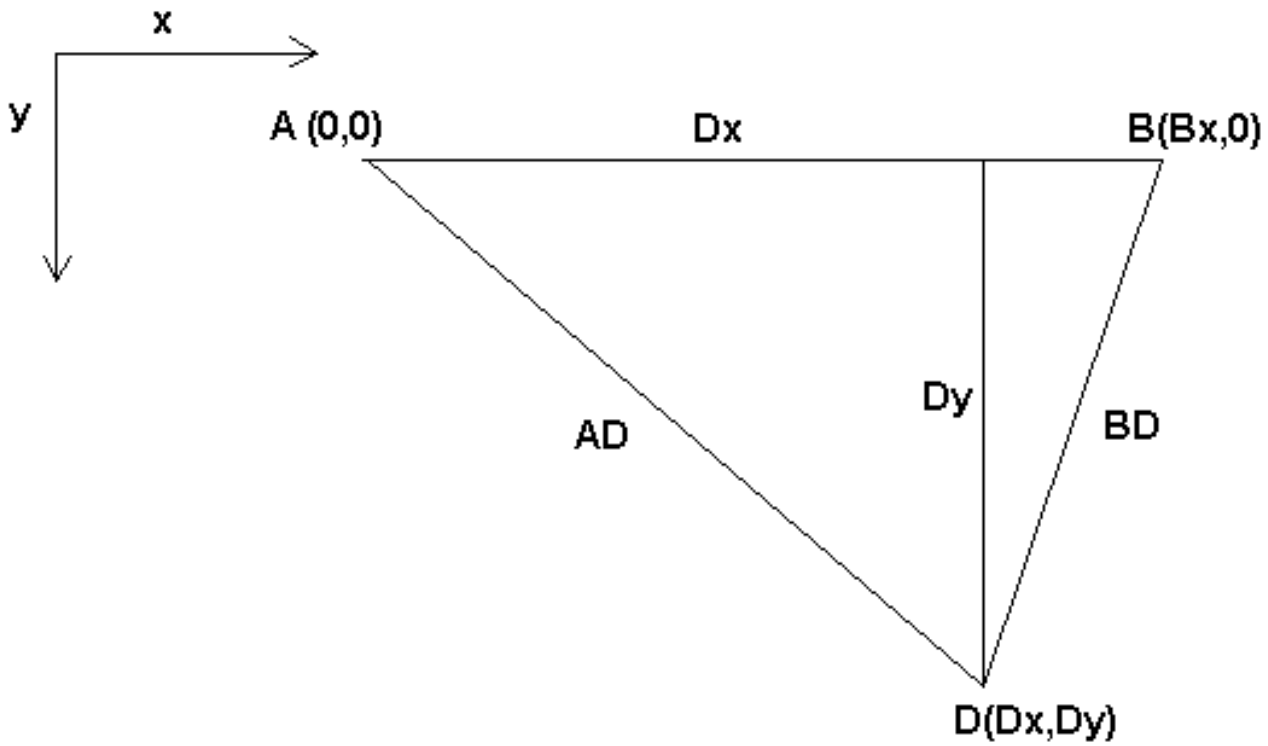


Figura 9.1: Bipod setup

[5] Historically, the trivkins module did not support the *coordinates*= parameter so lathe configs were often configured as XYZ machines. The unused Y axis was configured to 1) home immediately, 2) use a simple loopback to connect its position command HAL pin to its position feedback HAL pin, and 3) hidden in gui displays. Numerous sim configs use these methods in order to share common HAL files.

The Bipod we are talking about is a device that consists of 2 motors placed on a wall, from which a device is hung using some wire. The joints in this case are the distances from the motors to the device (named AD and BD in the figure).

The position of the motors is fixed by convention. Motor A is in (0,0), which means that its X coordinate is 0, and its Y coordinate is also 0. Motor B is placed in (Bx, 0), which means that its X coordinate is Bx.

Our tooltip will be in point D which gets defined by the distances AD and BD, and by the Cartesian coordinates Dx, Dy.

The job of the kinematics is to transform from joint lengths (AD, BD) to Cartesian coordinates (Dx, Dy) and vice-versa.

9.1.3.1. Forward transformation

To transform from joint space into Cartesian space we will use some trigonometry rules (the right triangles determined by the points (0,0), (Dx,0), (Dx,Dy) and the triangle (Dx,0), (Bx,0) and (Dx,Dy)).

We can easily see that:

$$AD^2 = x^2 + y^2 \quad BD^2 = (Bx - x)^2 + y^2$$

likewise:

$$BD^2 = (Bx - x)^2 + y^2$$

If we subtract one from the other we will get:

$$AD^2 - BD^2 = x^2 + y^2 - x^2 + 2 * x * Bx - Bx^2 - y^2$$

and therefore:

$$x = \frac{AD^2 - BD^2 + Bx^2}{2 * Bx}$$

From there we calculate:

$$y = \sqrt{AD2 - x2}$$

Note that the calculation for y involves the square root of a difference, which may not result in a real number. If there is no single Cartesian coordinate for this joint position, then the position is said to be a singularity. In this case, the forward kinematics return -1.

Translated to actual code:

```
double AD2 = joints[0] * joints[0];
double BD2 = joints[1] * joints[1];
double x = (AD2 - BD2 + Bx * Bx) / (2 * Bx);
double y2 = AD2 - x * x;
if(y2 < 0) return -1;
pos->tran.x = x;
pos->tran.y = sqrt(y2);
return 0;
```

9.1.3.2. Inverse transformation

La cinemática inversa es mucho más fácil en nuestro ejemplo, ya que podemos escribir directamente:

$$AD = \sqrt{x^2 + y^2}$$

$$BD = \sqrt{(Bx - x)^2 + y^2}$$

or translated to actual code:

```
double x2 = pos->tran.x * pos->tran.x;
double y2 = pos->tran.y * pos->tran.y;
joints[0] = sqrt(x2 + y2);
joints[1] = sqrt((Bx - pos->tran.x)*(Bx - pos->tran.x) + y2);
return 0;
```

9.1.4. Implementation details

A kinematics module is implemented as a HAL component, and is permitted to export pins and parameters. It consists of several ”C” functions (as opposed to HAL functions):

```
int kinematicsForward(const double *joint, EmcPose *world,
const KINEMATICS_FORWARD_FLAGS *fflags,
KINEMATICS_INVERSE_FLAGS *iflags)
```

Implements the forward kinematics function.

```
int kinematicsInverse(const EmcPose * world, double *joints,
const KINEMATICS_INVERSE_FLAGS *iflags,
KINEMATICS_FORWARD_FLAGS *fflags)
```

Implements the inverse kinematics function.

```
KINEMATICS_TYPE kinematicsType(void)
```

Returns the kinematics type identifier, típicamente *KINEMATICS_BOTH*:

1. KINEMATICS_IDENTITY (each joint number corresponds to an axis letter)
2. KINEMATICS_BOTH (se proporcionan funciones de cinemática normal e inversa)
3. KINEMATICS_FORWARD_ONLY
4. KINEMATICS_INVERSE_ONLY

---

nota
GUIs may interpret KINEMATICS_IDENTITY to hide the distinctions between joint numbers and axis letters when in joint mode (typically prior to homing).

```
int kinematicsSwitchable(void)
int kinematicsSwitch(int switchkins_type)
KINS_NOT_SWITCHABLE
```

The function kinematicsSwitchable() returns 1 if multiple kinematics types are supported. The function kinematicsSwitch() selects the kinematics type. See Switchable Kinematitcs.

nota
The majority of provided kinematics modules support a single kinematics type and use the directive "**KINS_NOT_SWITCHABLE**" to supply defaults for the required kinematicsSwitchable() and kinematicsSwitch() functions.

```
int kinematicsHome(EmcPose *world, double *joint,
KINEMATICS_FORWARD_FLAGS *fflags,
KINEMATICS_INVERSE_FLAGS *iflags)
```

The home kinematics function sets all its arguments to their proper values at the known home position. When called, these should be set, when known, to initial values, e.g., from an INI file. If the home kinematics can accept arbitrary starting points, these initial values should be used.

```
int rtapi_app_main(void)
void rtapi_app_exit(void)
```

These are the standard setup and tear-down functions of RTAPI modules.

When they are contained in a single source file, kinematics modules may be compiled and installed by *halcompile*. See the *halcompile(1)* manpage or the HAL manual for more information.

9.1.4.1. Kinematics module using the userkins.comp template

Another way to create a custom kinematics module is to adapt the HAL component *userkins*. This template component can be modified locally by a user and can be built using halcompile.

See the userkins man pages for more information.

Note that to create switchable kinematic modules the required modifications are somewhat more complicated.

See *millturn.comp* as an example of a switchable kinematic module that was created using the *userkins.comp* template.

9.2. Setting up "modified" Denavit-Hartenberg (DH) parameters for *genserkins*

9.2.1. Prelude

LinuxCNC supports a number of kinematics modules including one that supports a generalized set of serial kinematics commonly specified via Denavit-Hartenberg parameters.

This document illustrates a method to set up the DH-parameters for a Mitsubishi RV-6SDL in LinuxCNC using *genserkins* kinematics.

nota
This document does not cover the creation of a *vismach* model which, while certainly very useful, requires just as much careful modeling if it is to match the *genserkins* model derived in this document.

nota
There may be errors and/or shortcomings — use at your own risk!

9.2.2. En general

With the proliferation of industrial robots comes an increased interest to control used robots with LinuxCNC. A common type of robot used in industry and manufacturing is the ”serial manipulator” designed as a series of motorized joints connected by rigid links. Serial robots often have six joints as required for the six degrees of freedom needed to both position (XYZ) and orient (ABC or pitch, roll, yaw) an object in space. Often these robots have an arm structure that extends from a base to an end-effector.

Control of such a serial robot requires the calculation of the end-effector’s position and orientation in relation to a reference coordinate system when the joint angles are known (**forward kinematics**) and also the more complex reverse calculation of the required joint angles for a given end-effector position and orientation in relation to the reference coordinate system (**inverse kinematics**). The standard mathematical tools used for these calculations are matrices which are basically tables of parameters and formulas that make it easier to handle the rotations and translations involved in forward and inverse kinematics calculations.

Detailed familiarity with the math is not required for a serial robot since LinuxCNC provides a kinematics module that implements an algorithm called *genserkins* to calculate the forward and inverse kinematics for a generic serial robot. In order to control a specific serial robot, *genserkins* must be provided with data so that it can build a mathematical model of the robot’s mechanical structure and thus do the math.

The required data needs to be in a standardized form that has been introduced by Jacques Denavit and Richard Hartenberg back in the fifties and are called the DH-Parameters. Denavit and Hartenberg used four parameters to describe how one joint is linked to the next. These parameters describe basically two rotations (*alpha* and *theta*) and two translations (*a* and *d*).

9.2.3. Modified DH-Parameters

As is often the case, this ”standard” has been modified by other authors who have introduced ”modified DH-parameters” and one must be very careful because *genserkins* uses ”modified DH-parameters” as described in the publication ”Introduction to Robotics, Mechanics and Control” by John J. Craig. Beware there is a lot of information to be found on *DH-parameters* but rarely does the author define which convention is actually used. In addition, some people have found it necessary to change the parameter named *a* to *r* and have thus added to the confusion. This document adheres to the convention in the above mentioned publication by Craig with the difference that joint and parameter enumeration begins with the number *0* in order to be consistent with *genserkins* and its HAL pins.

Los parámetros DH estándar y modificados constan de cuatro valores numéricos para cada articulación (*a*, *d*, *alpha* y *theta*) que describen cómo el sistema de coordenadas (CS) asociado a una articulación tiene que ser movido y girado para alinearse con la siguiente articulación. Alineado significa que el eje Z de nuestro CS coincide con el eje de rotación de la articulación y apunta en la dirección positiva de modo que, usando la regla de la mano derecha con el pulgar apuntando a la dirección positiva del eje Z, los dedos apuntan a la dirección positiva de rotación de la articulación. Queda claro que para hacer esto, uno debe decidir sobre las direcciones positivas de todas las articulaciones antes de comenzar a derivar parámetros!

The difference between ”standard” and ”modified” notations is in how the parameters are allocated to the links. Using the ”standard” DH-Parameters in *genserkins* will **not** give the correct mathematical model.

9.2.4. Modified DH-Parameters as used in *genserkins*

Note that *genserkins* does not handle offsets to theta-values — theta is the joint variable that is **controlled** by LinuxCNC. With the CS aligned with the joint, a rotation around its Z-Axis is identical to

the rotation commanded to that joint by LinuxCNC. This makes it impossible to define the 0° position of our robots joints arbitrarily.

The three configurable parameters are:

1. **alpha**: rotación positiva o negativa (en radianes) alrededor del eje X del ”sistema de coordenadas actual”
2. **a**: distancia positiva, a lo largo de X, entre dos ejes de articulacion especificados en *unidades máquina* (mm o pulgadas) definidas en el archivo INI del sistema.
3. **d**: longitud positiva o negativa a lo largo de Z (también en *unidades máquina*)

The parameter sets are always derived in the same order and a set is completed by setting the d-parameter. This does not leave the Z-axis of our CS aligned with the next joint! This may seem confusing but sticking to this rule will yield a working set of parameters. Once the **d**-parameter is set, the X-axis of our-CS needs to point to the axis of the next joint.

9.2.5. Numbering of joints and parameters

The first joint in LinuxCNC is joint-0 (because in software counting starts with 0) while most publications start with the number *1*. That goes for all the parameters as well. That is, numbering starts with a-0, alpha-0, d-0 and ends with a-5, alpha-5 and d-5. Keep this in mind when following a publication to set up *genserkins* parameters.

9.2.6. How to start

Convention is to start by placing the reference-CS in the base of the robot with it’s Z-axis coinciding with the axis of the first joint and its X-axis pointing toward the next joint’s axis.

This will also result in the DRO values in LinuxCNC being referenced to that point. Having done so sets a-0 and alpha-0 to 0. The above mentioned publication (Craig) also sets d-0 to 0, which is confusing when a displacement offset is needed in order to have the reference-CS at the bottom of the base. Setting d-0 = to the displacement gives correct results. In this manner, the first set of parameters are alpha-0 = 0, a-0 = 0, d0 = displacement, and the X-axis of the CS points to the axis of the next joint (joint-1).

Derivation of the net set (alpha-1, a-1, d-1) follows — always using the same sequence all the way to the sixth set (alpha-5, a-5, d-5).

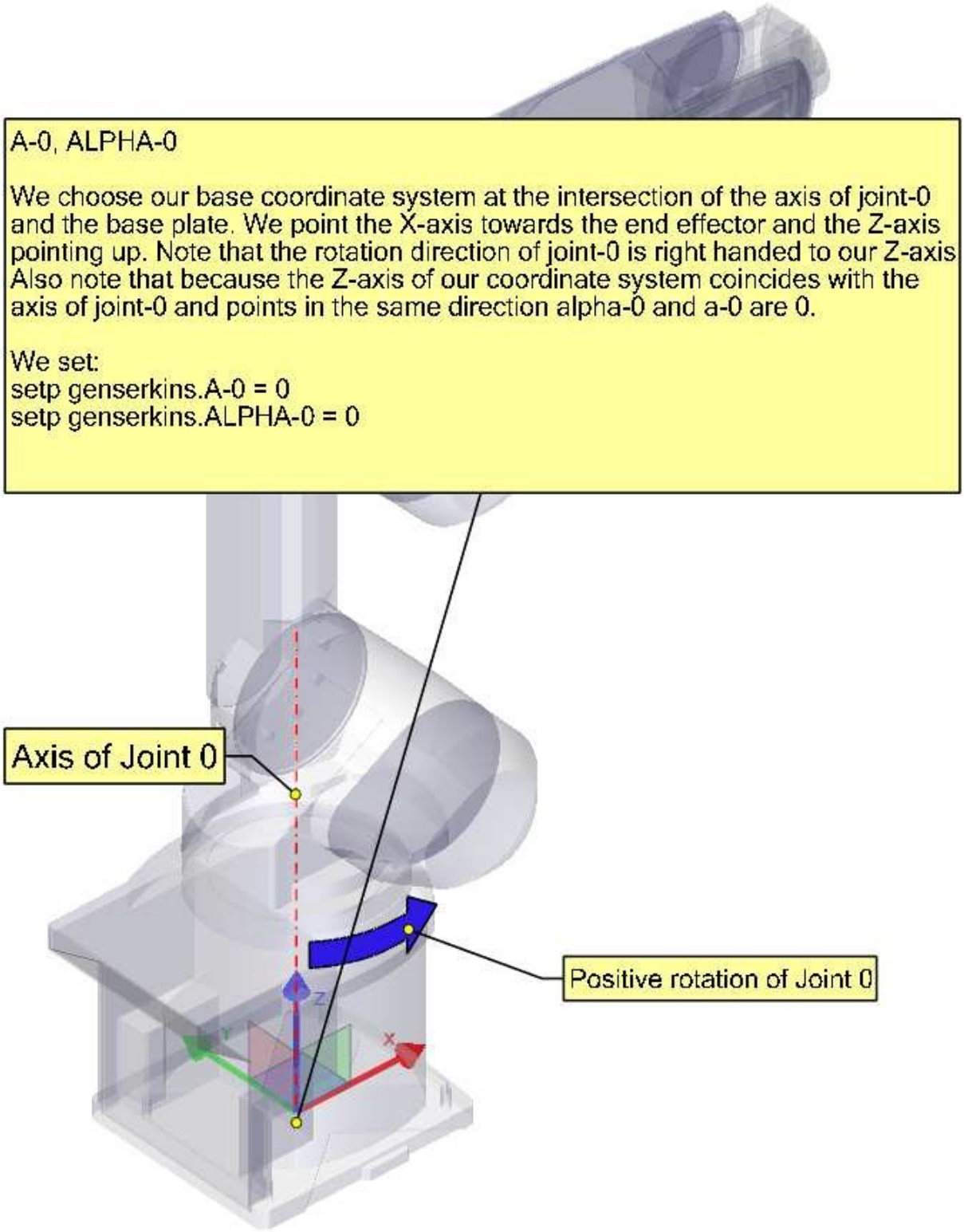
And thus, the TCP-CS of the end-effector is sitting in the center of the hand flange.

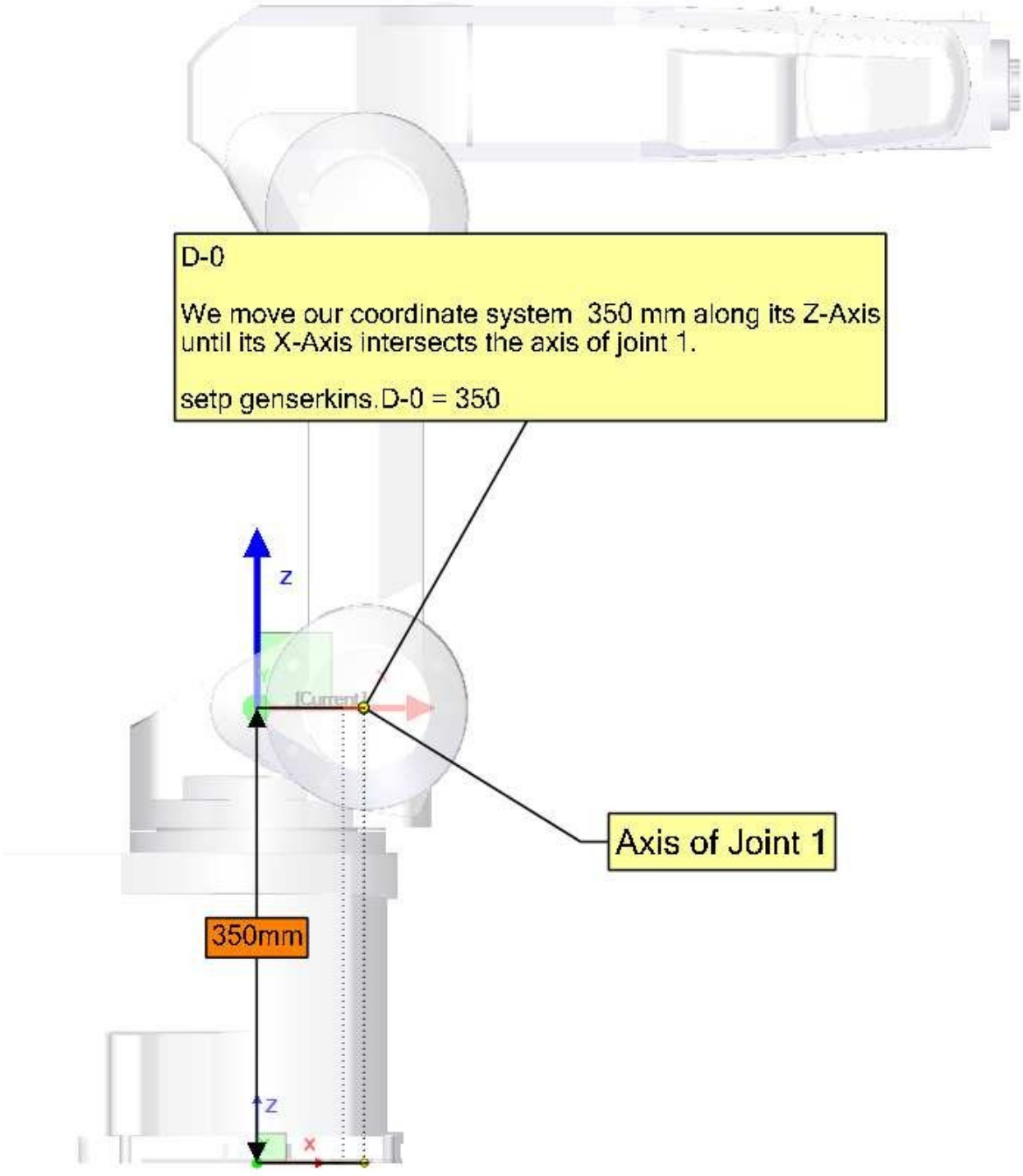
9.2.7. Special cases

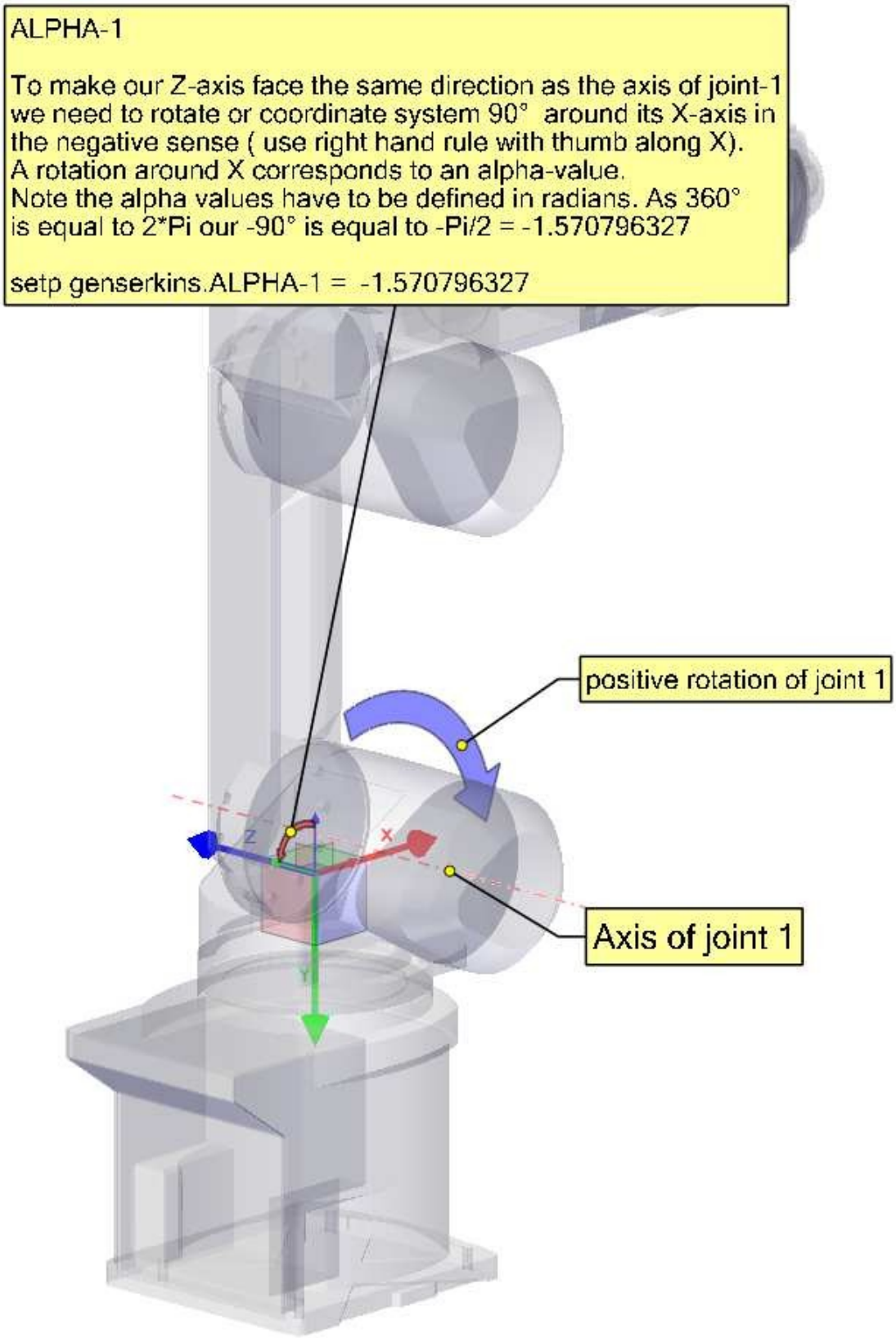
If the next joint-axis is parallel to the last then one could arbitrarily choose a value for the d-parameter but there is no point in setting it other than 0.

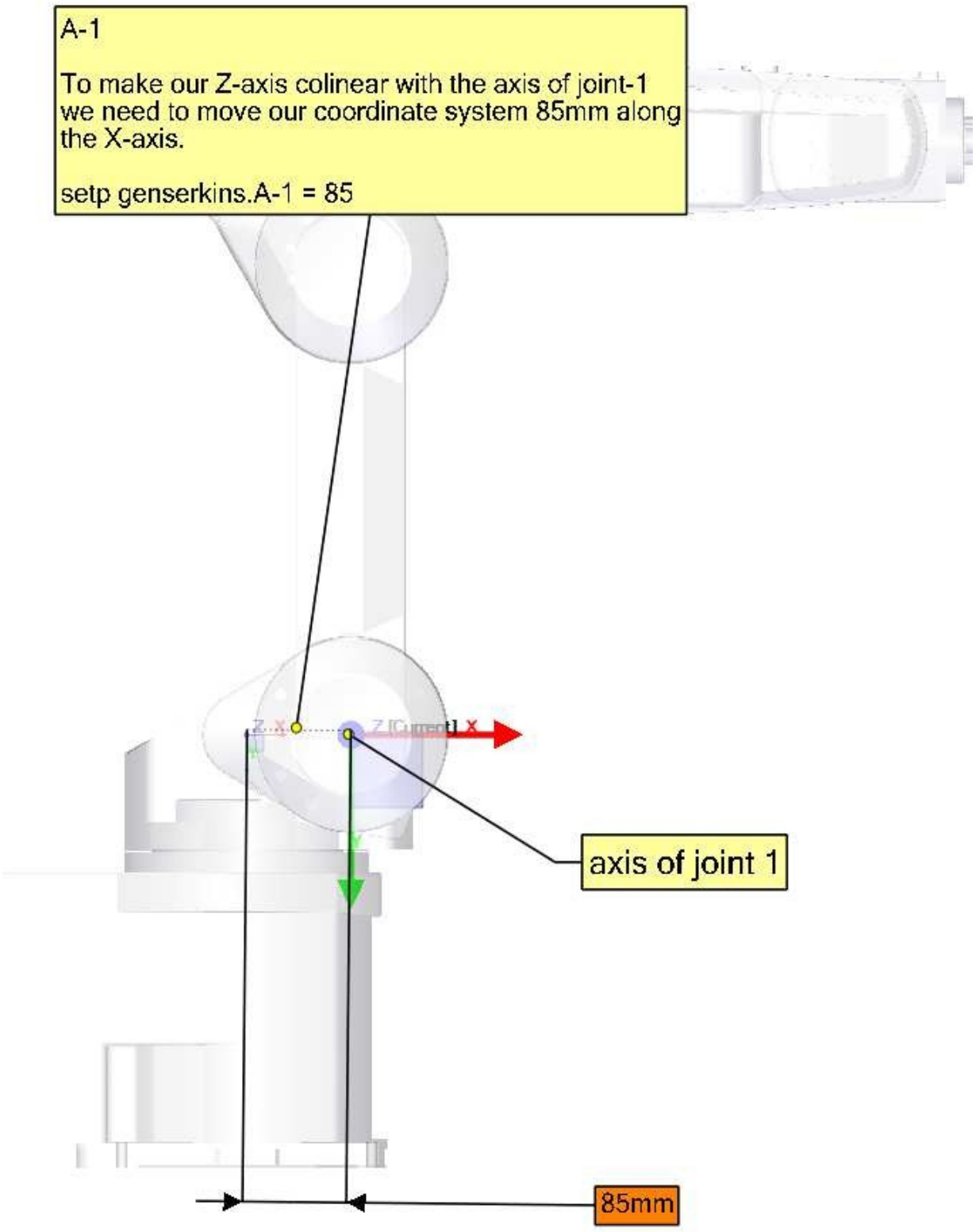
9.2.8. Detailed Example (RV-6SL)

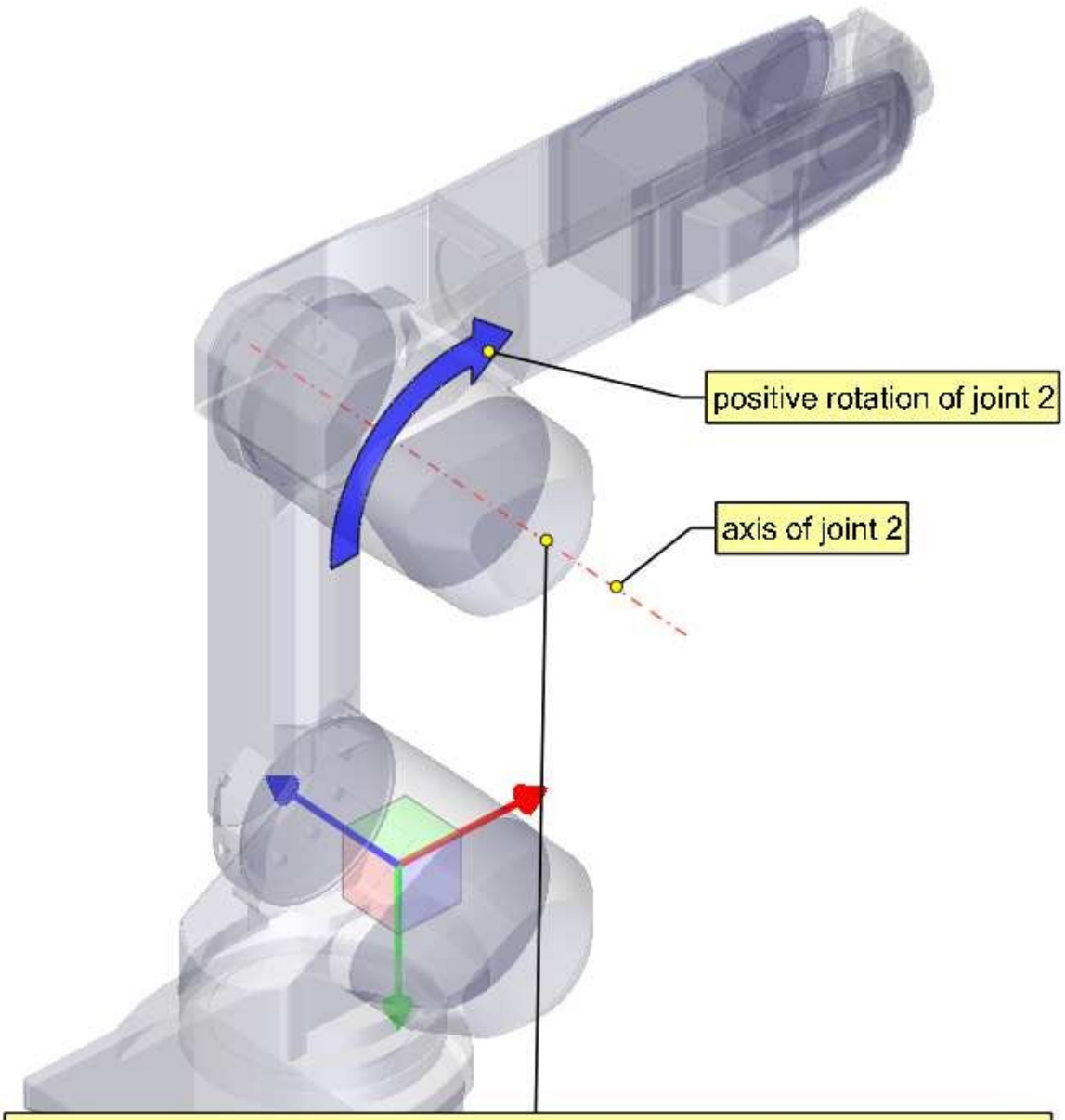
Described below is a method to derive the required ”modified DH-parameters” for a Mitsubishi RV-6SDL and how to set the parameters in the HAL file to be used with the *genserkins* kinematics in LinuxCNC. The necessary dimensions are best taken from a dimensional drawing provided by the manufacturer of the robot.


A-0, ALPHA-0
We choose our base coordinate system at the intersection of the axis of joint-0 and the base plate. We point the X-axis towards the end effector and the Z-axis pointing up. Note that the rotation direction of joint-0 is right handed to our Z-axis Also note that because the Z-axis of our coordinate system coincides with the axis of joint-0 and points in the same direction alpha-0 and a-0 are 0.
We set:
setp genserkins.A-0 = 0
setp genserkins.ALPHA-0 = 0
Axis of Joint 0
Positive rotation of Joint 0
Z
Y
X

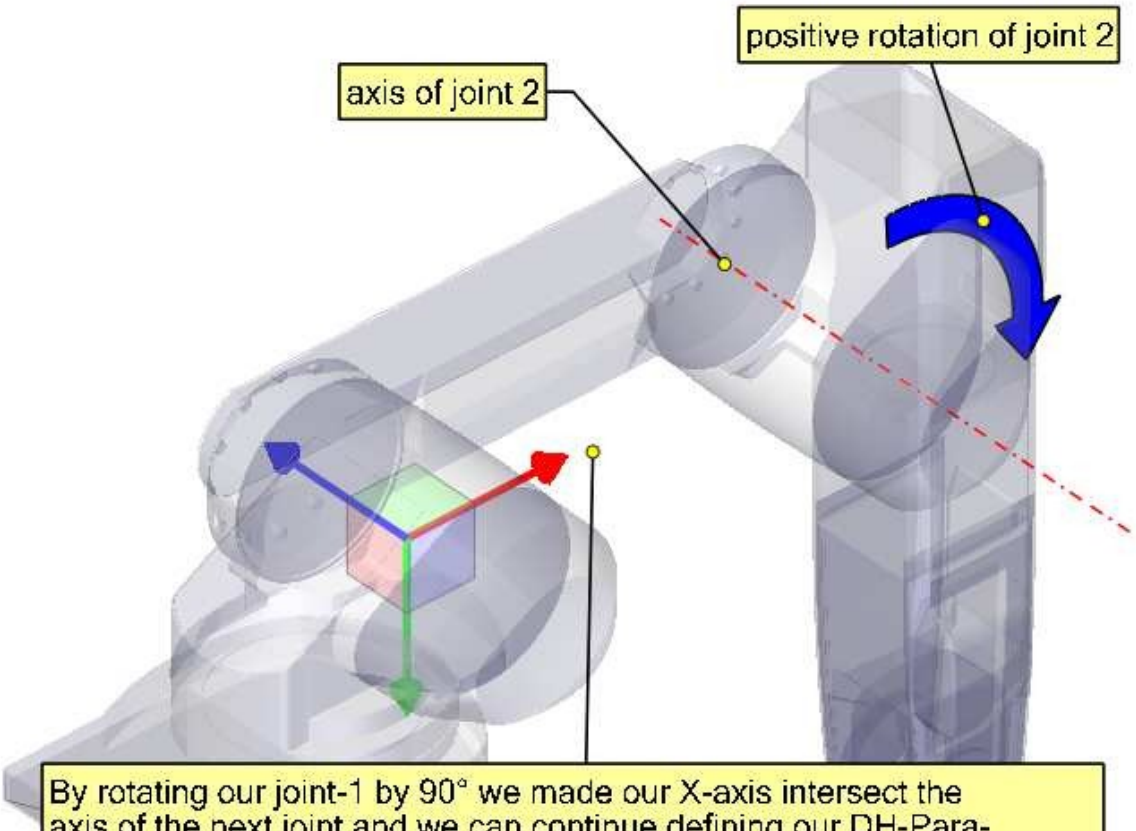

D-0
We move our coordinate system 350 mm along its Z-Axis until its X-Axis intersects the axis of joint 1.
setp genserkins.D-0 = 350
Z
Axis of Joint 1
350mm
Z
X


ALPHA-1
To make our Z-axis face the same direction as the axis of joint-1 we need to rotate or coordinate system 90° around its X-axis in the negative sense (use right hand rule with thumb along X).
A rotation around X corresponds to an alpha-value.
Note the alpha values have to be defined in radians. As 360° is equal to 2*Pi our -90° is equal to -Pi/2 = -1.570796327
setp genserkins.ALPHA-1 = -1.570796327
positive rotation of joint 1
Axis of joint 1
Z
X
Y


A-1
To make our Z-axis colinear with the axis of joint-1 we need to move our coordinate system 85mm along the X-axis.
setp genserkins.A-1 = 85
axis of joint 1
85mm



Note:
In order to make our X-axis intersect the axis of joint-2 we would need to rotate our coordinate system around its Z-axis. To do this we could, in theory, define theta-1 equal to -90°.
However genserkins does not allow the definition of theta values. In genserkins.c we see that the theta values for all joints are set to 0.
Now theta of course is the rotation of the joint itself and so is variable in an angular joint. Theta values are only used to define the home pose of a robot in the way of an offset.
So if we could define theta-1 equal to -90° we could define joint-1 to be oriented this way for 0°. Since we cannot define it we need to rotate joint-1 in a way so that our X-axis intersects with the axis of the next joint.



By rotating our joint-1 by 90° we made our X-axis intersect the axis of the next joint and we can continue defining our DH-Parameters.

D-1

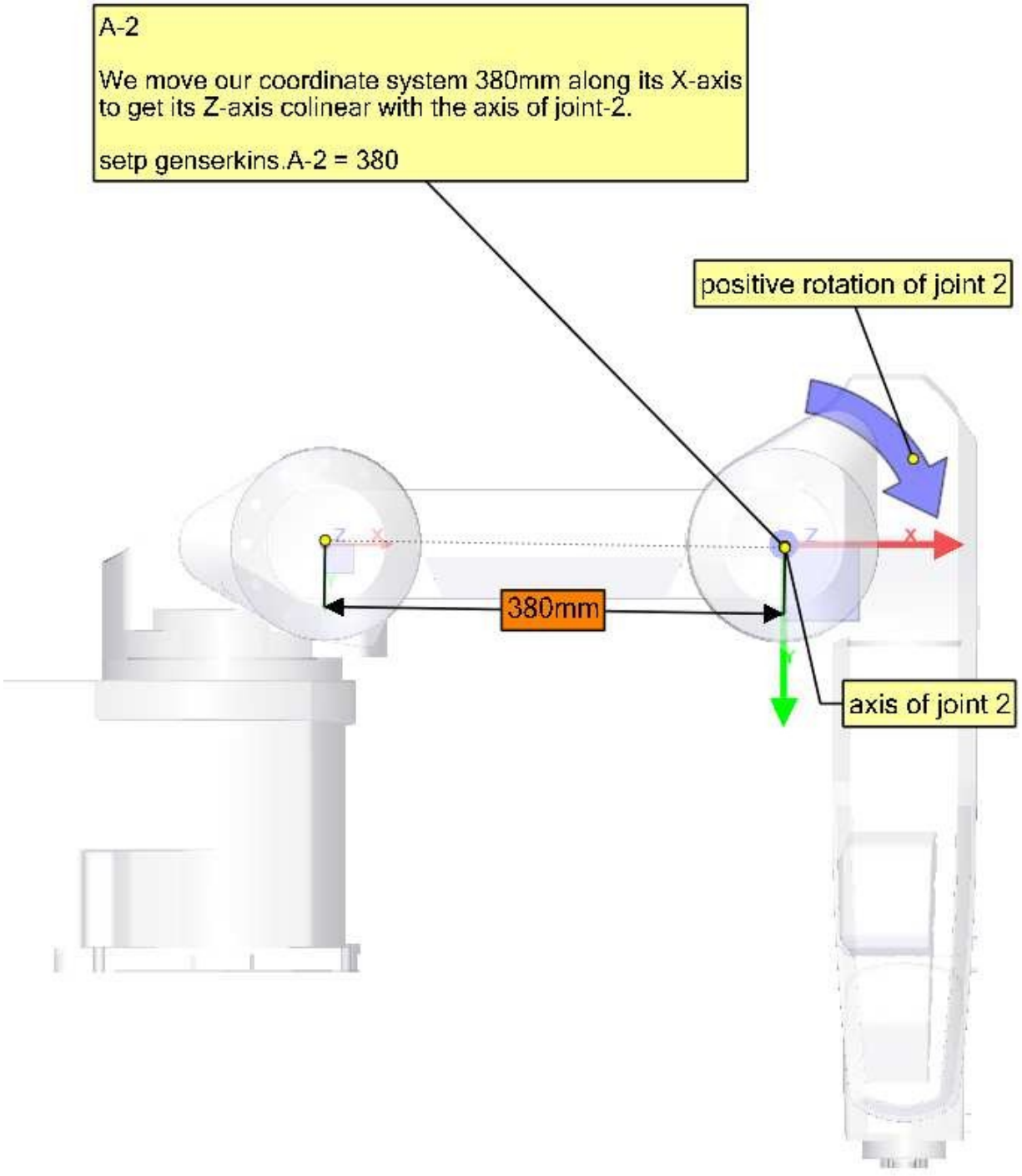
Since, after we rotated joint 1, our X-axis already intersects the axis of joint-2 we do not need to move our coordinate system along the Z-axis.

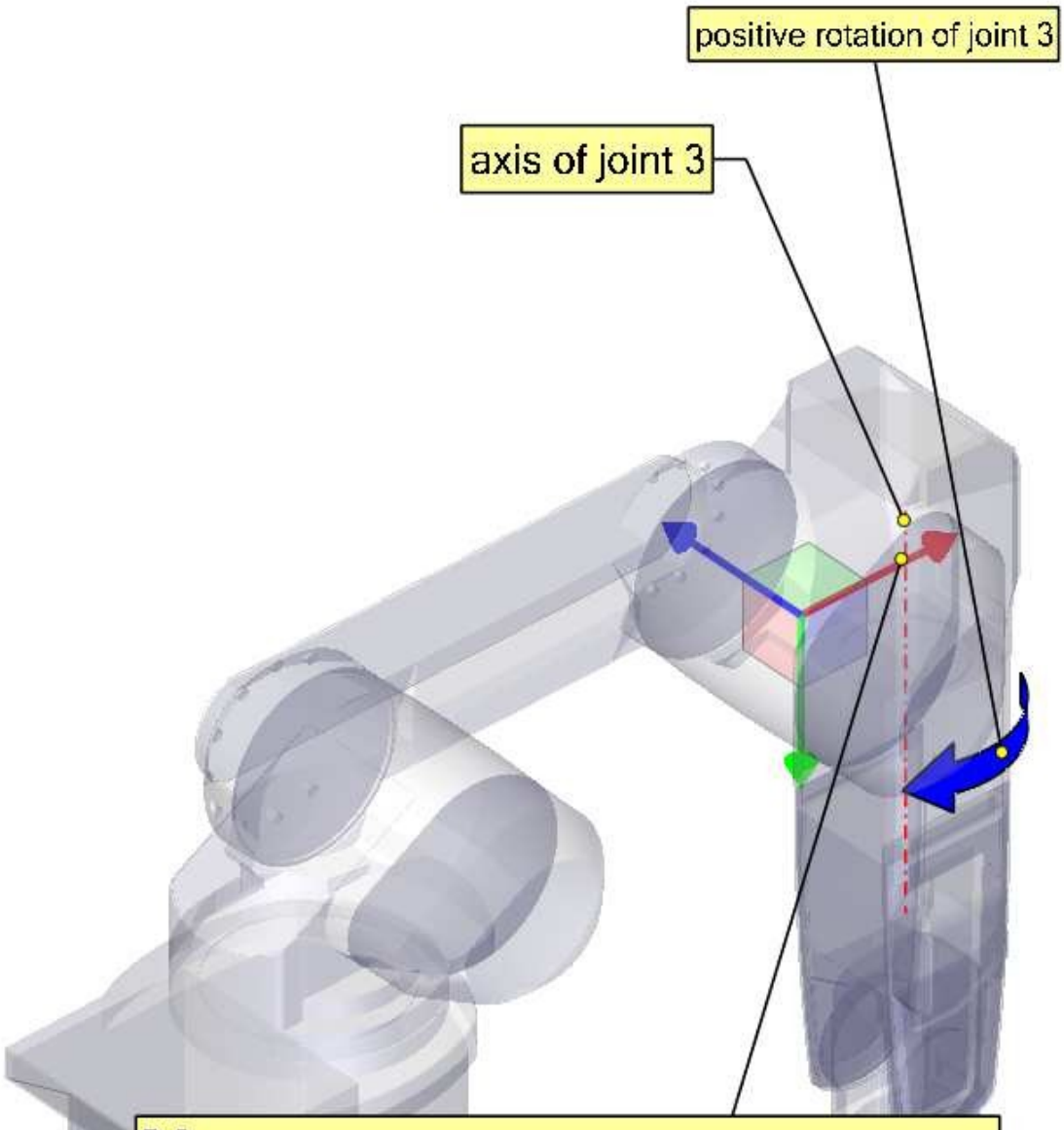
setp genserkins.D-1 = 0

ALPHA-2

The axis of joint -2 is parallel to the axis of joint -1 and points in the same direction. Thus we do not need to rotate our Z-axis.

setp genserkins.ALPHA-2 = 0


A-2
We move our coordinate system 380mm along its X-axis to get its Z-axis colinear with the axis of joint-2.
setp genserkins.A-2 = 380
positive rotation of joint 2
380mm
axis of joint 2



D-2

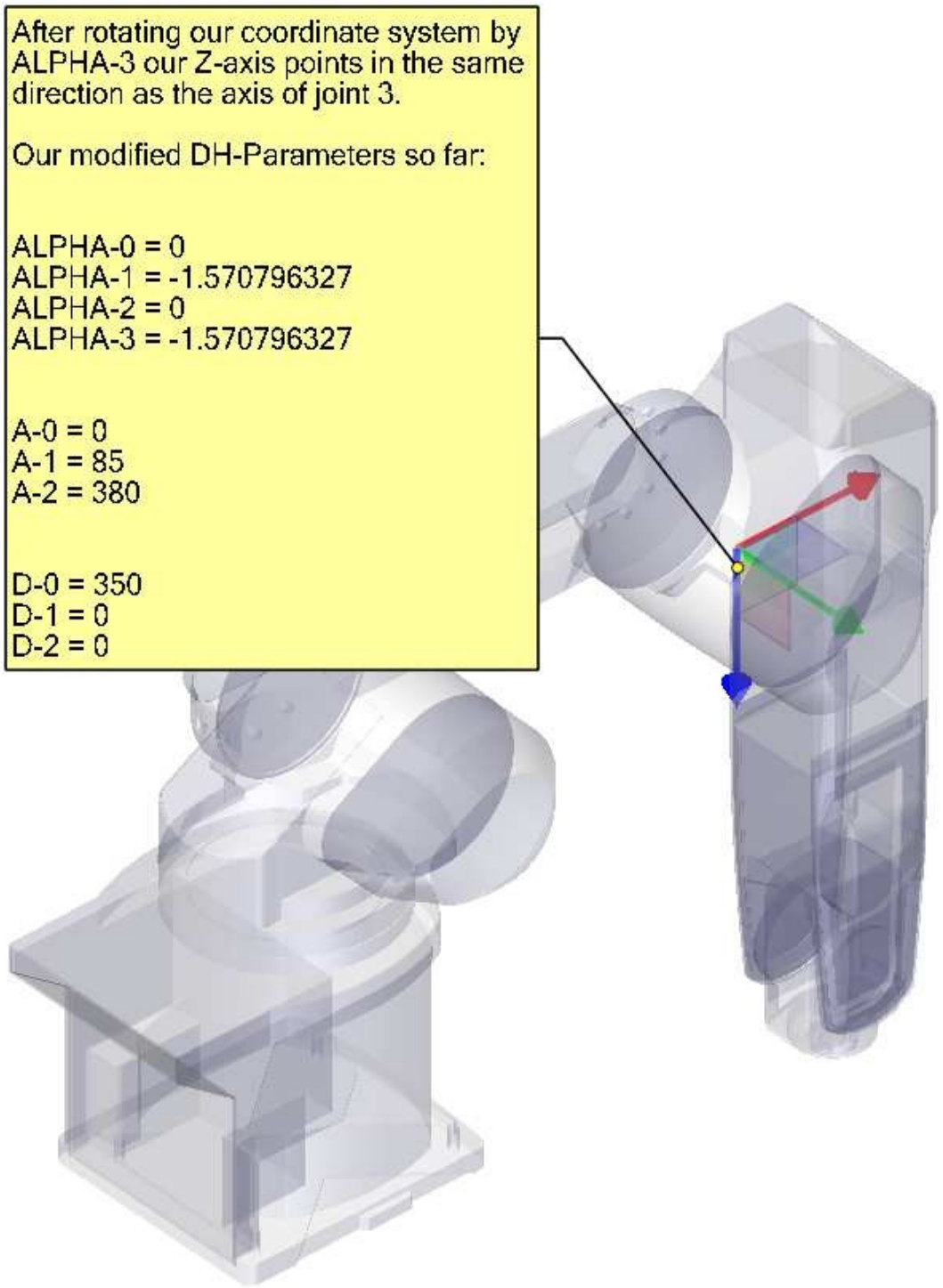
Our X-axis again already intersects the axis of the next joint 3.
So our d2 parameter is again 0.

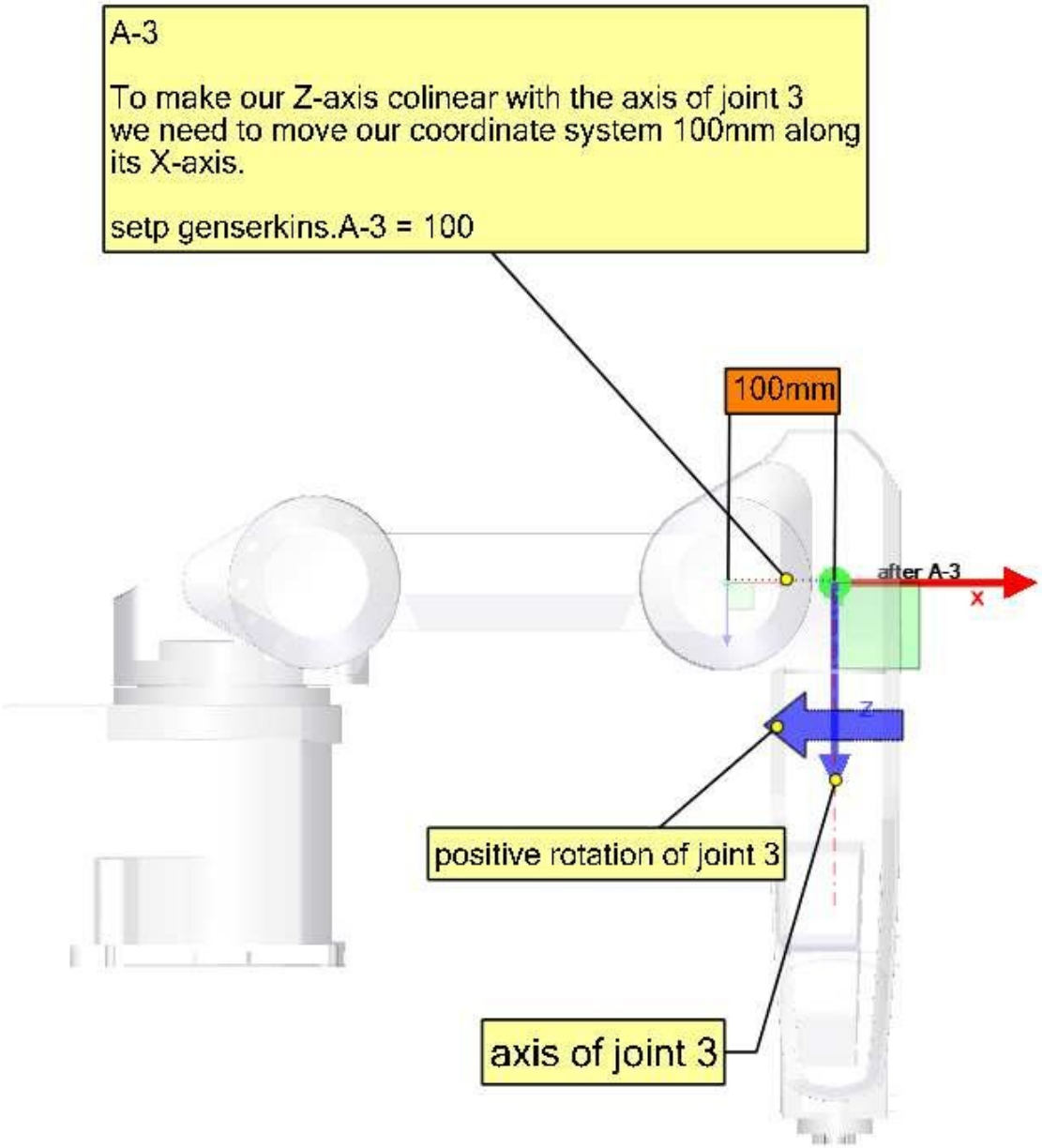
setp genserkins.D-2 = 0

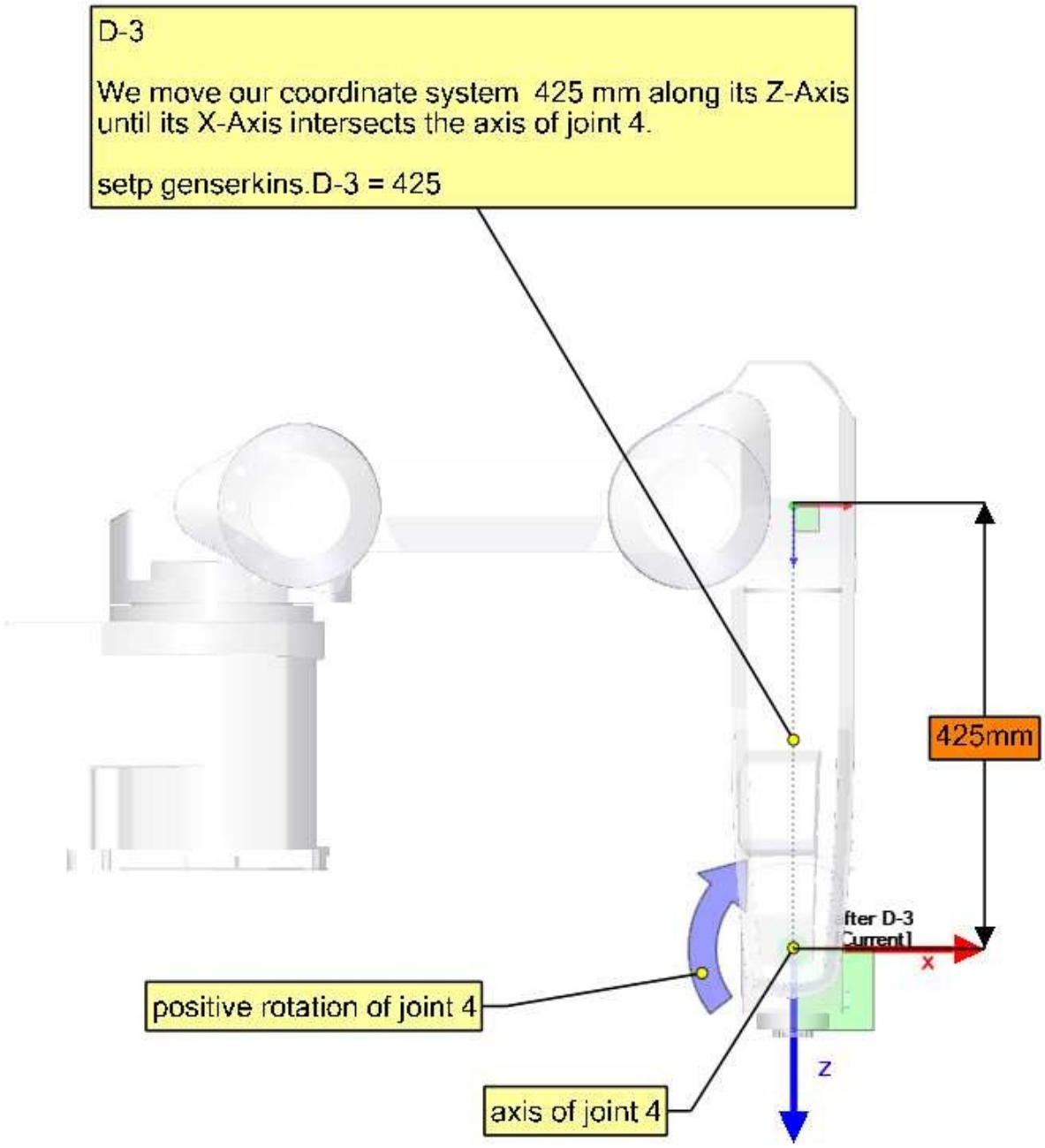
ALPHA-3

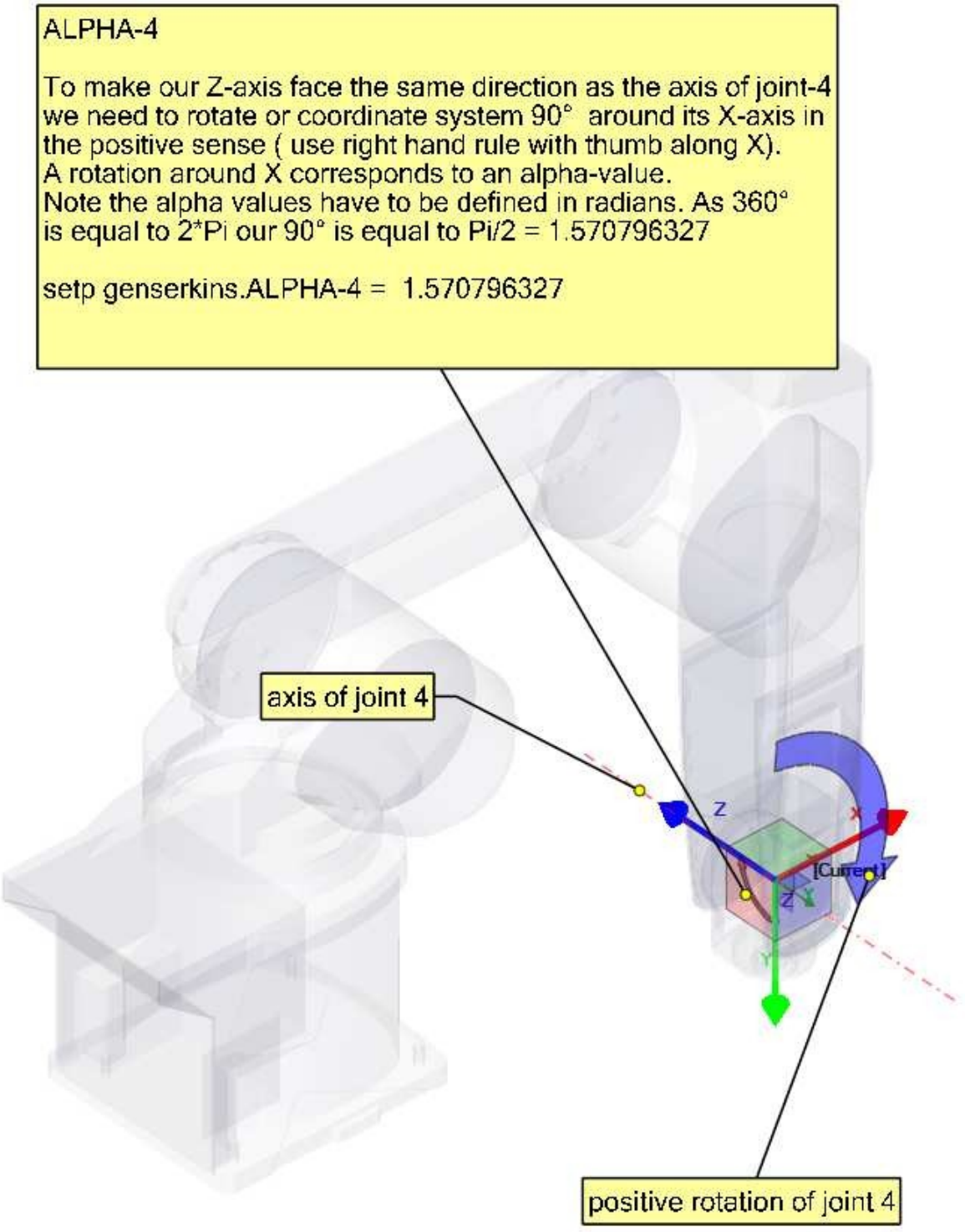
To make our Z-axis face the same direction as the axis of joint-3 we need to rotate or coordinate system 90° around its X-axis in the negative sense (use right hand rule with thumb along X).
A rotation around X corresponds to an alpha-value.
Note the alpha values have to be defined in radians. As 360° is equal to 2*Pi our -90° is equal to -Pi/2 = -1.570796327

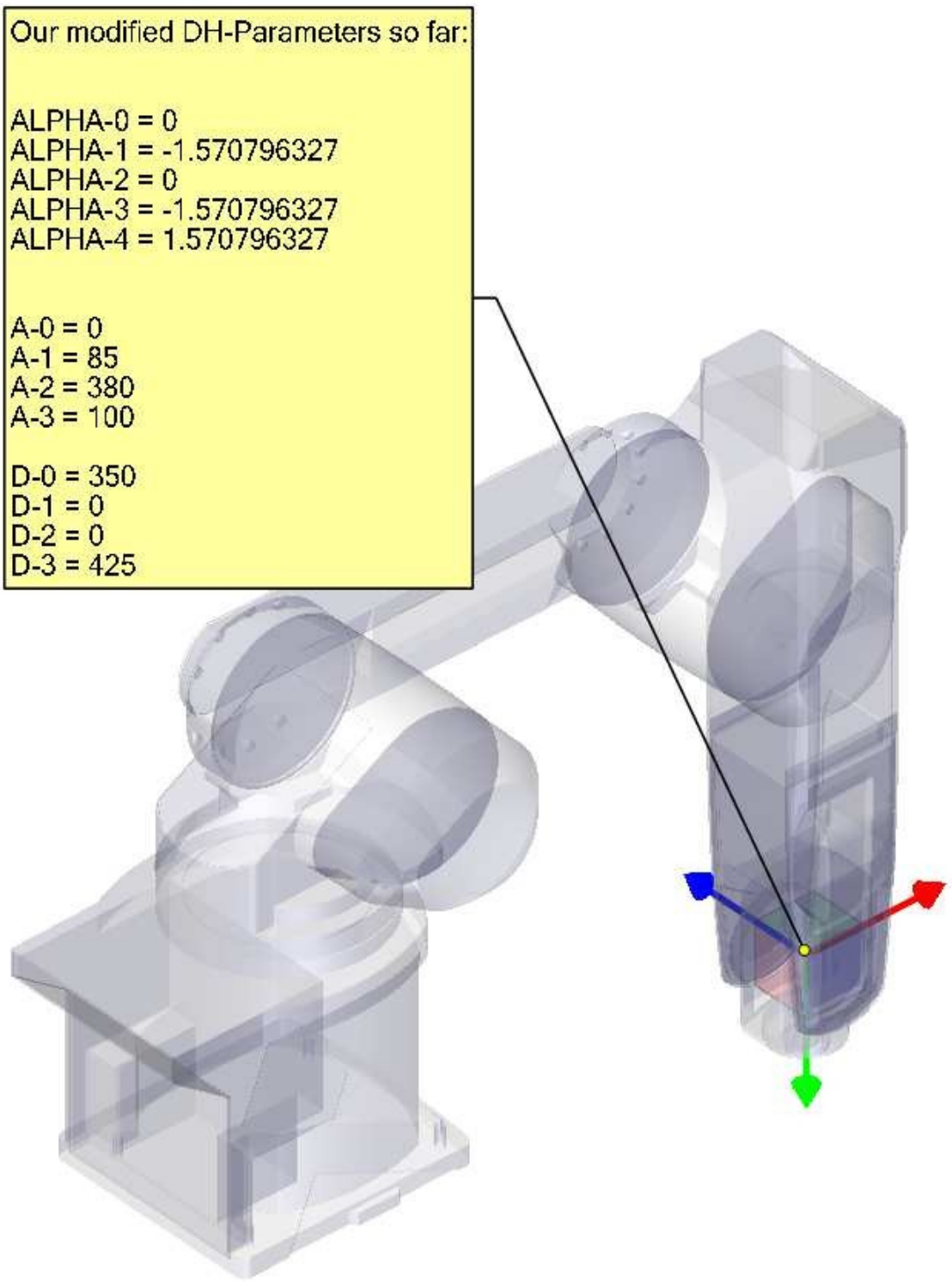
setp genserkins.ALPHA-3 = -1.570796327

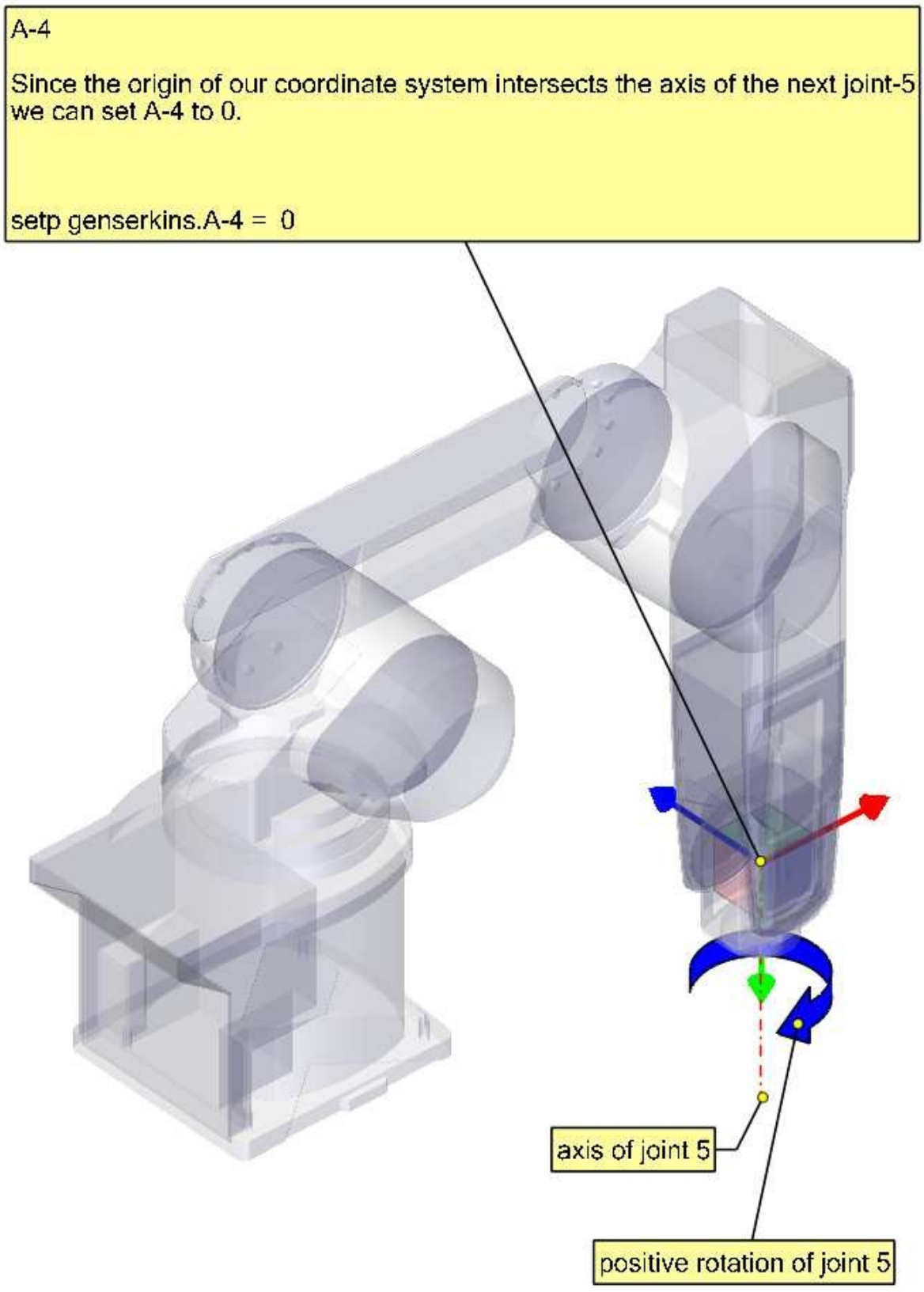

After rotating our coordinate system by ALPHA-3 our Z-axis points in the same direction as the axis of joint 3.
Our modified DH-Parameters so far:
ALPHA-0 = 0
ALPHA-1 = -1.570796327
ALPHA-2 = 0
ALPHA-3 = -1.570796327
A-0 = 0
A-1 = 85
A-2 = 380
D-0 = 350
D-1 = 0
D-2 = 0


A-3
To make our Z-axis colinear with the axis of joint 3 we need to move our coordinate system 100mm along its X-axis.
setp genserkins.A-3 = 100
100mm
after A-3
X
Z
positive rotation of joint 3
axis of joint 3


D-3
We move our coordinate system 425 mm along its Z-Axis until its X-Axis intersects the axis of joint 4.
setp genserkins.D-3 = 425
425mm
positive rotation of joint 4
axis of joint 4
X
Z


ALPHA-4
To make our Z-axis face the same direction as the axis of joint-4 we need to rotate or coordinate system 90° around its X-axis in the positive sense (use right hand rule with thumb along X).
A rotation around X corresponds to an alpha-value.
Note the alpha values have to be defined in radians. As 360° is equal to 2*Pi our 90° is equal to Pi/2 = 1.570796327
setp genserkins.ALPHA-4 = 1.570796327
axis of joint 4
positive rotation of joint 4
Z
X
Y
[Current]


Our modified DH-Parameters so far:
ALPHA-0 = 0
ALPHA-1 = -1.570796327
ALPHA-2 = 0
ALPHA-3 = -1.570796327
ALPHA-4 = 1.570796327
A-0 = 0
A-1 = 85
A-2 = 380
A-3 = 100
D-0 = 350
D-1 = 0
D-2 = 0
D-3 = 425


A-4
Since the origin of our coordinate system intersects the axis of the next joint-5 we can set A-4 to 0.
setp genserkins.A-4 = 0
axis of joint 5
positive rotation of joint 5

D-4

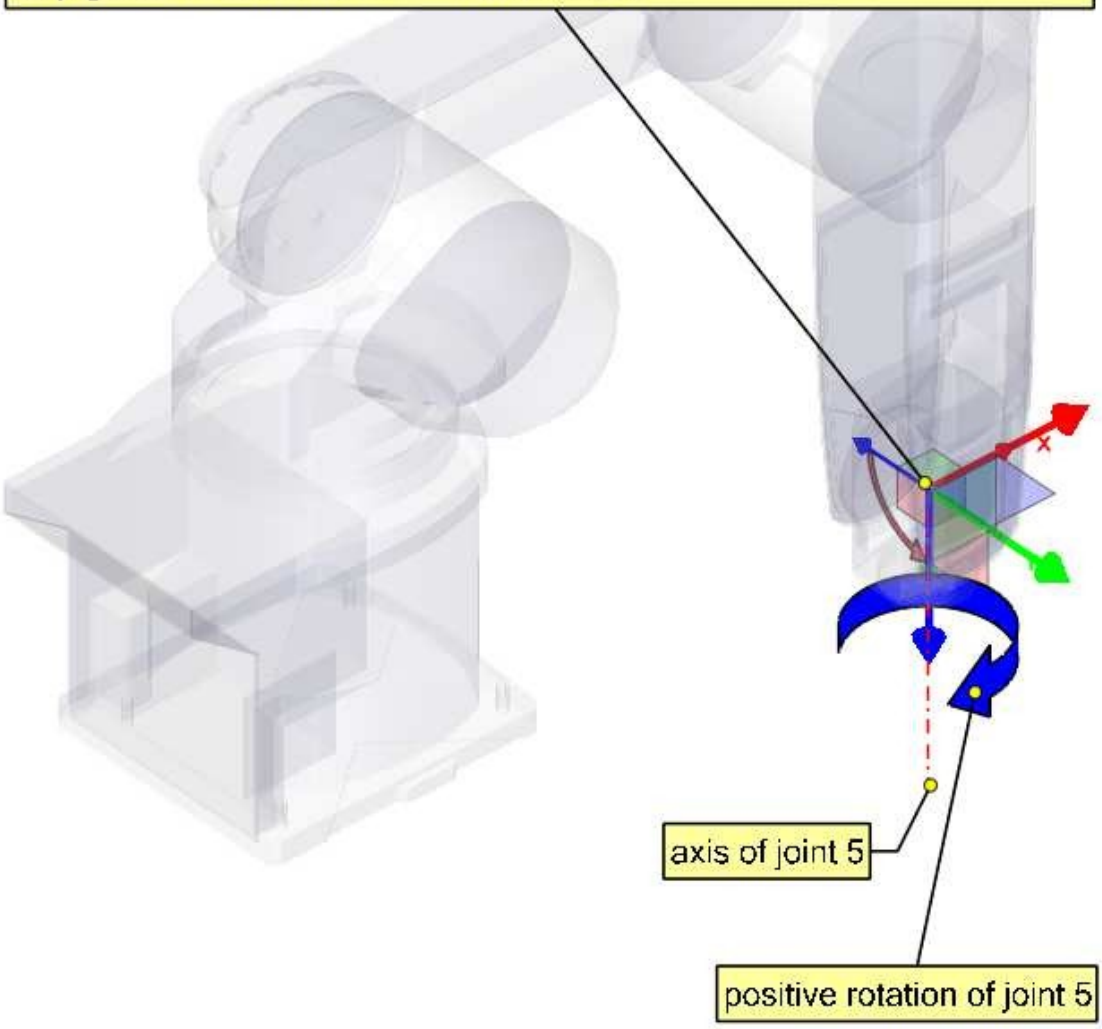
Since the origin of our coordinate system lies on the axis of the next joint our d4 parameter is also 0.

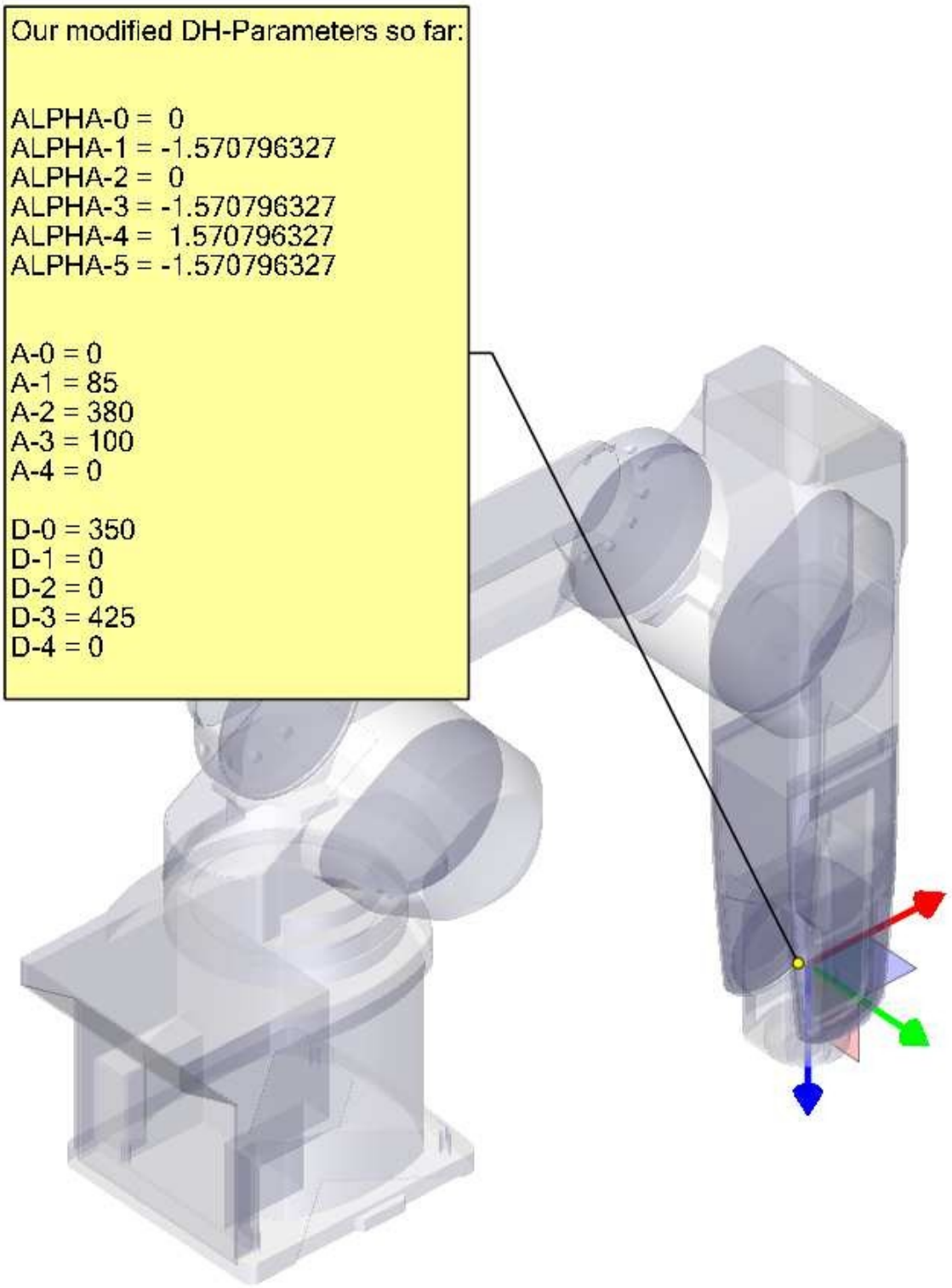
setp genserkins.D-4 = 0

ALPHA-5

To make our Z-axis face the same direction as the axis of joint-5 we need to rotate or coordinate system 90° around its X-axis in the negative sense (use right hand rule with thumb along X).
A rotation around X corresponds to an alpha-value.
Note the alpha values have to be defined in radians. As 360° is equal to 2*Pi our -90° is equal to -Pi/2 = -1.570796327

setp genserkins.ALPHA-5 = -1.570796327



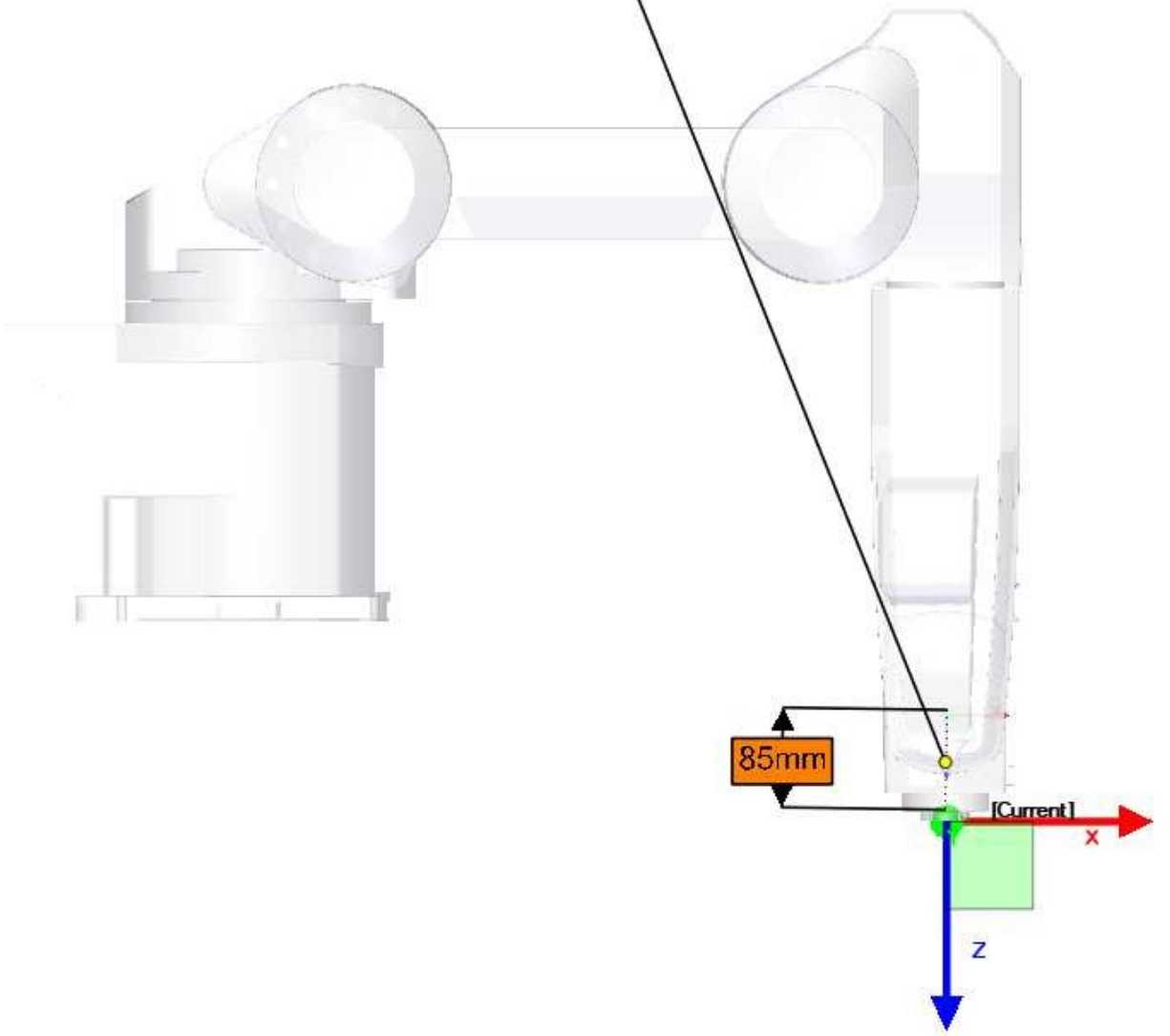

Our modified DH-Parameters so far:
ALPHA-0 = 0
ALPHA-1 = -1.570796327
ALPHA-2 = 0
ALPHA-3 = -1.570796327
ALPHA-4 = 1.570796327
ALPHA-5 = -1.570796327
A-0 = 0
A-1 = 85
A-2 = 380
A-3 = 100
A-4 = 0
D-0 = 350
D-1 = 0
D-2 = 0
D-3 = 425
D-4 = 0

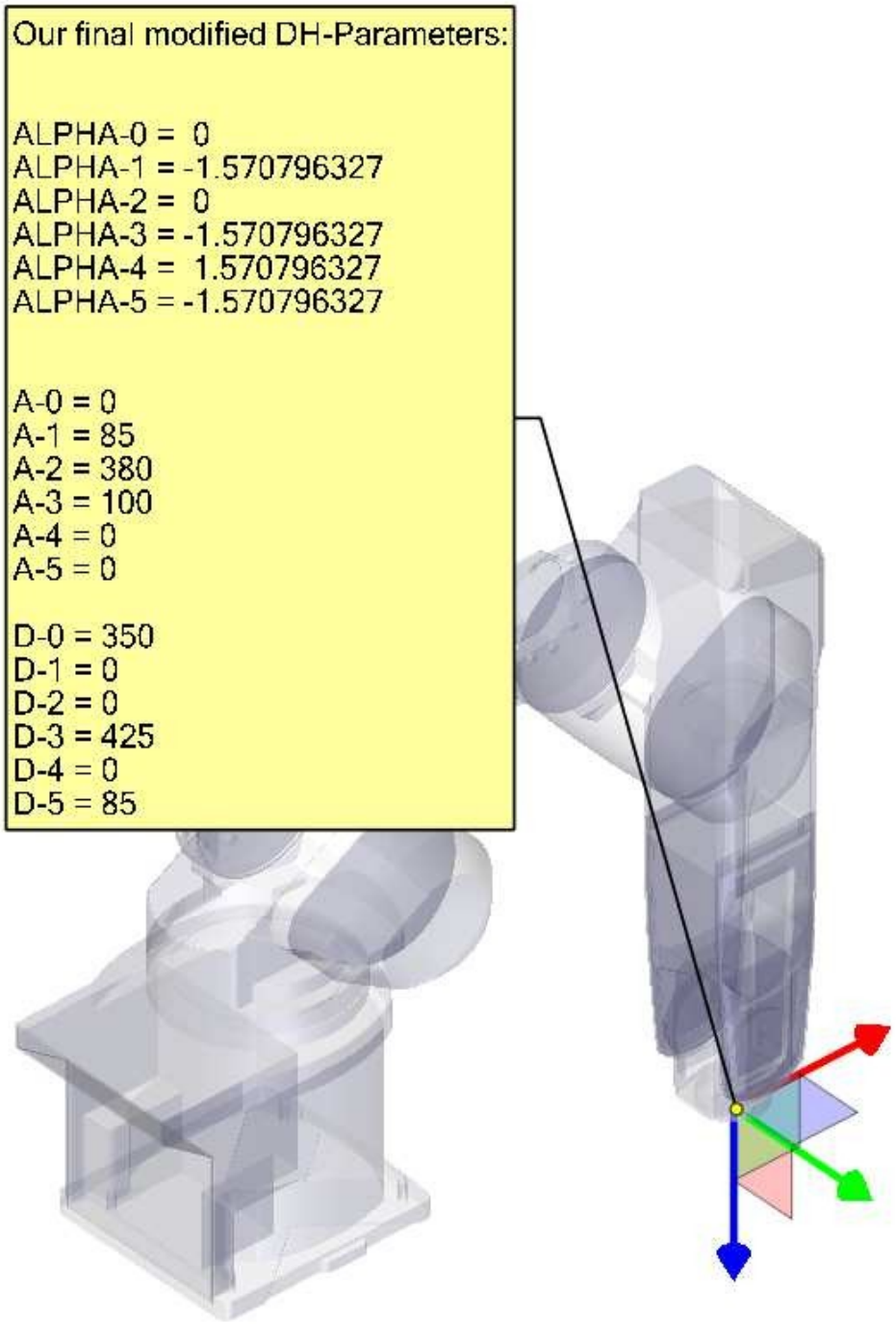
D-5

We move our coordinate system for 85mm along its Z-axis.

With this we have finished setting up our modified DH- parameters and leaves our coordinate system at the center of the hand flange.

setp genserkins.D-5 = 85




Our final modified DH-Parameters:
ALPHA-0 = 0
ALPHA-1 = -1.570796327
ALPHA-2 = 0
ALPHA-3 = -1.570796327
ALPHA-4 = 1.570796327
ALPHA-5 = -1.570796327
A-0 = 0
A-1 = 85
A-2 = 380
A-3 = 100
A-4 = 0
A-5 = 0
D-0 = 350
D-1 = 0
D-2 = 0
D-3 = 425
D-4 = 0
D-5 = 85

9.2.9. Credits

¡Gracias al usuario Aciera por todo el texto y los gráficos para el robot RV-6SL!

9.3. 5-Axis Kinematics

9.3.1. Introducción

Coordinated multi-axis CNC machine tools controlled with LinuxCNC, require a special kinematics component for each type of machine. This chapter describes some of the most popular 5-axis machine configurations and then develops the forward (from work to joint coordinates) and inverse (from joint to work) transformations in a general mathematical process for two types of machine.

The kinematics components are given as well as vismach simulation models to demonstrate their behaviour on a computer screen. Examples of HAL file data are also given.

Note that with these kinematics, the rotational axes move in the opposite direction of what the convention is. See section [”rotational axes”](https://linuxcnc.org/docs/html/gcode/machining-center.html#_rotatio for details.

9.3.2. 5-Axis Machine Tool Configurations

In this section we deal with the typical 5-axis milling or router machines with five joints or degrees-of-freedom which are controlled in coordinated moves.

3-axis machine tools cannot change the tool orientation, so 5-axis machine tools use two extra axes to set the cutting tool in an appropriate orientation for efficient machining of freeform surfaces.

Typical 5-axis machine tool configurations are shown in Figs. 3, 5, 7 and 9-11 [1,2] in section Figures.

The kinematics of 5-axes machine tools are much simpler than that of 6-axis serial arm robots, since 3 of the axes are normally linear axes and only two are rotational axes.

9.3.3. Tool Orientation and Location

CAD/CAM systems are typically used to generate the 3D CAD models of the workpiece as well as the CAM data for input to the CNC 5-axis machine. The tool or cutter location (CL) data, is composed of the cutter tip position and the cutter orientation relative to the workpiece coordinate system. Two vectors, as generated by most CAM systems and shown in Fig. 1, contain this information:

$$K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} \quad \text{orientation vector;} \qquad Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \quad \text{position vector} \qquad (1)$$

The K vector is equivalent to the 3rd vector from the pose matrix E_6 that was used in the 6-axis robot kinematics [3] and the Q vector is equivalent to the 4th vector of E_6. In MASTERCAM for example this information is contained in the intermediate output ”.nci” file.

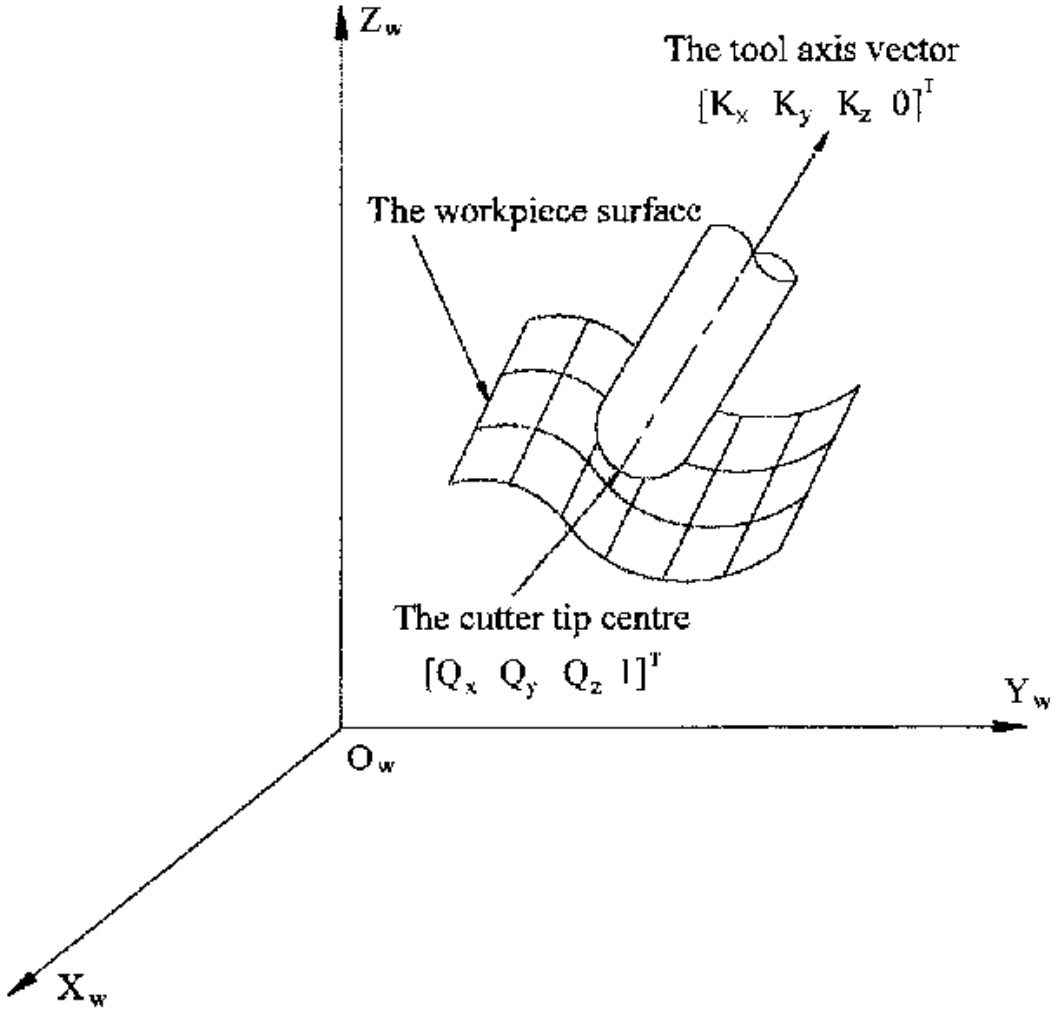


Figura 9.2: Cutter location data

9.3.4. Translation and Rotation Matrices

Las transformaciones homogéneas proporcionan una manera simple de describir las matemáticas de la cinemática de máquinas multieje. Una transformación del espacio H es una matriz 4x4 y puede representar transformaciones de translacion y rotación. Dado un punto x, y, z descrito por un vector $u=\{x, y, z, 1\}^T$, su transformación v está representada por la matriz producto

$$v = H \cdot u$$

There are four fundamental transformation matrices on which 5-axis kinematics can be based:

$$T(a,b,c) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R(X,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\theta & -S\theta & 0 \\ 0 & S\theta & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$R(Y,\theta) = \begin{bmatrix} C\theta & 0 & S\theta & 0 \\ 0 & 1 & 0 & 0 \\ -S\theta & 0 & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R(Z,\theta) = \begin{bmatrix} C\theta & -S\theta & 0 & 0 \\ S\theta & C\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

The matrix T(a,b,c) implies a translation in the X, Y, Z coordinate directions by the amounts a, b, c respectively. The R matrices imply rotations of the angle theta about the X, Y and Z coordinate axes respectively. The *C* and *S* symbols refer to cosine and sine functions respectively.

9.3.5. Configuraciones de 5 ejes con mesas giratorias/inclinables

In these machine tools the two rotational axes mount on the work table of the machine. Two forms are typically used:

- A rotary table which rotates about the vertical Z-axes (C-rotation, secondary) mounted on a tilting table which rotates about the X- or Y-axis (A- or B-rotation, primary). The workpiece is mounted on the rotary table.
- A tilting table which rotates about the X- or Y-axis (A- or B-rotation, secondary) is mounted on a rotary table which rotates about the Z-axis (C-rotation, primary), with the workpiece on the tilting table.

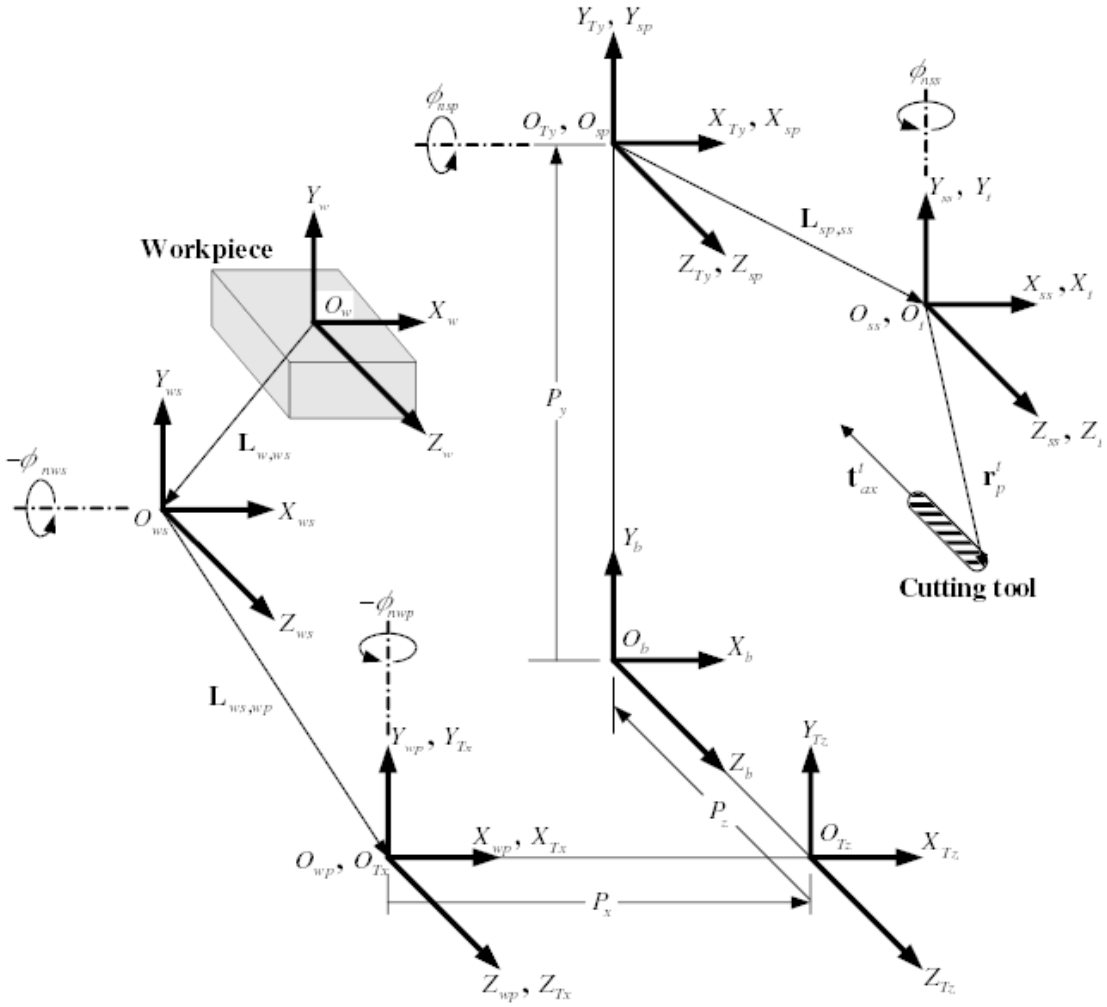


Figura 9.3: General configuration and coordinate systems

Se puede considerar que una máquina multieje consiste en una serie de eslabones conectados por articulaciones. Al asignar un sistema de coordenadas en cada eslabon de la máquina, y usando transformaciones homogéneas, podemos describir la posición relativa y la orientación entre estos sistemas de coordenadas

We need to describe a relationship between the workpiece coordinate system and the tool coordinate system. This can be defined by a transformation matrix ${}^{w}A_{t}$, which can be found by subsequent transformations between the different structural elements or links of the machine, each with its own defined coordinate system. In general such a transformation may look as follows:

$$ {}^{w}A_{t} = {}^{w}A_{1} \cdot {}^{1}A_{2} \cdot {}^{2}A_{3} \cdots {}^{n}A_{t} \qquad (4) $$

where each matrix ${}^{i-1}A_{j}$ is a translation matrix T or a rotation matrix R of the form (2,3).

Matrix multiplication is a simple process in which the elements of each row of the lefthand matrix A is multiplied by the elements of each column of the righthand matrix B and summed to obtain an element in the result matrix C, ie.

$$ C_{ij} = \sum_{k=1,n}^{n} A_{ik} B_{kj}; \quad i = 1,n; \quad j = 1,n $$

In Fig. 2 a generic configuration with coordinate systems is shown [4]. It includes table rotary/tilting axes as well as spindle rotary/tilting axes. Only two of the rotary axes are actually used in a machine tool.

First we will develop the transformations for the first type of configuration mentioned above, ie. a table tilting/rotary (trt) type with no rotating axis offsets. We may give it the name xyzac-trt configuration.

We also develop the transformations for the same type (xyzac-trt), but with rotating axis offsets.

Then we develop the transformations for a xyzbc-trt configuration with rotating axis offsets.

9.3.5.1. Transformations for a xyzac-trt machine tool with work offsets

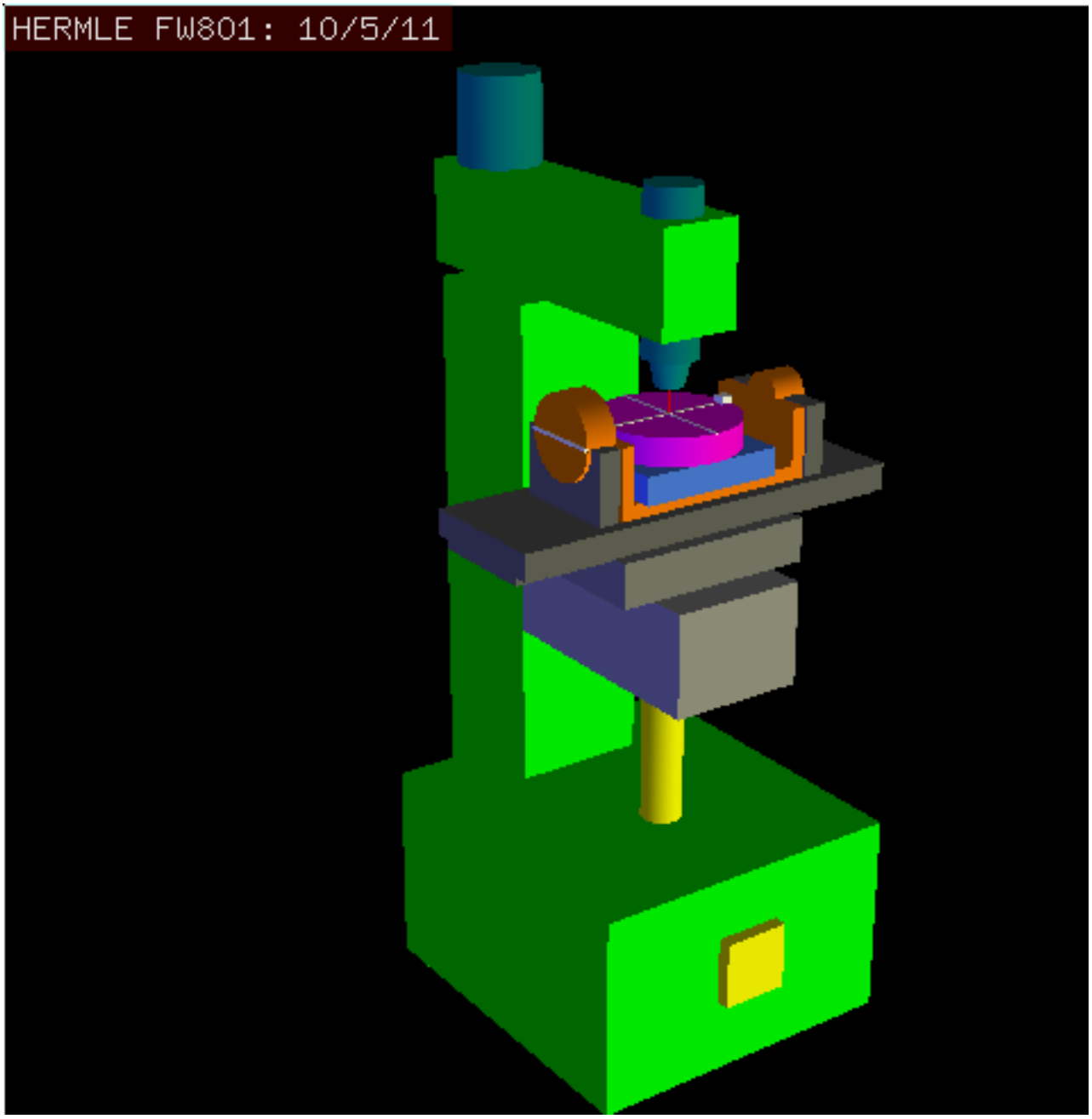


Figura 9.4: vismach model of xyzac-trt with coincident rotation axes

We deal here with a simplified configuration in which the tilting axis and rotary axis intersects at a point called the pivot point as shown in Fig. 4. therefore the two coordinate systems O_{ws} and O_{wp} of Fig. 2 are coincident.

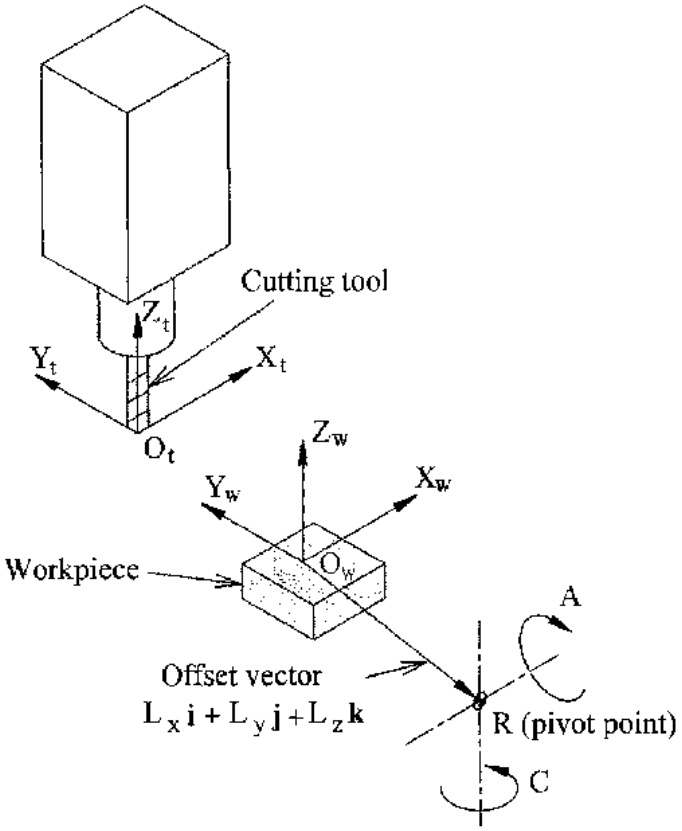


Figura 9.5: Table tilting/rotary configuration

The transformation can be defined by the sequential multiplication of the matrices:

$$ {}^{w}A_t = {}^{w}A_C \cdot {}^{C}A_A \cdot {}^{A}A_P \cdot {}^{P}A_t \qquad (5) $$

with the matrices built up as follows:

$$ {}^{w}A_C = \begin{bmatrix} 1 & 0 & 0 & L_x \\ 0 & 1 & 0 & L_y \\ 0 & 0 & 1 & L_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{C}A_A = \begin{bmatrix} C_C & S_C & 0 & 0 \\ -S_C & C_C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6) $$

$$ {}^{A}A_P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_A & S_A & 0 \\ 0 & -S_A & C_A & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{P}A_t = \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (7) $$

In these equations L_x, L_y, L_z defines the offsets of the pivot point of the two rotary axes A and C relative to the workpiece coordinate system origin. Furthermore, P_x, P_y, P_z are the relative distances of the pivot point to the cutter tip position, which can also be called the ”joint coordinates” of the pivot point. The pivot point is at the intersection of the two rotary axes. The signs of the S_A and S_C terms are different to those in [2,3] since there the table rotations are negative relative to the workpiece coordinate axes (note that sin(-theta) = -sin(theta), cos(-theta) = cos(theta)).

When multiplied in accordance with (5), we obtain:

$$ {}^{w}A_t = \begin{bmatrix} C_C & S_C C_A & S_C S_A & C_C P_x + S_C C_A P_y + S_C S_A P_z + L_x \\ -S_C & C_C C_A & C_C S_A & -S_C P_x + C_C C_A P_y + C_C S_A P_z + L_y \\ 0 & -S_A & C_A & -S_A P_y + C_A P_z + L_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (8) $$

We can now equate the third column of this matrix with our given tool orientation vector K, ie.:

$$ K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} = \begin{bmatrix} S_C S_A \\ C_C S_A \\ C_A \\ 0 \end{bmatrix} \qquad (9) $$

From these equations we can solve for the rotation angles $theta_A$, $theta_C$. From the third row we find:

$$ \theta_A = cos^{-1}(K_z) \qquad (0 < \theta_A < \pi) \qquad (10) $$

y al dividir la primera fila por la segunda fila encontramos:

$$ \theta_C = tan2^{-1}(K_x, K_y) \qquad (-\pi < \theta_C < \pi) \qquad (11) $$

These relationships are typically used in the CAM post-processor to convert the tool orientation vectors to rotation angles.

Equating the last column of (8) with the tool position vector Q, we can write:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C P_x + S_C C_A P_y + S_C S_A P_z + L_x \\ -S_C P_x + C_C C_A P_y + C_C S_A P_z + L_y \\ -S_A P_y + C_A P_z + L_z \\ 1 \end{bmatrix} \qquad (12) $$

El vector en el lado derecho también se puede escribir como el producto de una matriz y un vector, que da como resultado:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C & S_C C_A & S_C S_A & L_x \\ -S_C & C_C C_A & C_C S_A & L_y \\ 0 & -S_A & C_A & L_z \\ 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = {}^{Q}A_P \cdot P \qquad (13) $$

This can be expanded to give

$$ \begin{aligned} Q_x &= C_c P_x + S_C C_A P_y + S_C S_A P_z + L_x \\ Q_y &= -S_C P_x + C_C C_A P_y + C_C S_A P_z + L_y \\ Q_z &= -S_A P_y + C_A P_z + L_z \end{aligned} \qquad (14) $$

which is the *forward transformation* of the kinematics.

We can solve for P from equation (13) as $P = ({}^{Q}A_{P})^{-1} * Q$. Noting that the square matrix is a homogeneous 4x4 matrix containing a rotation matrix R and translation vector q, for which the inverse can be written as:

$$ {}^{q}A_{p} = \begin{bmatrix} R & q \\ 0 & 1 \end{bmatrix} \qquad ({}^{Q}A_{P})^{-1} = \begin{bmatrix} R^{T} & -R^{T}q \\ 0 & 1 \end{bmatrix} \qquad (15) $$

where R^T is the transpose of R (rows and columns swappped). We therefore obtain:

$$ \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C & -S_C & 0 & -C_C L_x + S_C L_y \\ S_C C_A & C_C C_A & -S_A & -S_C C_A L_x - C_C C_A L_y + S_A L_z \\ S_C S_A & C_C S_A & C_A & -S_C S_A L_x - C_C S_A L_y - C_A L_z \\ 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \qquad (16) $$

The desired equations for the *inverse transformation* of the kinematics thus can be written as:

$$ \begin{aligned} P_x &= C_c(Q_x - L_x) - S_C(Q_y - L_y) \\ P_y &= S_C C_A(Q_x - L_x) + C_C C_A(Q_y - L_y) - S_A(Q_z - L_z) \\ P_z &= S_C S_A(Q_x - L_x) + C_C S_A(Q_y - L_y) + C_A(Q_z - L_z) \end{aligned} \qquad (17) $$

9.3.5.2. Transformations for a xyzac-trt machine with rotary axis offsets

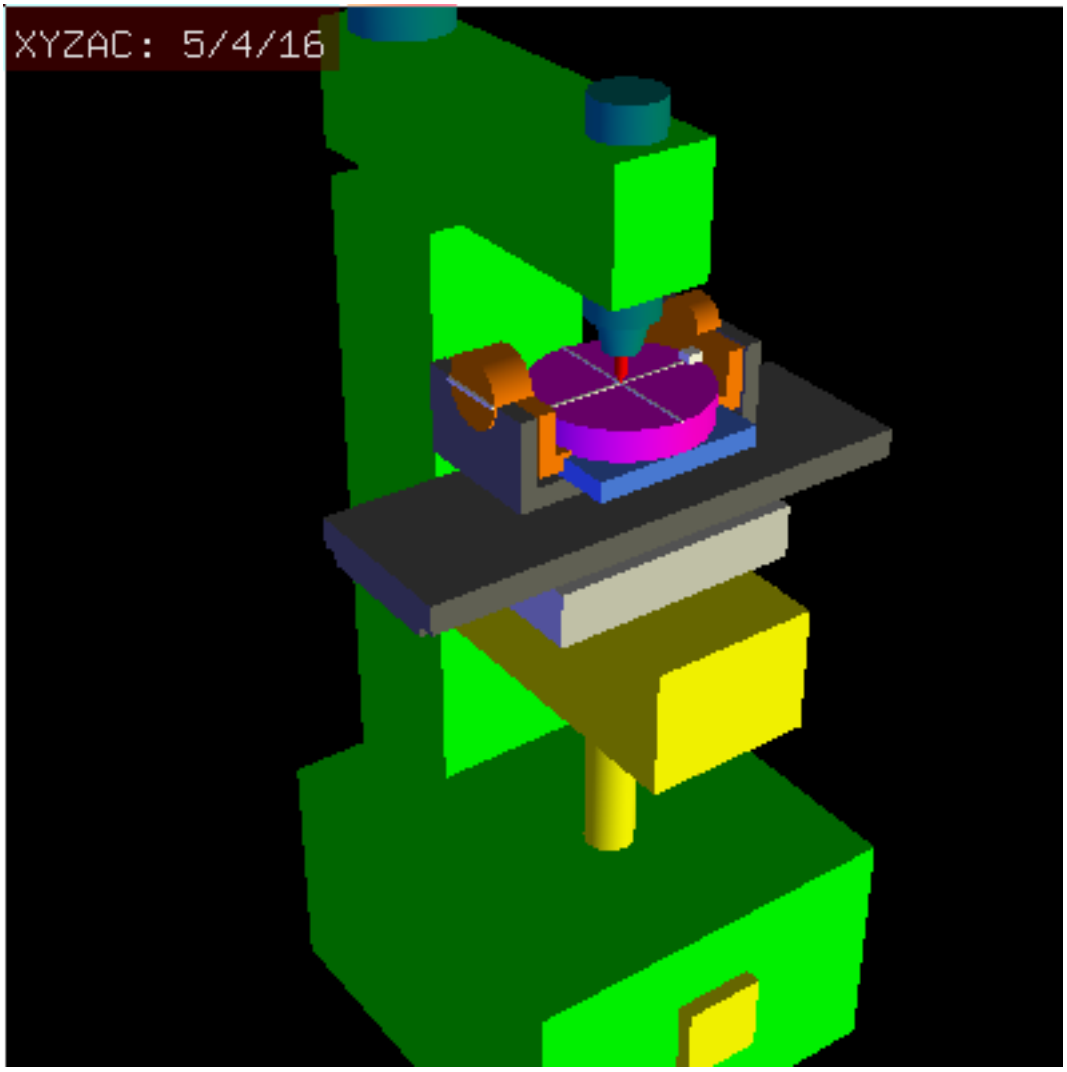


Figura 9.6: vismach model of xyzac-trt with rotational axis offsets (positive)

We deal here with a extended configuration in which the tilting axis and rotary axis do not intersect at a point but have an offset D_y. Furthermore, there is also an z-offset between the two coordinate systems O_{ws} and O_{wp} of Fig. 2, called D_z. A vismach model is shown in Fig. 5 and the offsets are shown in Fig. 6 (positive offsets in this example). To simplify the configuration, the offsets L_x, L_y, L_z of the previous case are not included. They are probably not necessary if one uses the G54 offsets in LinuxCNC by means of the ”touch of” facility.

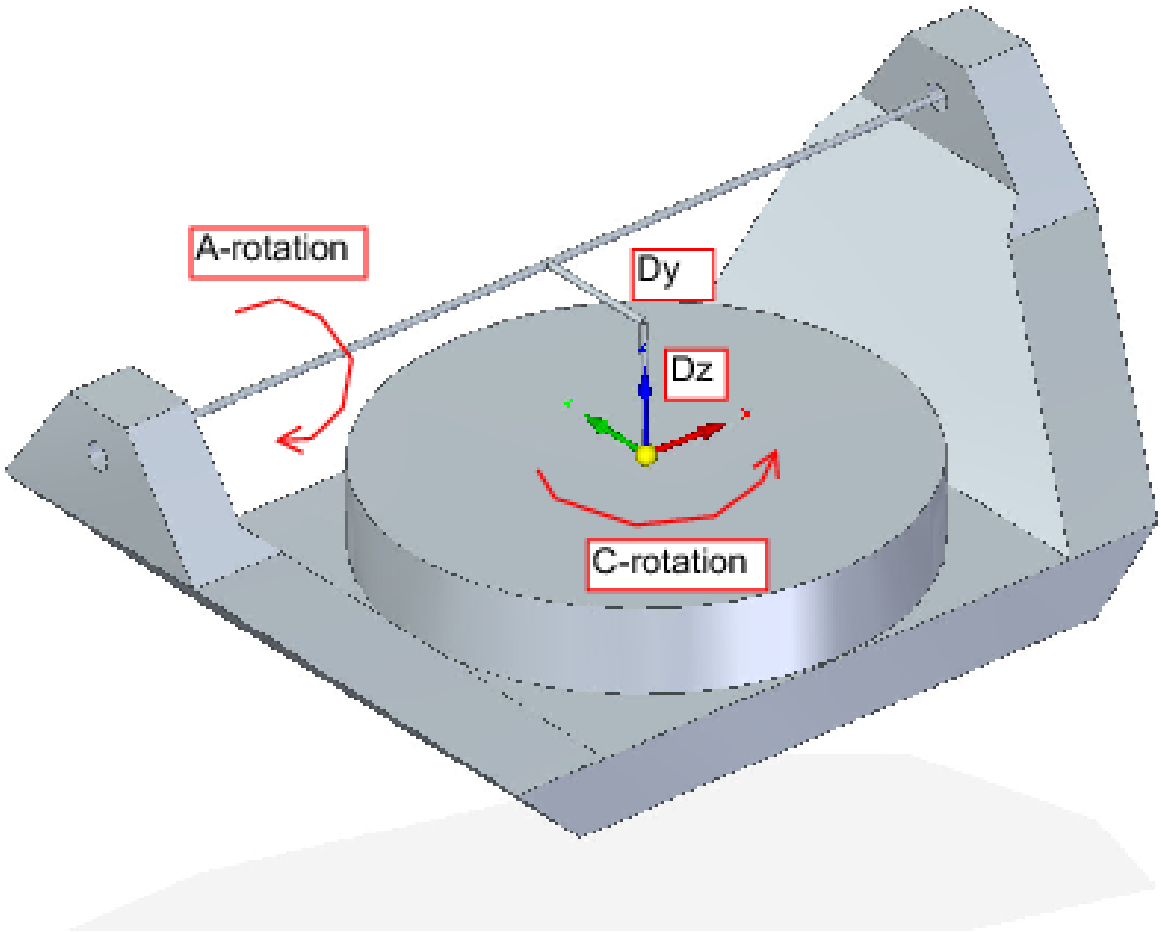


Figura 9.7: Table tilting/rotary xyzac-trt configuration, with axis offsets

The transformation can be defined by the sequential multiplication of the matrices:

$$ {}^{w}A_{t} = {}^{w}A_{O} \cdot {}^{O}A_{A} \cdot {}^{A}A_{P} \cdot {}^{P}A_{t} \qquad (18) $$

with the matrices built up as follows:

$$ {}^{w}A_{O} = \begin{bmatrix} C_C & S_C & 0 & 0 \\ -S_C & C_C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{O}A_{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & D_y \\ 0 & 0 & 1 & D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (19) $$

$$ {}^{A}A_{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_A & S_A & 0 \\ 0 & -S_A & C_A & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{P}A_{t} = \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y - D_y \\ 0 & 0 & 1 & P_z - D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (20) $$

In these equations D_y, D_z defines the offsets of the pivot point of the rotary axes A relative to the workpiece coordinate system origin. Furthermore, P_x, P_y, P_z are the relative distances of the pivot point to the cutter tip position, which can also be called the ”joint coordinates” of the pivot point. The pivot point is on the A rotary axis.

When multiplied in accordance with (18), we obtain:

$$ {}^{w}A_t = \begin{bmatrix} C_C & S_C C_A & S_C S_A & C_C P_x + S_C C_A (P_y - D_y) + S_C S_A (P_z - D_z) + S_C D_y \\ -S_C & C_C C_A & C_C S_A & -S_C P_x + C_C C_A (P_y - D_y) + C_C S_A (P_z - D_z) + C_C D_y \\ 0 & -S_A & C_A & -S_A (P_y - D_y) + C_A (P_z - D_z) + D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (21) $$

We can now equate the third column of this matrix with our given tool orientation vector K, ie.:

$$ K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} = \begin{bmatrix} S_C S_A \\ C_C S_A \\ C_A \\ 0 \end{bmatrix} \qquad (22) $$

From these equations we can solve for the rotation angles $theta_A$, $theta_C$. From the third row we find:

$$ \theta_A = cos^{-1}(K_z) \qquad (0 < \theta_A < \pi) \qquad (23) $$

and by dividing the second row by the first row we find:

$$ \theta_C = tan2^{-1}(K_x, K_y) \qquad (-\pi < \theta_C < \pi) \qquad (24) $$

These relationships are typically used in the CAM post-processor to convert the tool orientation vectors to rotation angles.

Equating the last column of (21) with the tool position vector Q, we can write:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C P_x + S_C C_A (P_y - D_y) + S_C S_A (P_z - D_z) + S_C D_y \\ -S_C P_x + C_C C_A (P_y - D_y) + C_C S_A (P_z - D_z) + C_C D_y \\ -S_A (P_y - D_y) + C_A (P_z - D_z) + D_z \\ 1 \end{bmatrix} \qquad (25) $$

El vector en el lado derecho también se puede escribir como el producto de una matriz y un vector, que da como resultado:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C & S_C C_A & S_C S_A & -S_c C_A D_y - S_C S_A D_z + S_C D_y \\ -S_C & C_C C_A & C_C S_A & -C_C C_A D_y - C_C S_A D_z + C_C D_y \\ 0 & -S_A & C_A & S_A D_y - C_A D_z + D_z \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = {}^{Q}A_P \cdot P \qquad (26) $$

which is the *forward transformation* of the kinematics.

We can solve for P from equation (25) as $P = ({}^{Q}A_P)^{-1} * Q$ using (15) as before. We thereby obtain:

$$ \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C & -S_C & 0 & 0 \\ S_C C_A & C_C C_A & -S_A & -C_A D_y + S_A D_z + D_y \\ S_C S_A & C_S S_A & C_A & -S_A D_y - C_A D_z + D_z \\ 0 & 0 & 1 & \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \qquad (27) $$

The desired equations for the *inverse transformation* of the kinematics thus can be written as:

$$ \begin{aligned} P_x &= C_c Q_x - S_C Q_y \\ P_y &= S_C C_A Q_x + C_C C_A Q_y - S_A Q_z - C_A D_y + S_A D_z + D_y \\ P_z &= S_C S_A Q_x + C_S S_A Q_y + C_A Q_z - S_A D_y - C_A D_z + D_z \end{aligned} \qquad (28) $$

9.3.5.3. Transformations for a xyzbc-trt machine with rotary axis offsets

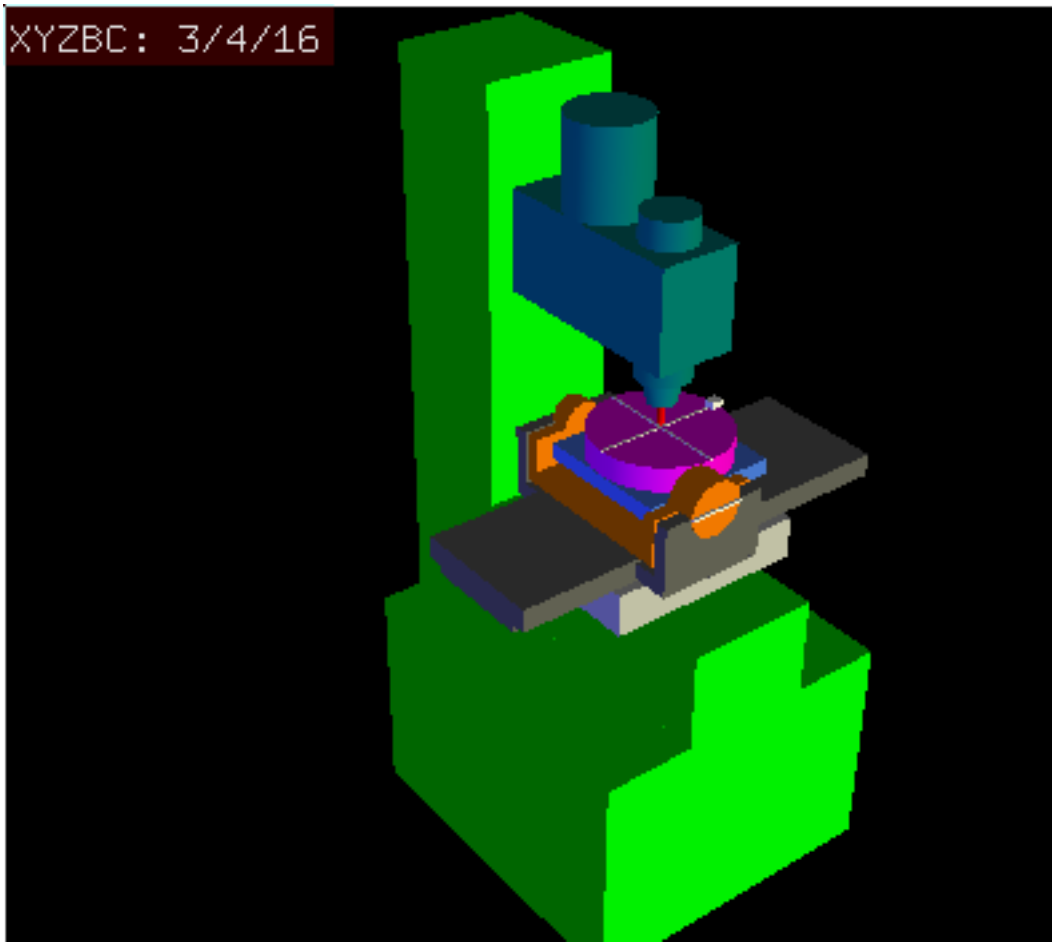


Figura 9.8: vismach model of xyzbc-trt with rotational axis offsets (negative)

We deal here again with a extended configuration in which the tilting axis (about the y-axis) and rotary axis do not intersect at a point but have an offset D_x. Furthermore, there is also an z-offset between the two coordinate systems O_{ws} and O_{wp} of Fig. 2, called D_z. A vismach model is shown in Fig. 7 (negative offsets in this example) and the positive offsets are shown in Fig. 8.

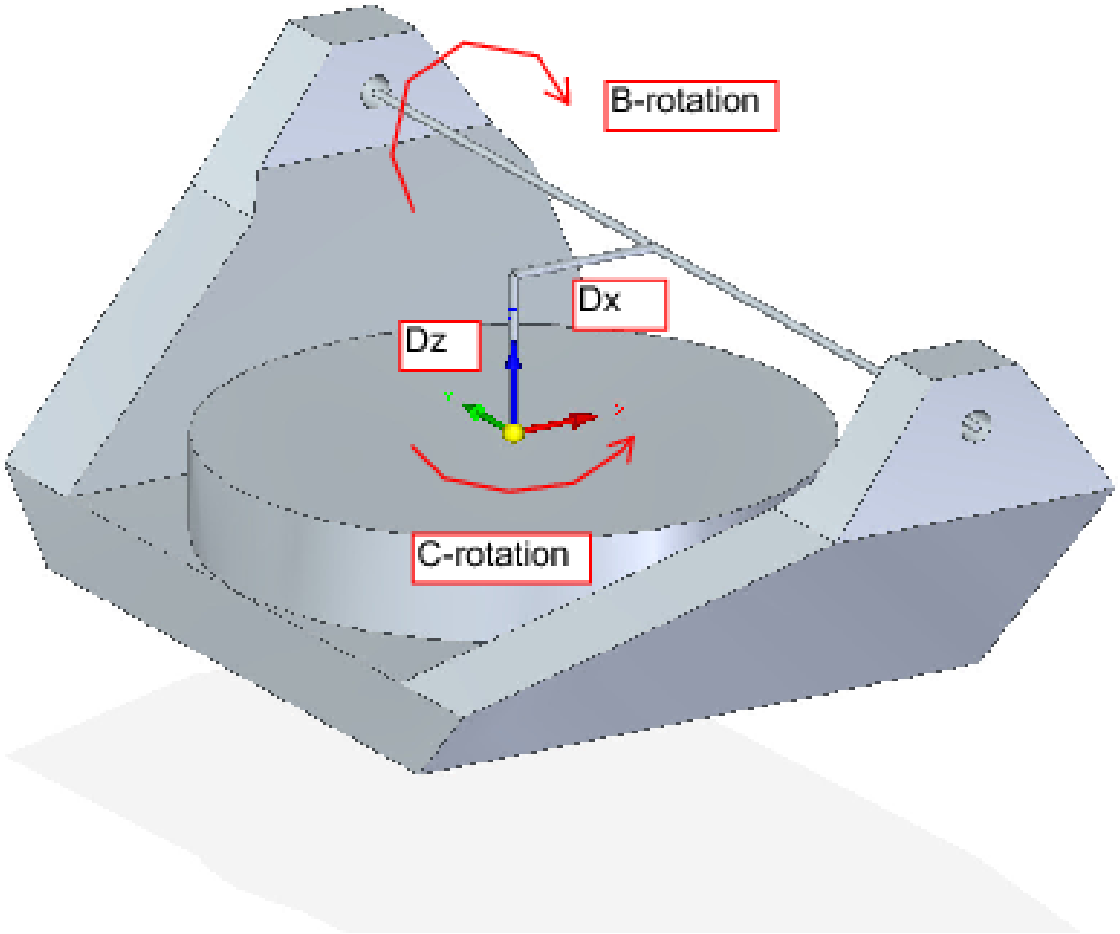


Figura 9.9: Table tilting/rotary xyzbc-trt configuration, with axis offsets

The transformation can be defined by the sequential multiplication of the matrices:

$$ {}^{w}A_t = {}^{w}A_O \cdot {}^{O}A_B \cdot {}^{B}A_P \cdot {}^{P}A_t \qquad (29) $$

with the matrices built up as follows:

$$ {}^{w}A_O = \begin{bmatrix} C_C & S_C & 0 & 0 \\ -S_C & C_C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{O}A_B = \begin{bmatrix} 1 & 0 & 0 & D_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (30) $$

$$ {}^{B}A_P = \begin{bmatrix} C_B & 0 & -S_B & 0 \\ 0 & 1 & 0 & 0 \\ S_B & 0 & C_B & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{P}A_t = \begin{bmatrix} 1 & 0 & 0 & P_x - D_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z - D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (31) $$

In these equations D_x, D_z defines the offsets of the pivot point of the rotary axes B relative to the workpiece coordinate system origin. Furthermore, P_x, P_y, P_z are the relative distances of the pivot point to the cutter tip position, which can also be called the ”joint coordinates” of the pivot point. The pivot point is on the B rotary axis.

When multiplied in accordance with (29), we obtain:

$$ {}^{w}A_t = \begin{bmatrix} C_C C_B & S_C & -C_C S_B & C_C C_B(P_x - D_x) + S_C P_y - C_C S_B(P_z - D_z) + C_C D_x \\ -S_C C_B & C_C & S_C S_B & -S_C C_B(P_x - D_X) + C_C P_y + S_C S_B(P_z - D_z) - S_C D_x \\ S_B & 0 & C_B & S_B(P_x - D_x) + C_B(P_z - D_z) + D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (32) $$

We can now equate the third column of this matrix with our given tool orientation vector K, i.e.:

$$ K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} = \begin{bmatrix} -C_C S_B \\ S_C S_B \\ C_B \\ 0 \end{bmatrix} \qquad (33) $$

From these equations we can solve for the rotation angles theta$_B$, theta$_C$. From the third row we find:

$$ \theta_B = cos^{-1}(K_z) \qquad (0 < \theta_B < \pi) \qquad (34) $$

and by dividing the second row by the first row we find:

$$ \theta_C = tan2^{-1}(K_y, K_x) \qquad (-\pi < \theta_C < \pi) \qquad (35) $$

These relationships are typically used in the CAM post-processor to convert the tool orientation vectors to rotation angles.

Equating the last column of (32) with the tool position vector Q, we can write:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C C_B(P_x - D_x) + S_C P_y - C_C S_B(P_z - D_z) + C_C D_x \\ -S_C C_B(P_x - D_X) + C_C P_y + S_C S_B(P_z - D_z) - S_C D_x \\ S_B(P_x - D_x) + C_B(P_z - D_z) + D_z \\ 1 \end{bmatrix} \qquad (36) $$

El vector en el lado derecho también se puede escribir como el producto de una matriz y un vector, que da como resultado:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C C_B & S_C & -C_C S_B & -C_c C_B D_x + C_C S_B D_z + C_C D_x \\ -S_C C_B & C_C & S_C S_B & S_C C_B D_x - S_C S_B D_z - S_C D_x \\ S_B & 0 & C_B & -S_B D_x - C_B D_z + D_z \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = {}^{Q}A_P \cdot P \qquad (37) $$

which is the *forward transformation* of the kinematics.

Podemos resolver para P de la ecuación (37) como $P=({}^{Q}A_P)^{-1}*Q$.

With the same approach as before, we obtain:

$$ \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C C_B & -S_C C_B & S_B & -C_B D_x - S_B D_z + D_x \\ S_C & C_C & 0 & 0 \\ -C_C S_B & S_C S_B & C_B & S_B D_x - C_B D_z + D_z \\ 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \qquad (38) $$

The desired equations for the *inverse transformation* of the kinematics thus can be written as:

$$ \begin{aligned} P_x &= C_c C_B Q_x - S_C C_B Q_y + S_B Q_z - C_B D_x - S_B D_z + D_x \\ P_y &= S_C Q_x + C_C Q_y \\ P_z &= -C_C S_B Q_x + S_C S_B Q_y + C_B Q_z + S_B D_x - C_B D_z + D_z \end{aligned} \qquad (39) $$

9.3.6. Table Rotary/Tilting Examples

LinuxCNC includes kinematics modules for the *xyzac-trt* and *xyzbc-trt* topologies described in the mathematics detailed above. For interested users, the source code is available in the git tree in the *src/emc/kinematics/* directory.

Example xyzac-trt and xyzbc-trt simulation configurations are located in the Sample Configurations (*configs/sim/axis/vismach/5axis/table-rotary-tilting/*) directory.

The example configurations include the required INI files and an examples subdirectory with G-code (NGC) files. These sim configurations invoke a realistic 3-dimensional model using the LinuxCNC vismach facility.

9.3.6.1. Vismach Simulation Models

Vismach is a library of python routines to display a dynamic simulation of a CNC machine on the PC screen. The python script for a particular machine is loaded in HAL and data passed by HAL pin connections. The non-realtime vismach model is loaded by a HAL command like:

```
loadusr -W xyzac-trt-gui
```

and connections are made using HAL commands like:

```
net  :table-x   joint.0.pos-fb xyzac-trt-gui.table-x
net  :saddle-y  joint.1.pos-fb xyzac-trt-gui.saddle-y
...
```

See the simulation INI files for details of the HAL connections used for the vismach model.

9.3.6.2. Tool-Length Compensation

In order to use tools from a tool table sequentially with tool-length compensation applied automatically, a further Z-offset is required. For a tool that is longer than the ”master” tool, which typically has a tool length of zero, LinuxCNC has a variable called ”motion.tooloffset.z”. If this variable is passed on to the kinematic component (and vismach python script), then the necessary additional Z-offset for a new tool can be accounted for by adding the component statement, for example:

$$D_z = D_z + \text{tool-offset}$$

The required HAL connection (for xyzac-trt) is:

```
net :tool-offset motion.tooloffset.z xyzac-trt-kins.tool-offset
```

where:

```
:tool-offset ---------------- signal name
motion.tooloffset.z --------- output HAL pin from LinuxCNC motion module
xyzac-trt-kins.tool-offset -- input  HAL pin to xyzac-trt-kins
```

9.3.7. Custom Kinematics Components

LinuxCNC implements kinematics using a HAL component that is loaded at startup of LinuxCNC. The most common kinematics module, *trivkins*, implements identity (trivial) kinematics where there is a one-to-one correspondence between an axis coordinate letter and a motor joint. Additional kinematics modules for more complex systems (including *xyzac-trt* and *xyzbc-trt* described above) are available.

See the kins manpage (**\$ man kins**) for brief descriptions of the available kinematics modules.

The kinematics modules provided by LinuxCNC are typically written in the C-language. Since a standard structure is used, creation of a custom kinematics module is facilitated by copying an existing source file to a user file with a new name, modifying it, and then installing.

Installation is done using halcompile:

```
sudo halcompile --install kinsname.c
```

where ”kinsname” is the name you give to your component. The sudo prefix is required to install it and you will be asked for your root password. See the halcompile man page for more information (**\$ man halcompile**)

Una vez compilado e instalado, puede hacer referencia a él en la configuración de su máquina. Esto se hace en el archivo INI de su directorio config. Por ejemplo, la especificación INI común:

```
[KINS]
KINEMATICS = trivkins
```

is replaced by

```
[KINS]
KINEMATICS = kinsname
```

where ”kinsname” is the name of your kins program. Additional HAL pins may be created by the module for variable configuration items such as the D_x, D_y, D_z, tool-offset used in the xyzac-trt kinematics module. These pins can be connected to a signal for dynamic control or set once with HAL connections like:

```
# set offset parameters
net :tool-offset motion.tooloffset.z xyzac-trt-kins.tool-offset
setp xyzac-trt-kins.y-offset 0
setp xyzac-trt-kins.z-offset 20
```

9.3.8. Figures

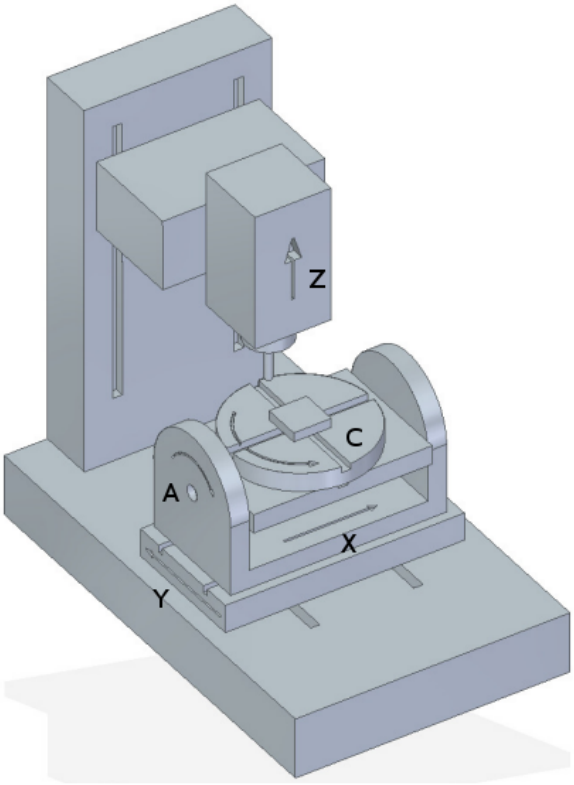


Figura 9.10: Table tilting/rotating configuration

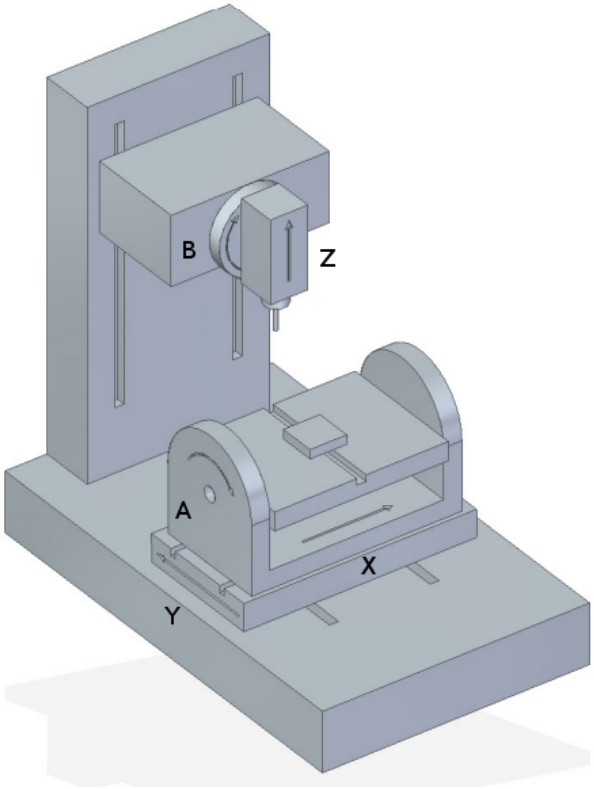


Figura 9.11: Spindle/table tilting configuration

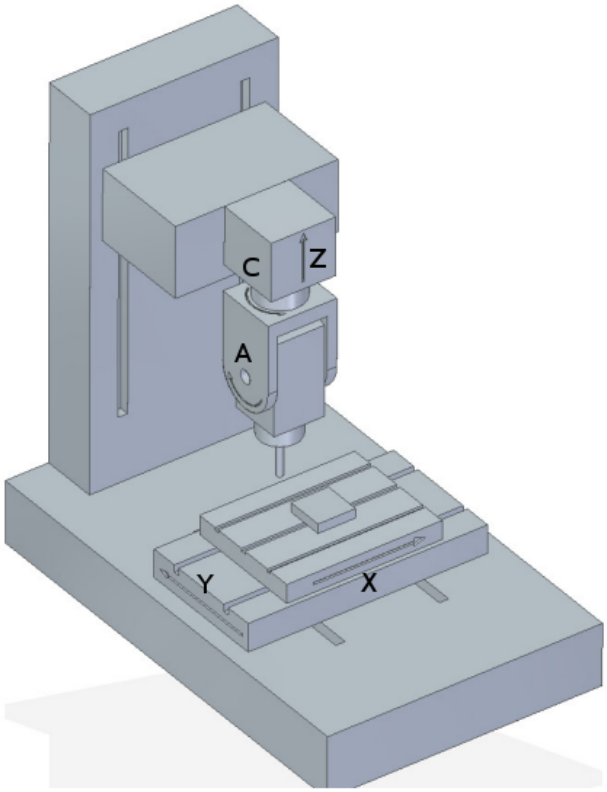


Figura 9.12: Spindle tilting/rotary configuration

9.3.9. REFERENCES

1. AXIS MACHINE TOOLS: Kinematics and Vismach Implementation in LinuxCNC, RJ du Preez, SA-CNC-CLUB, April 7, 2016.
2. A Postprocessor Based on the Kinematics Model for General Five-Axis machine Tools: C-H She, R-S Lee, J Manufacturing Processes, V2 N2, 2000.
3. NC Post-processor for 5-axis milling of table-rotating/tilting type: YH Jung, DW Lee, JS Kim, HS Mok, J Materials Processing Technology,130-131 (2002) 641-646.
4. 3D 6-DOF Serial Arm Robot Kinematics, RJ du Preez, SA-CNC-CLUB, Dec. 5, 2013.
5. Design of a generic five-axis postprocessor based on generalized kinematics model of machine tool: C-H She, C-C Chang, Int. J Machine Tools & Manufacture, 47 (2007) 537-545.

9.4. Switchable Kinematics (switchkins)

9.4.1. Introducción

A number of kinematics modules support the switching of kinematics calculations. These modules support a default kinematics method (type0), a second built-in method (type1), and (optionally) a user-provided kinematics method (type2). Identity kinematics are typically used for the type1 method.

The switchkins functionality can be used for machines where post-homing joint control is needed during setup or to avoid movement near singularities from G-code. Such machines use specific kinematics calculations for most operations but can be switched to identity kinematics for control of individual joints after homing.

The kinematics type is selected by a motion module HAL pin that can be updated from a G-code program or by interactive MDI commands. The halui provisions for activating MDI commands can be used to allow buttons to select the kinematics type using hardware controls or a virtual panel (PyVCP, GladeVCP, etc.).

When a kinematics type is changed, the G-code must also issue commands to **force synchronization** of the interpreter and motion parts of LinuxCNC. Typically, a HAL pin *read* command (M66 E0 L0) is used immediately after altering the controlling HAL pin to force synchronization.

9.4.2. Switchable Kinematic Modules

The following kinematics modules support switchable kinematics:

1. **xyzac-trt-kins** (type0:xyzac-trt-kins type1:identity)
2. **xyzbc-trt-kins** (type0:xyzbc-trt-kins type1:identity)
3. **genhexkins** (type0:genhexkins type1:identity)
4. **genserkins** (type0:genserkins type1:identity) (puma560 example)
5. **pumakins** (type0:pumakins type1:identity)
6. **scarakins** (type0:scarakins type1:identity)
7. **5axiskins** (type0:5axiskins type1:identity) (bridgemill)

The xyz[ab]c-trt-kins modules by default use type0==xyz[ab]c-trt-kins for backwards compatibility. The provided sim configs alter the type0/type1 convention by forcing type0==identity kinematics using the module string parameter *sparm* with an INI file setting like:

```
[KINS]
KINEMATICS = xyzac-trt-kins sparm=identityfirst
...
```

9.4.2.1. Identity letter assignments

When using an **identity** kinematics type, the module parameter *coordinates* can be used to assign letters to joints in arbitrary order from the set of allowed coordinate letters. Examples:

```
[KINS]
JOINTS = 6

# conventional identity ordering: joint0==x, joint1==y, ...
KINEMATICS = genhexkins coordinates=xyzabc

# custom identity ordering: joint0==c, joint1==b, ...
# KINEMATICS = genhexkins coordinates=cbazyx
```

---

nota
If the coordinates= parameter is omitted, the default joint-letter identity assignments are joint0==x,joint1=y,...

The joint assignments provided for **identity** kinematics when using the coordinates parameter are identical to those provided for the trivkins module. However, duplication of axis letters to assign multiple joints for a coordinate letter is not generally applicable for serial or parallel kinematics (like genserkins, pumakins, genhexkins, etc.) where there is no simple relationship between joints and coordinates.

Duplication of axis coordinate letters is supported in the kinematics modules xyzac-trt-kins, xyzbc-trt-kins, and 5axiskins (bridgemill). Typical applications for duplicate coordinates are gantry machines where two motors (joints) are used for the transverse axis.

9.4.2.2. Backwards compatibility

Switchable kinematics initialize with motion.switchkins-type==0 implementing their eponymous kinematics method. If the the motion.switchkins-type pin is not connected — as in legacy configurations — only the default kinematics type is available.

9.4.3. HAL Pins

Kinematics switching is controlled by the motion module input HAL pin **motion.switchkins-type**. The floating point pin value is truncated to integer and used to select one of the provided kinematics types. The zero startup value selects the type0 default kinematics type.

---

nota
The motion.switchkins-type input pin is floating point in order to facilitate connections to motion module output pins like motion.analog-out-0n that are controllable by standard M-codes (typically M68EnL0).

Output HAL pins are provided to inform GUIs of the current kinematics type. These pins can also be connected to digital inputs that are read by G-code programs to enable or disable program behavior in accordance with the active kinematics type.

9.4.3.1. HAL Pin Summary

1. **motion.switchkins-type** Input (float)
2. **kinstype.is-0** Output (bit)
3. **kinstype.is-1** Output (bit)
4. **kinstype.is-2** Output (bit)

9.4.4. Utilización

9.4.4.1. HAL Connections

Switchkins functionality is enabled by the pin **motion.switchkins-type**. Typically, this pin is sourced by an analog output pin like motion.analog-out-03 so that it can be set by M68 commands. Example:

```
net :kinstype-select <= motion.analog-out-03
net :kinstype-select => motion.switchkins-type
```

9.4.4.2. G-/M-code commands

Kinstype selection is managed using G-code sequences like:

```
...
M68 E3 Q1 ;update analog-out-03 to select kinstype 1
M66 E0 L0 ;sync interp-motion
...
...       ;user G-code
...
M68 E3 Q0 ;update analog-out-03 to select kinstype 0
M66 E0 L0 ;sync interp-motion
...
```

nota
An M66 *wait-on-input* command updates the #5399 variable. If the current value of this variable is needed for subsequent purposes, it should be copied to an additional variable before invoking M66.

These G-code command sequences are typically implemented in G-code subroutines as remapped M-codes or with conventional M-code scripts.

Suggested codes (as used in sim configs) are:

Códigos M de usuario convencionales:

1. M128 Select kinstype 0 (startup default kinematics)
2. M129 Select kinstype 1 (typically identity kinematics)
3. M130 Select kinstype 2 (user-provided kinematics)

Códigos M remapeados:

1. M428 Select kinstype 0 (startup default kinematics)
2. M429 Select kinstype 1 (typically identity kinematics)
3. M430 Select kinstype 2 (user-provided kinematics)

nota
Conventional user M-codes (in the range M100-M199) are in modal group 10. Remapped M-codes (in the range M200 to M999) can specify a modalgroup. See the remap documentation for additional information.

9.4.4.3. INI file limit settings

LinuxCNC trajectory planning uses limits for position (min,max), velocity, and acceleration for each applicable coordinate letter specified in the configuration INI file. Example for letter L (in the set *XYZABCUVW*):

```
[AXIS_L]
MIN_LIMIT =
MAX_LIMIT =
MAX_VELOCITY =
MIN_ACCELERATION =
```

The INI file limits specified apply to the type 0 default kinematics type that is activated at startup. These limits may **not** be applicable when switching to alternative kinematics. However, since an interpreter-motion synchronization is required when switching kinematics, INI-HAL pins can be used to setup limits for a pending kinematics type.

---

nota
INI-HAL pins are typically not recognized during a G-code program unless a synchronization (queue-buster) command is issued. See the milltask manpage for more information ($ man milltask).

The relevant INI-HAL pins for a joint number (*N*) are:

```
ini.N.min_limit
ini.N.max_limit
ini.N.max_acceleration
ini.N.max_velocity
```

The relevant INI-HAL pins for an axis coordinate (*L*) are:

```
ini.L.min_limit
ini.L.max_limit
ini.L.max_velocity
ini.L.max_acceleration
```

---

nota
In general, there are no fixed mappings between joint numbers and axis coordinate letters. There may be specific mappings for some kinematics modules especially those that implement identity kinematics (trivkins). See the kins man page for more information ($ man kins).

A user-provided M-code can alter any or all of the axis coordinate limits prior to changing the motion.switchki type pin and synchronizing the interpreter and motionparts of LinuxCNC. As an example, a bash script invoking halcmd can be *hardcoded* to set any number of HAL pins:

```
#!/bin/bash
halcmd -f <<EOF
setp ini.x.min_limit -100
setp ini.x.max_limit  100
# ... repeat for other limit parameters
EOF
```

Scripts like this can be invoked as a user M-code and used **prior** to the kins switching M-code that updates the `motion.switchkins-type` HAL pin and forces an interp-motion sync. Typically, separate scripts would be used for each kinstype (0,1,2).

When identity kinematics are provided as a means to control individual joints, it may be convenient to set or restore limits as specified in the system INI file. For example, a robot starts with a complex (non-identity) kinematics (type0) after homing. The system is configured so that it can be switched to identity kinematics (type1) in order to manipulate individual joints using the conventional letters from the set *XYZABCUVW*. The INI file settings ([AXIS_L]) are **not** applicable when operating with identity (type1) kinematics. To address this use case, the user M-code scripts can be designed as follows:

M129 (Switch to identity type1)

1. read and parse INI file
2. HAL: setp the INI-HAL limit pins for each axis letter ([AXIS_L]) according to the *identity-referenced* joint number INI file setting ([JOINT_N])

3. HAL: `setp motion.switchkins-type 1`
4. MDI: execute a syncing G-code (M66E0L0)

M128 (restore robot default kinematics type 0)

1. read and parse INI file
2. HAL: setp the INI-HAL limit pins for each axis letter ([AXIS_L]) according to the appropriate INI file setting ([AXIS_L])
3. HAL: `setp motion.switchkins-type 0`
4. MDI: execute a syncing G-code (M66E0L0)

---

nota
The vismach simulation configurations for a puma robot demonstrate M-code scripts (M128,M129,M130) for this example use case.

9.4.4.4. Consideraciones de offsets del sistema de coordenadas

Like INI file limit settings, coordinate system offsets (G92, G10L2, G10L20, G43, etc) are generally applicable only for the type 0 default startup kinematics type. When switching kinematics types, it may be **important** to either reset all offsets prior to switching or update offsets based on system-specific requirements.

9.4.4.5. External offset considerations

External offsets (set to an axis (*L*) via axis.*L*.eoffset-request) are preserved during kinematics switches. When an offset is active on an axis before the switch (visible in axis.*L*.eoffset), the trajectory planner maintains that same offset after the switch, similar to how it maintains the commanded position from a G-code. This ensures consistent machine behavior regardless of the active kinematics.

If maintaining the offset will be an issue due to axis limit changes or other concerns, be sure to clear and possibly disable the eoffset before making a kinematics switch.

9.4.5. Simulation configs

Simulation configs (requiring no hardware) are provided with illustrative vismach displays in subdirectories of configs/sim/axis/vismach/ .

1. 5axis/table-rotary-tilting/xyzac-trt.ini (xyzac-trt-kins)
2. 5axis/table-rotary-tilting/xyzbc-trt.ini (xyzac-trt-kins)
3. 5axis/bridgemill/5axis.ini (5axiskins)
4. scara/scara.ini (scarakins)
5. puma/puma560.ini (genserkins)
6. puma/puma.ini (pumakins)
7. hexapod-sim/hexapod.ini (genhexkins)

9.4.6. User kinematics provisions

Custom kinematics can be coded and tested on Run-In-Place (*RIP*) builds. A template file src/emc/kinematics/userkfuncs.c is provided in the distribution. This file can be copied/renamed to a user directory and edited to supply custom kinematics with kinstype==2.

The user custom kinematics file can be compiled from out-of-tree source locations for rt-preempt implementations or by replacing the in-tree template file (src/emc/kinematics/userkfuncs.c) for rtai systems.

Preempt-rt make example:

```
$ userkfuncs=/home/myname/kins/mykins.c make && sudo make setuid
```

9.4.7. Warnings

Unexpected behavior can result if a G-code program is inadvertently started with an incompatible kinematics type. Unwanted behavior can be circumvented in G-code programs by:

1. Connecting appropriate kinstype.is.N HAL pins to digital input pins (like motion.digital-in-0m).
2. Reading the digital input pin (M66 E0 Pm) at the start of the G-code program
3. Aborting (M2) the G-code program with a message (DEBUG, problem_message) if the kinstype is not suitable.

When using jogging facilities or MDI commands interactively, operator caution is required. Guis should include indicators to display the current kinematics type.

---

nota
Switching kinematics can cause substantial operational changes requiring careful design, testing, and training for deployment. The management of coordinate offsets, tool compensation, and INI file limits may require complicated and non-standard operating protocols.

9.4.8. Notas sobre el código

Kinematic modules providing switchkins functionality are linked to the switchkins.o object (switchkins.c) that provides the module *main* program (rtapi_app_main()) and related functions. This *main* program reads (optional) module command-line parameters (coordinates, sparm) and passes them to the module-provided function switchkinsSetup().

The switchkinsSetup() function identifies kinstype-specific setup routines and the functions for forward an inverse calculation for each kinstype (0,1,2) and sets a number of configuration settings.

After calling switchkinsSetup(), rtapi_app_main() checks the supplied parameters, creates a HAL component, and then invokes the setup routine identified for each kinstype (0,1,2).

Each kinstype (0,1,2) setup routine can (optionally) create HAL pins and set them to default values. When all setup routines finish, rtapi_app_main() issues hal_ready() for the component to complete creation of the module.

9.5. PID Tuning

9.5.1. PID Controller

Un controlador PID (Controlador Proporcional-Integral-Derivativo) es un mecanismo de control por realimentación ampliamente usado en sistemas de control industrial. [6]

The Controller compares a measured value from a process (typically an industrial process) with a reference set point value. The difference (or *error* signal) is then used to calculate a new value for a manipulable input to the process that brings the process measured value back to its desired set point.

Unlike simpler control algorithms, the PID controller can adjust process outputs based on the history and rate of change of the error signal, which gives more accurate and stable control. (It can be shown mathematically that a PID loop will produce accurate, stable control in cases where a simple proportional control would either have a steady-state error or would cause the process to oscillate).

9.5.1.1. Control loop basics

Intuitively, the PID loop tries to automate what an intelligent operator with a gauge and a control knob would do. The operator would read a gauge showing the output measurement of a process, and use the knob to adjust the input of the process (the *action*) until the process's output measurement stabilizes at the desired value on the gauge.

En la literatura de control más antigua, este proceso de ajuste se denomina acción *reset*. La posición de la aguja en el medidor es una *medida*, *valor de proceso* o *variable de proceso*. El valor deseado en el medidor se llama *punto de ajuste* (también llamado *valor de ajuste*). La diferencia entre la aguja del indicador y el punto de ajuste es el *error*.

A control loop consists of three parts:

1. Measurement by a sensor connected to the process (e.g. encoder),
2. Decision in a controller element,
3. Action through an output device such as an motor.

As the controller reads a sensor, it subtracts this measurement from the *set point* to determine the *error*. It then uses the error to calculate a correction to the process's input variable (the *action*) so that this correction will remove the error from the process's output measurement.

In a PID loop, correction is calculated from the error in three ways: cancel out the current error directly (Proportional), the amount of time the error has continued uncorrected (Integral), and anticipate the future error from the rate of change of the error over time (Derivative).

A PID controller can be used to control any measurable variable which can be affected by manipulating some other process variable. For example, it can be used to control temperature, pressure, flow rate, chemical composition, speed, or other variables. Automobile cruise control is an example of a process outside of industry which utilizes crude PID control.

Some control systems arrange PID controllers in cascades or networks. That is, a *master* control produces signals used by *slave* controllers. One common situation is motor controls: one often wants the motor to have a controlled speed, with the *slave* controller (often built into a variable frequency drive) directly managing the speed based on a proportional input. This *slave* input is fed by the *master* controller's output, which is controlling based upon a related variable.

[6] Esta subsección está tomada de un articulo mucho más extenso encontrado en https://en.wikipedia.org/wiki/PID_controller

9.5.1.2. Theory

PID is named after its three correcting calculations, which all add to and adjust the controlled quantity. These additions are actually *subtractions* of error, because the proportions are usually negative:

Proportional To handle the present, the error is multiplied by a (negative) constant P (for *proportional*), and added to (subtracting error from) the controlled quantity. P is only valid in the band over which a controller's output is proportional to the error of the system. Note that when the error is zero, a proportional controller's output is zero.

Integral Para aprender del pasado, el error está integrado (acumulado) sobre un período de tiempo, y luego multiplicado por una constante I (negativa, haciendo un promedio), y añadido a (restando el error de) la cantidad controlada. I promedia el error medido para encontrar el error promedio del proceso desde el punto de ajuste. Un sistema proporcional simple oscila, avanzando y retrocediendo alrededor del punto de ajuste porque no hay nada para eliminar el error cuando se sobrepasa, u oscila y/o se estabiliza a un valor demasiado bajo o demasiado alto. Al agregar una proporción negativa (es decir, restar parte de) del error promedio de la entrada de proceso, la diferencia promedio entre la salida del proceso y el punto de ajuste siempre se está reduciendo. Por lo tanto, eventualmente, una salida del proceso del bucle PID bien ajustado se establecerá en el punto de ajuste.

Derivative To handle the future, the first derivative (the slope of the error) over time is calculated, and multiplied by another (negative) constant D, and also added to (subtracting error from) the controlled quantity. The derivative term controls the response to a change in the system. The larger the derivative term, the more rapidly the controller responds to changes in the process's output.

More technically, a PID loop can be characterized as a filter applied to a complex frequency-domain system. This is useful in order to calculate whether it will actually reach a stable value. If the values are chosen incorrectly, the controlled process input can oscillate, and the process output may never stay at the set point.

9.5.1.3. Loop Tuning

Tuning a control loop is the adjustment of its control parameters (gain/proportional band, integral gain/reset, derivative gain/rate) to the optimum values for the desired control response. The optimum behavior on a process change or set point change varies depending on the application. Some processes must not allow an overshoot of the process variable from the set point. Other processes must minimize the energy expended in reaching a new set point. Generally stability of response is required and the process must not oscillate for any combination of process conditions and set points.

El ajuste de los bucles se hace más complicado por el tiempo de respuesta del proceso; puede tomar minutos u horas que un cambio de punto de ajuste produzca un efecto estable. Algunos procesos tienen un grado de no linealidad y, por lo tanto, parámetros que funcionan bien en condiciones de carga completa no funcionan cuando el proceso se inicia sin carga. Esta sección describe algunos métodos manuales tradicionales para sintonización de bucle.

There are several methods for tuning a PID loop. The choice of method will depend largely on whether or not the loop can be taken *offline* for tuning, and the response speed of the system. If the system can be taken offline, the best tuning method often involves subjecting the system to a step change in input, measuring the output as a function of time, and using this response to determine the control parameters.

Simple method If the system must remain on line, one tuning method is to first set the I and D values to zero. Increase the P until the output of the loop oscillates. Then increase I until oscillation stops. Finally, increase D until the loop is acceptably quick to reach its reference. A fast PID loop tuning usually overshoots slightly to reach the set point more quickly; however, some systems cannot accept overshoot.

Parámetro	Rise Time	Overshoot	Settling Time	Steady State Error
P	Decrease	Increase	Small Change	Decrease
I	Decrease	Increase	Increase	Eliminate
D	Small Change	Decrease	Decrease	Small Change

Effects of increasing parameters

Ziegler-Nichols method Another tuning method is formally known as the *Ziegler-Nichols method*, introduced by John G. Ziegler and Nathaniel B. Nichols in 1942 [7]. It starts in the same way as the method described before: first set the I and D gains to zero and then increase the P gain and expose the loop to external interference for example knocking the motor axis to cause it to move out of equilibrium in order to determine critical gain and period of oscillation until the output of the loop starts to oscillate. Write down the critical gain (K_c) and the oscillation period of the output (P_c). Then adjust the P, I and D controls as the table shows:

Control type	P	I	D
P	$.5K_c$		
PI	$.45K_c$	$P_c/1.2$	
PID	$.6K_c$	$P_c/2$	$P_c/8$

Final Steps After tuning the axis check the following error with Halscope to make sure it is within your machine requirements. More information on Halscope is in the HAL User manual.

9.5.1.4. Automatic PID tuning

Since LinuxCNC version 2.9, the pid component support automatic tuning using the Relay method [8]. This is a replacement for the now removed and obsolete at_pid component.

The pid component uses several constants to calculate the output value based on current and wanted state, the most important among them being Pgain', *Igain*, *Dgain*, *bias*, *FF0*, *FF1*, *FF2* and *FF3*. All of these need to have a sensible value for the controller to behave properly.

The current implementation of automatic tuning implement two different algorithms, selected using the tune-type pin. When tune-type is zero, it affects *Pgain*, *Igain* and *Dgain* while setting *FF0*, *FF1* and *FF2* to zero. If tune-type is 1, it affects *Pgain*, *Igain* and *FF1* while setting *Dgain*, *FF0* and *FF2* to zero. Note type 1 require scaling be set so output is in user units per second.

When autotuning a motor with tune-type 0, the algorithm will produce a square wave pattern centered around the *bias* value on the output pin of the PID controller, moving from the positive extreme to the negative extreme of the output range. This can be seen using the HAL Scope provided by LinuxCNC. For a motor controller taking +-10 V as its control signal, this might accelerate the motor full speed in one direction for a short period before telling it to go full speed in the opposite direction. Make sure to have a lot of room on either side of the starting position, and start with a low `tune-effort` value to limit the speed used. The `tune-effort` value define the extreme `output` value used, so if `tune-effort` is 1, the `output` value during tuning will move from 1 to -1. In other words, the extremes of the wave pattern is controlled by the *tune-effort* pin. Using too high *tune-effort* might overload the motor driver.

The number of cycles in the tune pattern is controlled by the *tune-cycles* pin. Of course, trying to change the direction of a physical object instantly (as in going directly from a positive voltage to the equivalent negative voltage in the motor controller case) do not change velocity instantly, and it take some time for the object to slow down and move in the opposite direction. This result in a more smooth wave form on the position pin, as the axis in question were vibrating back and forth. When the

[7] Ziegler, J. G. and Nichols, N. B. (1942), *Optimum Settings for Automatic Controllers*, Transactions of the ASME, DOI 10.1115/1.2899060 and The Internet Archive.

[8] Åström, Karl Johan and Hägglund, Tore (1984), *Automation paper Automatic Tuning of Simple Regulators with Specifications on Phase and Amplitude Margins*, DOI 10.1016/0005-1098(84)90014-1

axis reached the target speed in the opposing direction, the auto tuner change direction again. After several of these changes, the average time delay between the ”peaks” and ”valleys” of this movement graph is used to calculate proposed values for `Pgain`, `Igain` and `Dgain`, and insert them into the HAL model to use by the pid controller. The auto tuned settings are not perfect, but might provide a good starting point for further parameter tuning.

FIXME: The author of these instructions have not tested automatic tuning with tune-type set to 1, so this approach remain to be documented.

Armed with this knowledge, it is time to look at how to do the tuning. Lets say the HAL configuration in question load the PID component for X, Y and Z like this, using named pin names instead of *count=3*:

```
loadrt pid names=pid.x,pid.y,pid.z
```

If the component had used *count=3* instead, all use of pid.x, pid,y and pid.z need to be changed to pid.1, pid.2 and pid.3 respectively. To start tuning the X axis, move the axis to the middle of its range, to make sure it do not hit anything when it start moving back and forth. You also want to extend the axis ferror limit (following error) to make LinuxCNC accept the higher position deviation during tuning. The sensible ferror limit depends on the machine and setup, but 1 inch or 20 mm might be useful starting points. Next, set the initial *tune-effort* to a low number in the output range, for example 1/100 of the maximum output, and slowly increase it to get more accurate tuning values. Assign 1 to the *tune-mode* value. Note, this will disable the pid controlling part and feed the *bias* value to the output pin, which can cause a lot of drift. It might be a good idea to tune the motor driver to ensure zero input voltage do not cause any motor rotation, or adjust the *bias* value for the same effect. Finally, after setting *tune-mode*, set *tune-start* to 1 to activate the auto tuning. If all go well, your axis will vibrate and move back and forth for a few seconds and when it is done, new values for Pgain, Igain and Dgain will be active. To test them, change *tune-mode* back to 0. Note that setting *tune-mode* back to zero might cause the axis to suddenly jerk as it bring the axis back to its commanded position, which it might have drifted away from during tuning. To summarize, these are the halcmd instructions you need to issue to do automatic tuning:

```
setp pid.x.tune-effort 0.1
setp pid.x.tune-mode 1
setp pid.x.tune-start 1
# wait for the tuning to complete
setp pid.x.tune-mode 0
```

A script to help doing the automatic tuning is provided in the LinuxCNC code repository as *scripts/run-auto-pid-tuner*. This will ensure the machine is powered on and ready to run, home all axes if it is not already done, check that the extra tuning pins are available, move the axis to its mid point, run the auto tuning and re-enable the pid controller when it is done. It can be run several times.

9.6. Remap Extending G-code

9.6.1. Introduction: Extending the RS274NGC Interpreter by Remapping Codes

9.6.1.1. A Definition: Remapping Codes

By *remapping codes* we mean one of the following:

1. Definir la semántica de un código M o G nuevo, es decir, actualmente sin asignar
2. Redefine the semantics of a - currently limited - set of existing codes.

9.6.1.2. Why would you want to extend the RS274NGC Interpreter?

The set of codes (M,G,T,S,F) currently understood by the RS274NGC interpreter is fixed and cannot be extended by configuration options.

In particular, some of these codes implement a fixed sequence of steps to be executed. While some of these, like M6, can be moderately configured by activating or skipping some of these steps through INI file options, overall the behavior is fairly rigid. So - if your are happy with this situation, then this manual section is not for you.

In many cases, this means that supporting a non *out of the box* configuration or machine is either cumbersome or impossible, or requires resorting to changes at the *C/C+\+* language level. The latter is unpopular for good reasons - changing internals requires in-depth understanding of interpreter internals, and moreover brings its own set of support issues. While it is conceivable that certain patches might make their way into the main LinuxCNC distribution, the result of this approach is a hodge-podge of special-case solutions.

A good example for this deficiency is tool change support in LinuxCNC. While random tool changers are supported well, it is next to impossible to reasonably define a configuration for a manual-tool change machine with, for example, an automatic tool length offset switch being visited after a tool change, and offsets set accordingly. Also, while a patch for a very specific rack tool changer exists, it has not found its way back into the main code base.

However, many of these things may be fixed by using an O-word procedure instead of built-in code. Whenever the insufficient built-in code is to be executed, call the O-word procedure instead. While possible, this approach is cumbersome - it requires source-editing of NGC programs, replacing all calls to the deficient code by an O-word procedure call.

In its simplest form, a remapped code isn't much more than a spontaneous call to an O-word procedure. This happens behind the scenes - the procedure is visible at the configuration level, but not at the NGC program level.

Generally, the behavior of a remapped code may be defined in the following ways:

- Usted define una subrutina O-word que implementa el comportamiento deseado
- Alternatively, you may employ a Python function which extends the interpreter's behavior.

How to glue things together M- and G-codes, and O-words subroutine calls have some fairly different syntax.

O-word procedures, for example, take positional parameters with a specific syntax like so:

```
o<test> call [1.234] [4.65]
```

whereas M- or G-codes typically take required or optional *word* parameters. For instance, G76 (threading) requires the P,Z,I,J and K words, and optionally takes the R,Q,H, E and L words.

So it isn't simply enough to say *whenever you encounter code X, please call procedure Y* - at least some checking and conversion of parameters needs to happen. This calls for some *glue code* between the new code, and its corresponding NGC procedure to execute before passing control to the NGC procedure.

This glue code is impossible to write as an O-word procedure itself, since the RS274NGC language lacks the introspective capabilities and access into interpreter internal data structures to achieve the required effect. Doing the glue code in - again - *C/C+\+* would be an inflexible and therefore unsatisfactory solution.

How Embedded Python fits in To make a simple situation easy and a complex situation solvable, the glue issue is addressed as follows:

- For simple situations, a built-in glue procedure (`argspec`) covers most common parameter passing requirements.

- Para el remapeado de T, M6, M61, S, F hay un algo de código de unión Python estándar que debería cubrir la mayoría de las situaciones, ver unión estándar.
- Para situaciones más complejas, puede escribir su propio código de unión Python para implementar un nuevo comportamiento.

Embedded Python functions in the Interpreter started out as glue code, but turned out very useful well beyond that. Users familiar with Python will likely find it easier to write remapped codes, glue, O-word procedures, etc. in pure Python, without resorting to the somewhat cumbersome RS274NGC language at all.

A Word on Embedded Python Many people are familiar with *extending* the Python interpreter by *C/C++* modules, and this is heavily used in LinuxCNC to access Task, HAL and Interpreter internals from Python scripts. *Extending Python* basically means: Your Python script executes as *it is in the driver seat*, and may access non-Python code by importing and using extension modules written in *C/C+\+*. Examples for this are the LinuxCNC `hal`, `gcode` and `emc` modules.

Embedded Python is a bit different and less commonly known: The main program is written in C/C++ and may use Python like a subroutine. This is powerful extension mechanism and the basis for the *scripting extensions* found in many successful software packages. Embedded Python code may access *C/C+\+* variables and functions through a similar extension module method.

9.6.2. Getting started

Defining a code involves the following steps:

- Elegir un código - usar un código no asignado, o redefinir un código existente.
- Decide how parameters are handled.
- Decide if and how results are handled.
- Decide about the execution sequence.

9.6.2.1. Builtin Remaps

Please note that currently only some existing codes can be redefined, while there are many *free* codes that may be available for remapping. When developing redefined existing code, it is a good idea to start with an unassigned G- or M- code, so that you can use both an existing behavior as well as a new one. When you're done, redefine the existing code to use your remapping configuration.

- El conjunto actual de códigos M no utilizados, disponibles para definición de usuario, se puede encontrar en la sección de códigos M sin asignar.
- For G-codes, see the unallocated G-codes list.
- Existing codes that can be reassigned are listed in the remappable codes section.

Actualmente hay dos remapeos solo Python completos que están disponibles en stdglue.py:

- ignore_m6
- index_lathe_tool_with_wear

These are meant for use with lathe. Lathes don't use M6 to index the tools, they use the T command.

Este remapeo también agrega compensaciones de desgaste al offset de la herramienta, p. ej. T201 se indexaría a herramienta 2 (con el offset de herramienta de la herramienta 2) y añade compensación de desgaste 1. En la tabla de herramientas, los números de herramienta arriba de 10000 son compensaciones de desgaste, p. ej. en la tabla de herramientas, la herramienta 10001 sería compensación de desgaste 1.

Here is what you need in the INI to use them:

```
[RS274NGC]
REMAP=T python=index_lathe_tool_with_wear
REMAP=M6 python=ignore_m6

[PYTHON]
# where to find the Python code:

# code specific for this configuration
PATH_PREPEND=./

# generic support code - make sure this actually points to Python-stdglue
PATH_APPEND=../../nc_files/remap_lib/python-stdglue/

# import the following Python module
TOPLEVEL=toplevel.py

# the higher the more verbose tracing of the Python plugin
LOG_LEVEL = 0
```

También debe agregar el archivo Python requerido en su carpeta de configuración.

Upgrade an existing configuration

9.6.2.2. Picking a code

Note that currently only a few existing codes may be redefined, whereas there are many *free* codes which might be made available by remapping. When developing a redefined existing code, it might be a good idea to start with an unallocated G- or M-code, so both the existing and new behavior can be exercised. When done, redefine the existing code to use your remapping setup.

- El conjunto actual de códigos M no utilizados, disponibles para definición de usuario, se puede encontrar aquí.
- Unallocated G-codes are listed here.
- Existing codes which may be remapped are listed here.

9.6.2.3. Parameter handling

Let's assume the new code will be defined by an NGC procedure, and needs some parameters, some of which might be required, others might be optional. We have the following options to feed values to the procedure:

1. Extracting words from the current block and pass them to the procedure as parameters (like `X22.34` or `P47`),
2. referring to INI file variables,
3. refiriéndose a variables globales (como `#2200 = 47.11` o `#<_global_param> = 315.2`).

The first method is preferred for parameters of dynamic nature, like positions. You need to define which words on the current block have any meaning for your new code, and specify how that is passed to the NGC procedure. Any easy way is to use the argspec statement. A custom prolog might provide better error messages.

Using to INI file variables is most useful for referring to setup information for your machine, for instance a fixed position like a tool-length sensor position. The advantage of this method is that the parameters are fixed for your configuration, regardless which NGC file you're currently executing.

Referring to global variables is always possible, but they are easily overlooked.

Note there's a limited supply of words which may be used as parameters, so one might need to fall back to the second and third methods if many parameters are needed.

9.6.2.4. Handling results

Your new code might succeed or fail, for instance if passed an invalid parameter combination. Or you might choose to *just execute* the procedure and disregard results, in which case there isn't much work to do.

Epilog handlers help in processing results of remap procedures - see the reference section.

9.6.2.5. Execution sequencing

Executable G-code words are classified into modal groups, which also defines their relative execution behavior.

If a G-code block contains several executable words on a line, these words are executed in a predefined order of execution, not in the order they appear in block.

When you define a new executable code, the interpreter does not yet know where your code fits into this scheme. For this reason, you need to choose an appropriate modal group for your code to execute in.

9.6.2.6. An minimal example remapped code

To give you an idea how the pieces fit together, let's explore a fairly minimal but complete remapped code definition. We choose an unallocated M-code and add the following option to the INI file:

```
[RS274NGC]
REMAP=M400  modalgroup=10 argspec=Pq ngc=myprocedure
```

In a nutshell, this means:

- The `M400` code takes a required parameter `P` and an optional parameter `Q`. Other words in the current block are ignored with respect to the `M400` code. If the `P` word is not present, fail execution with an error.
- When an `M400` code is encountered, execute `myprocedure.ngc` along the other modal group 10 M-codes as per order of execution.
- The value of `P`, and `Q` are available in the procedure as local named parameters. The may be referred to as `#<P>` and `#<Q>`. The procedure may test whether the `Q` word was present with the EXISTS built in function.

The file `myprocedure.ngc` is expected to exists in the `[DISPLAY]NC_FILES` or `[RS274NGC]SUBROUTINE_PATH` directory.

A detailed discussion of REMAP parameters is found in the reference section below.

9.6.3. Configuring Remapping

9.6.3.1. The REMAP statement

To remap a code, define it using the `REMAP` option in `RS274NG` section of your INI file. Use one `REMAP` line per remapped code.

The syntax of the `REMAP` is:

`REMAP=<`*code*`> <`*options*`>`
: where <*code*> may be one of `T`,`M6`,`M61`,`S`,`F` (existing codes) or any of the unallocated M-codes or G-codes.

It is an error to omit the <*code*> parameter.

The options of the REMAP statement are separated by whitespace. The options are keyword-value pairs and currently are:

`modalgroup=<`*modal group*`>`

Códigos G
: the only currently supported modal group is 1, which is also the default value if no group is given. Group 1 means *execute alongside other G-codes*.

Códigos M
: Currently supported modal groups are: 5,6,7,8,9,10. If no modalgroup is give, it defaults to 10 (*execute after all other words in the block*).

T,S,F
: for these the modal group is fixed and any `modalgroup=` option is ignored.

`argspec=<`*argspec*`>`
: See description of the argspec parameter options. Optional.

`ngc=<`*ngc_basename*`>`
: Basename of an O-word subroutine file name. Do not specify an .ngc extension. Searched for in the directories specified in the directory specified in `[DISPLAY]PROGRAM_PREFIX`, then in `[RS274NGC]SUB`. Mutually exclusive with `python=`. It is an error to omit both `ngc=` and `python=`.

`python=<`*Python function name*`>`
: Instead of calling an ngc O-word procedure call a Python function. The function is expected to be defined in the `module_basename.oword` module. Mutually exclusive with `ngc=`.

`prolog=<`*Python function name*`>`
: Before executing an ngc procedure, call this Python function. The function is expected to be defined in the `module_basename.remap` module. Optional.

`epilog=<`*Python function name*`>`
: After executing an ngc procedure, call this Python function. The function is expected to be defined in the `module_basename.remap` module. Optional.

The `python`, `prolog` and `epilog` options require the Python Interpreter plugin to be configured, and appropriate Python functions to be defined there so they can be referred to with these options.

The syntax for defining a new code, and redefining an existing code is identical.

9.6.3.2. Useful REMAP option combinations

Note that while many combinations of argspec options are possible, not all of them make sense. The following combinations are useful idioms:

`argspec=<`*`words`*`> ngc=<`*`procname`*`> modalgroup=_<`*`group`*`>`
: The recommended way to call an NGC procedure with a standard argspec parameter conversion. Used if argspec is good enough. Note, it is not good enough for remapping the `Tx` and `M6`/`M61` tool change codes.

`prolog=<`*`pythonprolog`*`> ngc=<`*`procname`*`> epilog=<`*`pythonepilog`*`> modalgroup=<`*`group`*`>`
: Call a Python prolog function to take any preliminary steps, then call the NGC procedure. When done, call the Python epilog function to do any cleanup or result extraction work which cannot be handled in G-code. The most flexible way of remapping a code to an NGC procedure, since almost all of the Interpreter internal variables, and some internal functions may be accessed from the prolog and epilog handlers. Also, a longer rope to hang yourselves.

`python=<`*`pythonfunction`*`> modalgroup=<`*`group`*`>`
: Directly call to a Python function without any argument conversion. The most powerful way of remapping a code and going straight to Python. Use this if you do not need an NGC procedure, or NGC is just getting in your way.

`argspec=<`*`words`*`> python=<`*`pythonfunction`*`> modalgroup=<`*`group`*`>`
: Convert the argspec words and pass them to a Python function as keyword argument dictionary. Use it when you're too lazy to investigate words passed on the block yourself.

Note that if all you want to achieve is to call some Python code from G-code, there is the somewhat easier way of calling Python functions like O-word procedures.

9.6.3.3. The argspec parameter

The argument specification (keyword `argspec`) describes required and optional words to be passed to an ngc procedure, as well as optional preconditions for that code to execute.

An argspec consists of 0 or more characters of the class `[@A-KMNP-Za-kmnp-z^>]`. It can by empty (like `argspec=`).

An empty argspec, or no argspec argument at all implies the remapped code does not receive any parameters from the block. It will ignore any extra parameters present.

Note that RS274NGC rules still apply - for instance you may use axis words (e.g., `X`, `Y`, `Z`) only in the context of a G-code.

Axis words may also only be used if the axis is enabled. If only XYZ are enabled, ABCUVW will not be available to be used in argspec.

Words `F`, `S` and `T` (short `FST`) will have the normal functions but will be available as variables in the remapped function. `F` will set feedrate, `S` will set spindle RPM, `T` will trigger the tool prepare function. Words `FST` should not be used if this behavior is not desired.

Words `DEIJKPQR` have no predefined function and are recommended for use as argspec parameters.

`ABCDEFHIJKPQRSTUVWXYZ`
: Defines a required word parameter: an uppercase letter specifies that the corresponding word **must** be present in the current block. The word`s value will be passed as a local named parameter with a corresponding name. If the `@` character is present in the argspec, it will be passed as positional parameter, see below.

abcdefhijkpqrstuvwxyz
Defines an optional word parameter: a lowercase letter specifies that the corresponding word **may** be present in the current block. If the word is present, the word's value will be passed as a local named parameter. If the `@` character is present in the argspec, it will be passed as positional parameter, see below.

@
The `@` (at-sign) tells argspec to pass words as positional parameters, in the order defined following the `@` option. Note that when using positional parameter passing, a procedure cannot tell whether a word was present or not, see example below.

---

sugerencia
this helps with packaging existing NGC procedures as remapped codes. Existing procedures do expect positional parameters. With the `@` option, you can avoid rewriting them to refer to local named parameters.

---

^
The ^ (caret) character specifies that the current spindle speed must be greater than zero (spindle running), otherwise the code fails with an appropriate error message.

>
The > (greater-than) character specifies that the current feed must be greater than zero, otherwise the code fails with an appropriate error message.

n
The `n` (greater-than) character specifies to pass the current line number in the `n`local named parameter.

By default, parameters are passed as local named parameter to an NGC procedure. These local parameters appear as *already set* when the procedure starts executing, which is different from existing semantics (local variables start out with value 0.0 and need to be explicitly assigned a value).

Optional word parameters may be tested for presence by the `EXISTS(#<word>)` idiom.

Example for named parameter passing to NGC procedures Assume the code is defined as

```
REMAP=M400 modalgroup=10 argspec=Pq ngc=m400
```

and `m400.ngc` looks as follows:

```
o<m400> sub
(P is required since it is uppercase in the argspec)
(debug, P word=#<P>)
(the q argspec is optional since its lowercase in the argspec. Use as follows:)
o100 if [EXISTS[#<q>]]
    (debug, Q word set: #<q>)
o100 endif
o<m400> endsub
M2
```

- Ejecutando `M400` fallará con el mensaje `M400 definido por el usuario: falta: P`.
- La ejecución de `M400 P123` mostrará ` palabra P=123.000000`.
- La ejecución de `M400 P123 Q456` mostrará ` palabra P=123.000000` y `palabra asignada Q=456.000000`.

Example for positional parameter passing to NGC procedures Assume the code is defined as

`REMAP=M410 modalgroup=10 argspec=@PQr ngc=m410`

and `m410.ngc` looks as follows:

```
o<m410> sub
(debug, [1]=#1 [2]=#2 [3]=#3)
o<m410> endsub
M2
```

- La ejecución de `M410 P10` mostrará `m410.ngc: [1]=10.000000 [2]=0.000000`.
- La ejecución de `M410 P10 Q20` mostrará `m410.ngc: [1]=10.000000 [2]=20.000000`.

---

nota
you lose the capability to distinguish more than one optional parameter word, and you cannot tell whether an optional parameter was present but had the value 0, or was not present at all.

Simple example for named parameter passing to a Python function It's possible to define new codes *without* any NGC procedure. Here's a simple first example, a more complex one can be found in the next section.

Assume the code is defined as

`REMAP=G88.6 modalgroup=1 argspec=XYZp python=g886`

This instructs the interpreter to execute the Python function `g886` in the `module_basename.remap` module, which might look like so:

```
from interpreter import INTERP_OK
from emccanon import MESSAGE

def g886(self, **words):
    for key in words:
        MESSAGE("word '%s' = %f" % (key, words[key]))
    if words.has_key('p'):
        MESSAGE("the P word was present")
    MESSAGE("comment on this line: '%s'" % (self.blocks[self.remap_level].comment))
    return INTERP_OK
```

Try this with out with: g88.6 x1 y2 z3 g88.6 x1 y2 z3 p33 (a comment here)

You'll notice the gradual introduction of the embedded Python environment - see here for details. Note that with Python remapping functions, it make no sense to have Python prolog or epilog functions since it is executing a Python function in the first place.

Advanced example: Remapped codes in pure Python The `interpreter` and `emccanon` modules expose most of the Interpreter and some Canon internals, so many things which so far required coding in *C/C+\+* can be now be done in Python.

The following example is based on the `nc_files/involute.py` script - but canned as a G-code with some parameter extraction and checking. It also demonstrates calling the interpreter recursively (see `self.execute()`).

Assuming a definition like so (NB: this does not use argspec):

`REMAP=G88.1 modalgroup=1 py=involute`

The `involute` function in `python/remap.py` listed below does all word extraction from the current block directly. Note that interpreter errors can be translated to Python exceptions. Remember this is *readahead time* - execution time errors cannot be trapped this way.

```
import sys
import traceback
from math import sin,cos

from interpreter import *
from emccanon import MESSAGE
from util import lineno, call_pydevd
# raises InterpreterException if execute() or read() fails
throw_exceptions = 1

def involute(self, **words):
    """ remap function with raw access to Interpreter internals """

    if self.debugmask & 0x20000000: call_pydevd() # USER2 debug flag

    if equal(self.feed_rate,0.0):
        return "feedrate > 0 required"

    if equal(self.speed[0], 0.0):
        return "spindle speed > 0 required"

    plunge = 0.1 # if Z word was given, plunge - with reduced feed

    # inspect controlling block for relevant words
    c = self.blocks[self.remap_level]
    x0 = c.x_number if c.x_flag else 0
    y0 = c.y_number if c.y_flag else 0
    a  = c.p_number if c.p_flag else 10
    old_z = self.current_z

    if self.debugmask & 0x10000000:
        print("x0=%f y0=%f a=%f old_z=%f" % (x0,y0,a,old_z))

    try:
        #self.execute("G3456")  # would raise InterpreterException
        self.execute("G21",lineno())
        self.execute("G64 P0.001",lineno())
        self.execute("G0 X%f Y%f" % (x0,y0),lineno())

        if c.z_flag:
            feed = self.feed_rate
            self.execute("F%f G1 Z%f" % (feed * plunge, c.z_number),lineno())
            self.execute("F%f" % (feed),lineno())

        for i in range(100):
            t = i/10.
            x = x0 + a * (cos(t) + t * sin(t))
            y = y0 + a * (sin(t) - t * cos(t))
            self.execute("G1 X%f Y%f" % (x,y),lineno())

        if c.z_flag: # retract to starting height
            self.execute("G0 Z%f" % (old_z),lineno())

    except InterpreterException,e:
        msg = "%d: '%s' - %s" % (e.line_number,e.line_text, e.error_message)
  return msg

    return INTERP_OK
```

The examples described so far can be found in *configs/sim/axis/remap/getting-started* with complete working configurations.

9.6.4. Upgrading an existing configuration for remapping

The minimal prerequisites for using `REMAP` statements are as follows:

- The Python plug in must be activated by specifying a `[PYTHON]TOPLEVEL=<path-to-toplevel-script>` in the INI file.
- The toplevel script needs to import the `remap` module, which can be initially empty, but the import needs to be in place.
- The Python interpreter needs to find the remap.py module above, so the path to the directory where your Python modules live needs to be added with `[PYTHON]PATH_APPEND=<path-to-your-local-Python-d`
- Recommended: import the `stdglue` handlers in the `remap` module. In this case Python also needs to find `stdglue.py` - we just copy it from the distribution so you can make local changes as needed. Depending on your installation the path to `stdglue.py` might vary.

Assuming your configuration lives under `/home/user/xxx` and the INI file is `/home/user/xxx/xxx.ini`, execute the following commands.

```
$ cd /home/user/xxx
$ mkdir python
$ cd python
$ cp /usr/share/linuxcnc/ncfiles/remap_lib/python-stdglue/stdglue.py .
$ echo 'from stdglue import *' >remap.py
$ echo 'import remap' >toplevel.py
```

Now edit ``/home/user/`xxx``/`xxx``.ini`` and add the following:

```
[PYTHON]
TOPLEVEL=/home/user/xxx/python/toplevel.py
PATH_APPEND=/home/user/xxx/python
```

Now verify that LinuxCNC comes up with no error messages - from a terminal window execute:

```
$ cd /home/user/xxx
$ linuxcnc xxx.ini
```

9.6.5. Remapping tool change-related codes: T, M6, M61

9.6.5.1. Descripción general

If you are unfamiliar with LinuxCNC internals, first read the How tool change currently works section (dire but necessary).

Note than when remapping an existing code, we completely disable this codes' built-in functionality of the interpreter.

So our remapped code will need to do a bit more than just generating some commands to move the machine as we like - it will also need to replicate those steps from this sequence which are needed to keep the interpreter and `task` happy.

However, this does **not** affect the processing of tool change-related commands in `task` and `iocontrol`. This means when we execute step 6b this will still cause iocontrol to do its thing.

Decisions, decisions:

- Do we want to use an O-word procedure or do it all in Python code?

- ¿Es la secuencia HAL de `iocontrol` (los pines tool-prepare/tool-prepared y tool-change/tool-changed) lo suficientemente buena o necesitamos un tipo diferente de interacción HAL para nuestro cambiador de herramientas (por ejemplo: más pines HAL involucrados con una secuencia de interacción diferente)?

Depending on the answer, we have four different scenarios:

- When using an O-word procedure, we need prolog and epilog functions.
- If using all Python code and no O-word procedure, a Python function is enough.
- When using the `iocontrol` pins, our O-word procedure or Python code will contain mostly moves.
- Cuando necesitamos una interacción más compleja que la ofrecida por `iocontrol`, necesitamos definir completamente nuestra propia interacción, usando los pines `motion.digital*` y `motion.analog*`, y esencialmente ignorar los pines de `iocontrol` puenteándolos.

---

nota
If you hate O-word procedures and love Python, you are free to do it all in Python, in which case you would just have a `python=<`*function*`>` spec in the REMAP statement. But assuming most folks would be interested in using O-word procedures because they are more familiar with that, we'll do that as the first example.

So the overall approach for our first example will be:

1. We'd like to do as much as possible with G-code in an O-word procedure for flexibility. That includes all HAL interaction which would normally be handled by `iocontrol` - because we rather would want to do clever things with moves, probes, HAL pin I/O and so forth.
2. We'll try to minimize Python code to the extent needed to keep the interpreter happy, and cause `task` to actually do anything. That will go into the `prolog` and `epilog` Python functions.

9.6.5.2. Understanding the role of `iocontrol` with remapped tool change codes

`iocontrol` provides two HAL interaction sequences we might or might not use:

- When the NML message queued by a SELECT_TOOL() canon command is executed, this triggers the "raise tool-prepare and wait for tool-prepared to become high" HAL sequence in `iocontrol`, besides setting the *XXXX* pins
- When the NML message queued by the CHANGE_TOOL() canon command is executed, this triggers the "raise tool-change and wait for tool-changed to become high" HAL sequence in `iocontrol`, besides setting the *XXXX* pins

What you need to decide is whether the existing `iocontrol` HAL sequences are sufficient to drive your changer. Maybe you need a different interaction sequence - for instance more HAL pins, or maybe a more complex interaction. Depending on the answer, we might continue to use the existing `iocontrol` HAL sequences, or define our own ones.

For the sake of documentation, we'll disable these `iocontrol` sequences, and roll our own - the result will look and feel like the existing interaction, but now we have complete control over them because they are executed in our own O-word procedure.

So what we'll do is use some `motion.digital-*` and `motion.analog-*` pins, and the associated `M62 .. M68` commands to do our own HAL interaction in our O-word procedure, and those will effectively replace the `iocontrol` *tool-prepare/tool-prepared* and *tool-change/tool-changed* sequences. So we'll

define our pins replacing existing `iocontrol` pins functionally, and go ahead and make the `iocontrol` interactions a loop. We'll use the following correspondence in our example:

Correspondencia de pines `iocontrol` en los ejemplos

iocontrol.0 pin	**motion pin**
tool-prepare	digital-out-00
tool-prepared	digital-in-00
tool-change	digital-out-01
tool-changed	digital-in-01
tool-prep-number	analog-out-00
tool-prep-pocket	analog-out-01
tool-number	analog-out-02

Let us assume you want to redefine the M6 command, and replace it by an O-word procedure, but other than that things *should continue to work*.

So what our O-word procedure would do is to replace the steps outlined here. Looking through these steps you'll find that NGC code can be used for most of them, but not all. So the stuff NGC can't handle will be done in Python prolog and epilog functions.

9.6.5.3. Specifying the M6 replacement

To convey the idea, we just replace the built in M6 semantics with our own. Once that works, you may go ahead and place any actions you see fit into the O-word procedure.

Going through the steps, we find:

1. check for T command already executed - **execute in Python prolog**
2. check for cutter compensation being active - **execute in Python prolog**
3. stop the spindle if needed - **can be done in NGC**
4. quill up - **can be done in NGC**
5. if TOOL_CHANGE_AT_G30 was set:
   a. move the A, B and C indexers if applicable - **can be done in NGC**
   b. generate rapid move to the G30 position - **can be done in NGC**
6. send a CHANGE_TOOL Canon command to `task` - **execute in Python epilog**
7. set the numbered parameters 5400-5413 according to the new tool - **execute in Python epilog**
8. signal to `task` to stop calling the interpreter for readahead until tool change complete - **execute in Python epilog**

So we need a prolog, and an epilog. Lets assume our INI file incantation of the M6 remap looks as follows:

```
REMAP=M6   modalgroup=6  prolog=change_prolog ngc=change epilog=change_epilog
```

So the prolog covering steps 1 and 2 would look like so - we decide to pass a few variables to the remap procedure which can be inspected and changed there, or used in a message. Those are: `tool_in_spindle`, `selected_tool` (tool numbers) and their respective tooldata indices `current_pocket` and `selected_pocket`:

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).

```
def change_prolog(self, **words):
    try:
        if self.selected_pocket < 0:
            return "M6: no tool prepared"

        if self.cutter_comp_side:
            return "Cannot change tools with cutter radius compensation on"

        self.params["tool_in_spindle"] = self.current_tool
        self.params["selected_tool"] = self.selected_tool
        self.params["current_pocket"] = self.current_pocket
        self.params["selected_pocket"] = self.selected_pocket
        return INTERP_OK
    except Exception as e:
        return "M6/change_prolog: {}".format(e)
```

You will find that most prolog functions look very similar:

1. First test that all preconditions for executing the code hold, then
2. prepare the environment - inject variables and/or do any preparatory processing steps which cannot easily be done in NGC code;
3. then hand off to the NGC procedure by returning INTERP_OK.

Our first iteration of the O-word procedure is unexciting - just verify we got parameters right, and signal success by returning a positive value; steps 3-5 would eventually be covered here (see here for the variables referring to INI file settings):

```
O<change> sub
(debug, change: current_tool=#<current_tool>)
(debug, change: selected_pocket=#<selected_pocket>)
;
; insert any G-code which you see fit here, e.g.:
; G0  #<_ini[setup]tc_x>  #<_ini[setup]tc_y>  #<_ini[setup]tc_z>
;
O<change> endsub [1]
m2
```

Assuming success of `change.ngc`, we need to mop up steps 6-8:

```
def change_epilog(self, **words):
    try:
        if self.return_value > 0.0:
            # commit change
            self.selected_pocket =  int(self.params["selected_pocket"])
            emccanon.CHANGE_TOOL(self.selected_pocket)
            # cause a sync()
            self.tool_change_flag = True
            self.set_tool_parameters()
            return INTERP_OK
        else:
            return "M6 aborted (return code %.1f)" % (self.return_value)
```

```
    except Exception, e:
        return "M6/change_epilog: %s" % (e)
```

This replacement M6 is compatible with the built in code, except steps 3-5 need to be filled in with your NGC code.

Again, most epilogs have a common scheme:

1. First, determine whether things went right in the remap procedure,
2. then do any commit and cleanup actions which can't be done in NGC code.

9.6.5.4. Configuring `iocontrol` with a remapped M6

Note that the sequence of operations has changed: we do everything required in the O-word procedure - including any HAL pin setting/reading to get a changer going, and to acknowledge a tool change - likely with `motion.digital-*` and `motion-analog-*` IO pins. When we finally execute the `CHANGE_TOOL()` command, all movements and HAL interactions are already completed.

Normally only now `iocontrol` would do its thing as outlined here. However, we don't need the HAL pin wiggling anymore - all `iocontrol` is left to do is to accept we're done with prepare and change.

This means that the corresponding `iocontrol` pins have no function any more. Therefore, we configure `iocontrol` to immediately acknowledge a change by configuring like so:

```
# loop change signals when remapping M6
net tool-change-loop iocontrol.0.tool-change iocontrol.0.tool-changed
```

If you for some reason want to remap `Tx` (prepare), the corresponding `iocontrol` pins need to be looped as well.

9.6.5.5. Writing the change and prepare O-word procedures

The standard prologs and epilogs found in `ncfiles/remap_lib/python-stdglue/stdglue.py` pass a few *exposed parameters* to the remap procedure.

An *exposed parameter* is a named local variable visible in a remap procedure which corresponds to interpreter-internal variable, which is relevant for the current remap. Exposed parameters are set up in the respective prolog, and inspected in the epilog. They can be changed in the remap procedure and the change will be picked up in the epilog. The exposed parameters for remappable built in codes are:

- `T` (prepare_prolog): `#<tool>` , `#<pocket>`
- `M6` (change_prolog): `#<tool_in_spindle>`, `#<selected_tool>`, `#<current_pocket>`, `#<selected_pocket>`
- `M61` (settool_prolog): `#<tool>` , `#<pocket>`
- `S` (setspeed_prolog): `#<speed>`
- `F` (setfeed_prolog): `#<feed>`

If you have specific needs for extra parameters to be made visible, that can simply be added to the prolog - practically all of the interpreter internals are visible to Python.

9.6.5.6. Making minimal changes to the built in codes, including M6

Remember that normally remapping a code completely disables all internal processing for that code.

However, in some situations it might be sufficient to add a few codes around the existing M6 built in implementation, like a tool length probe, but other than that retain the behavior of the built in M6.

Since this might be a common scenario, the built in behavior of remapped codes has been made available within the remap procedure. The interpreter detects that you are referring to a remapped code within the procedure which is supposed to redefine its behavior. In this case, the built in behavior is used - this currently is enabled for the set: M6, M61,T, S, F. Note that otherwise referring to a code within its own remap procedure would be a error - a `remapping recursion`.

Slightly twisting a built in would look like so (in the case of M6):

```
REMAP=M6   modalgroup=6  ngc=mychange
```

```
o<mychange> sub
M6 (use built in M6 behavior)
(.. move to tool length switch, probe and set tool length..)
o<mychange> endsub
m2
```



atención
When redefining a built-in code, **do not specify any leading zeroes in G- or M-codes** - for example, say REMAP=M1 `..`, not REMAP=M01 `....`

See the `configs/sim/axis/remap/extend-builtins` directory for a complete configuration, which is the recommended starting point for own work when extending built in codes.

9.6.5.7. Specifying the T (prepare) replacement

If you're confident with the default implementation, you wouldn't need to do this. But remapping is also a way to work around deficiencies in the current implementation, for instance to not block until the "tool-prepared" pin is set.

What you could do, for instance, is: - En una T remapeada, simplemente establezca el equivalente del pin `tool-prepare`, pero **no** espere `tool-prepared` aquí. - In the corresponding remapped M6, wait for the `tool-prepared` at the very beginning of the O-word procedure.

Again, the `iocontrol` tool-prepare/tool-prepared pins would be unused and replaced by `motion.*` pins, so those would pins must be looped:

```
# loop prepare signals when remapping T
net tool-prep-loop iocontrol.0.tool-prepare iocontrol.0.tool-prepared
```

So, here's the setup for a remapped T:

```
REMAP=T  prolog=prepare_prolog epilog=prepare_epilog ngc=prepare
```

```
def prepare_prolog(self,**words):
    try:
        cblock = self.blocks[self.remap_level]
        if not cblock.t_flag:
            return "T requires a tool number"

        tool  = cblock.t_number
```

```
        if tool:
            (status, pocket) = self.find_tool_pocket(tool)
            if status != INTERP_OK:
                return "T%d: pocket not found" % (tool)
        else:
            pocket = -1 # this is a T0 - tool unload

        # these variables will be visible in the ngc O-word sub
        # as #<tool> and #<pocket> local variables, and can be
        # modified there - the epilog will retrieve the changed
        # values
        self.params["tool"] = tool
        self.params["pocket"] = pocket

        return INTERP_OK
    except Exception, e:
        return "T%d/prepare_prolog: %s" % (int(words['t']), e)
```

The minimal ngc prepare procedure again looks like so:

```
o<prepare> sub
; returning a positive value to commit:
o<prepare> endsub [1]
m2
```

And the epilog:

```
def prepare_epilog(self, **words):
    try:
        if self.return_value > 0:
            self.selected_tool = int(self.params["tool"])
            self.selected_pocket = int(self.params["pocket"])
            emccanon.SELECT_TOOL(self.selected_tool)
            return INTERP_OK
        else:
            return "T%d: aborted (return code %.1f)" % (int(self.params["tool"]),self. ↩
                return_value)

    except Exception, e:
        return "T%d/prepare_epilog: %s" % (tool,e)
```

The functions *prepare_prolog* and *prepare_epilog* are part of the *standard glue* provided by *nc_files/remap_li*
stdglue/stdglue.py. This module is intended to cover most standard remapping situations in a common way.

9.6.5.8. Error handling: dealing with abort

The built in tool change procedure has some precautions for dealing with a program abort, e.g., by hitting escape in AXIS during a change. Your remapped function has none of this, therefore some explicit cleanup might be needed if a remapped code is aborted. In particular, a remap procedure might establish modal settings which are undesirable to have active after an abort. For instance, if your remap procedure has motion codes (G0,G1,G38..) and the remap is aborted, then the last modal code will remain active. However, you very likely want to have any modal motion canceled when the remap is aborted.

The way to do this is by using the `[RS274NGC]ON_ABORT_COMMAND` feature. This INI option specifies a O-word procedure call which is executed if `task` for some reason aborts program execution. `on_abort` receives a single parameter indicating the cause for calling the abort procedure, which might be used for conditional cleanup.

The reasons are defined in nml_intf/emc.hh

```
EMC_ABORT_TASK_EXEC_ERROR = 1,
EMC_ABORT_AUX_ESTOP = 2,
EMC_ABORT_MOTION_OR_IO_RCS_ERROR = 3,
EMC_ABORT_TASK_STATE_OFF = 4,
EMC_ABORT_TASK_STATE_ESTOP_RESET = 5,
EMC_ABORT_TASK_STATE_ESTOP = 6,
EMC_ABORT_TASK_STATE_NOT_ON = 7,
EMC_ABORT_TASK_ABORT = 8,
EMC_ABORT_INTERPRETER_ERROR = 9,        // interpreter failed during readahead
EMC_ABORT_INTERPRETER_ERROR_MDI = 10,   // interpreter failed during MDI execution
EMC_ABORT_USER = 100  // user-defined abort codes start here
```

```
[RS274NGC]
ON_ABORT_COMMAND=O <on_abort> call
```

The suggested on_abort procedure would look like so (adapt to your needs):

```
o<on_abort> sub

G54 (origin offsets are set to the default)
G17 (select XY plane)
G90 (absolute)
G94 (feed mode: units/minute)
M48 (set feed and speed overrides)
G40 (cutter compensation off)
M5  (spindle off)
G80 (cancel modal motion)
M9  (mist and coolant off)

o100 if [#1 eq 5]
    (machine on)
o100 elseif [#1 eq 6]
    (machine off)
o100 elseif [#1 eq 7]
    (estopped)
o100 elseif [#1 eq 8]
    (msg, abort pressed)
o100 else
    (DEBUG, error parameter is [#1])
o100 endif

o<on_abort> endsub
m2
```



atención
Never use an M2 in a O-word subroutine, including this one. It will cause hard-to-find errors. For instance, using an M2 in a subroutine will not end the subroutine properly and will leave the subroutine NGC file open, not your main program.

Asegúrese de que `on_abort.ngc` esté en la ruta de búsqueda del intérprete (ubicación recomendada: `SUBROUTINE_PATH` para no desordenar su directorio `NC_FILES` con procedimientos internos).

Statements in that procedure typically would assure that post-abort any state has been cleaned up, like HAL pins properly reset. For an example, see `configs/sim/axis/remap/rack-toolchange`.

Note that terminating a remapped code by returning INTERP_ERROR from the epilog (see previous section) will also cause the `on_abort` procedure to be called.

9.6.5.9. Error handling: failing a remapped code NGC procedure

Si determina en su procedimiento de manejo que ocurrió alguna condición de error, no use M2 para finalizar su manejador - vea mas arriba:

If displaying an operator error message and stopping the current program is good enough, use the (abort, `__<message>__`) feature to terminate the handler with an error message. Note that you can substitute numbered, named, INI and HAL parameters in the text like in this example (see also tests/interp/abort-hot-comment/test.ngc):

```
o100 if [..] (some error condition)
     (abort, Bad Things! p42=#42 q=#<q> INI=#<_ini[a]x> pin=#<_hal[component.pin])
o100 endif
```

nota
INI and HAL variable expansion is optional and can be disabled in the INI file

If more fine grained recovery action is needed, use the idiom laid out in the previous example:

- Defina una función de epílogo, incluso si es solo para señalar una condición de error,
- pass a negative value from the handler to signal the error,
- inspeccionar el valor de retorno en la función de epílogo,
- take any recovery action needed,
- devolver la cadena de mensaje de error desde el manejador, que establecerá el mensaje de error del intérprete y aborta el programa (casi como abort, message=).

This error message will be displayed in the UI, and returning INTERP_ERROR will cause this error handled like any other runtime error.

Note that both (abort, <*msg*>) and returning INTERP_ERROR from an epilog will cause any ON_ABORT handler to be called as well if defined (see previous section).

9.6.6. Reasignando otros códigos existentes:

9.6.6.1. Automatic gear selection be remapping S (set spindle speed)

A potential use for a remapped S code would be *automatic gear selection* depending on speed. In the remap procedure one would test for the desired speed attainable given the current gear setting, and change gears appropriately if not.

9.6.6.2. Adjusting the behavior of M0, M1

A use case for remapping M0/M1 would be to customize the behavior of the existing code. For instance, it could be desirable to turn off the spindle, mist and flood during an M0 or M1 program pause, and turn these settings back on when the program is resumed.

For a complete example doing just that, see configs/sim/axis/remap/extend-builtins/, which adapts M1 as laid out above.

9.6.6.3. Adjusting the behavior of M7, M8, M9

An example for remapping the built in behavior of M7/M8/M9 is the option to pass optional arguments like a P word for more complex coolant control (eg through tool vs external coolant flow).

See `configs/sim/axis/remap/extend-builtins/`, for an example of such an extension of the built in behavior for M7,M8 and M9.

9.6.7. Creating new G-code cycles

A G-code cycle as used here is meant to behave as follows:

- En la primera invocación, se recogen las palabras asociadas y se ejecuta el ciclo de código G.
- If subsequent lines just continue parameter words applicable to this code, but no new G-code, the previous G-code is re-executed with the parameters changed accordingly.

An example: Assume you have `G84.3` defined as remapped G-code cycle with the following INI segment (see here for a detailed description of `cycle_prolog` and `cycle_epilog`):

```
[RS274NGC]
# A cycle with an O-word procedure: G84.3 <X- Y- Z- Q- P->
REMAP=G84.3 argspec=xyzabcuvwpr prolog=cycle_prolog ngc=g843 epilog=cycle_epilog modalgroup ↩
    =1
```

Executing the following lines:

```
g17
(1)   g84.3 x1 y2 z3  r1
(2)   x3 y4 p2
(3)   x6 y7 z5
(4)   G80
```

causes the following (note *R* is sticky, and *Z* is sticky since the plane is *XY*):

1. `g843.ngc` is called with words x=1, y=2, z=3, r=1
2. `g843.ngc` is called with words x=3, y=4, z=3, p=2, r=1
3. `g843.ngc` is called with words x=6, y=7, z=3, r=1
4. The `G84.3` cycle is canceled.

Besides creating new cycles, this provides an easy method for repackaging existing G-codes which do not behave as cycles. For instance, the `G33.1` Rigid Tapping code does not behave as a cycle. With such a wrapper, a new code can be easily created which uses `G33.1` but behaves as a cycle.

See *configs/sim/axis/remap/cycle* for a complete example of this feature. It contains two cycles, one with an NGC procedure like above, and a cycle example using just Python.

9.6.8. Configuring Embedded Python

The Python plugin serves both the interpreter, and `task` if so configured, and hence has its own section `PYTHON` in the INI file.

9.6.8.1. Python plugin : INI file configuration

[PYTHON]

TOPLEVEL = <*filename*>
Nombre de archivo de la secuencia inicial de comandos de Python para ejecutar en la puesta en marcha. Este script es responsable de configurar la estructura del nombre del paquete, ver más abajo.

PATH_PREPEND = <*directory*>
Prepend this directory to PYTHON_PATH. A repeating group.

PATH_APPEND = <*directory*>
Append this directory to PYTHON_PATH. A repeating group.

LOG_LEVEL = <*integer*>
Nivel de registro de las acciones relacionadas con el plugin. Aumente esto si sospecha problemas. Puede ser muy detallado.

RELOAD_ON_CHANGE = [0|1]
Reload the *TOPLEVEL* script if the file was changed. Handy for debugging but currently incurs some runtime overhead. Turn this off for production configurations.

9.6.8.2. Executing Python statements from the interpreter

For ad-hoc execution of commands the Python *hot comment* has been added. Python output by default goes to stdout, so you need to start LinuxCNC from a terminal window to see results. Example for the MDI window:

```
;py,print(2*3)
```

Note that the interpreter instance is available here as `this`, so you could also run:

```
;py,print(this.tool_table[0].toolno)
```

Here is an approach to use an O-word subroutine to read a preference file entry and add it as a G-code parameter.

```
(filename myofile.ngc)
o<mi-archivo-o> sub

;py,from interpreter import *
;py,import os
;py,from qtvcp.lib.preferences import Access

; find and print the preference file path
;py,CONFPATH = os.environ.get('CONFIG_DIR', '/dev/null')
; adjust for your preference file name
;py,PREFFILE = os.path.join(CONFPATH,'qtdragon.pref')
;py,print(PREFFILE)

; get an preference instance
;py,Pref = Access(PREFFILE)

; load a preference and print it
;py,this.params['toolToLoad']=Pref.getpref('Tool to load', 0, int,'CUSTOM_FORM_ENTRIES')
;py,print('Tool to load->:',this.params['toolToLoad'])

; return the value
o<myofile> endsub [#<toolToLoad>]
M2
```

9.6.9. Programming Embedded Python in the RS274NGC Interpreter

9.6.9.1. The Python plugin namespace

The namespace is expected to be laid out as follows:

oword
: Cualquier código llamable en este módulo es candidato para procedimientos Python O-word. Tenga en cuenta que el módulo de Python `oword` se verifica **antes** que un procedimiento NGC con el mismo nombre - en efecto, nombres en `oword` ocultarán los archivos NGC del mismo nombre base.

remap
: Python callables referenced in an argspec `prolog`,`epilog` or `python` option are expected to be found here.

namedparams
: Python functions in this module extend or redefine the namespace of predefined named parameters, see adding predefined parameters.

9.6.9.2. The Interpreter as seen from Python

The interpreter is an existing C++ class (*Interp*) defined in *src/emc/rs274ngc*. Conceptually all `oword.<funct` and `remap.<function>` Python calls are methods of this Interp class, although there is no explicit Python definition of this class (it is a *Boost.Python* wrapper instance) and hence receive the as the first parameter `self` which can be used to access internals.

9.6.9.3. The Interpreter `__init__` and `__delete__` functions

If the `TOPLEVEL` module defines a function `__init__`, it will be called once the interpreter is fully configured (INI file read, and state synchronized with the world model).

If the `TOPLEVEL` module defines a function `__delete__`, it will be called once before the interpreter is shutdown and after the persistent parameters have been saved to the `PARAMETER_FILE`.

Note_ at this time, the `__delete__` handler does not work for interpreter instances created by importing the `gcode` module. If you need an equivalent functionality there (which is quite unlikely), please consider the Python `atexit` module.

```
# this would be defined in the TOPLEVEL module

def __init__(self):
    # add any one-time initialization here
    if self.task:
  # this is the milltask instance of interp
  pass
    else:
  # this is a non-milltask instance of interp
        pass

def __delete__(self):
    # add any cleanup/state saving actions here
    if self.task: # as above
  pass
    else:
        pass
```

This function may be used to initialize any Python-side attributes which might be needed later, for instance in remap or O-word functions, and save or restore state beyond what PARAMETER_FILE provides.

If there are setup or cleanup actions which are to happen only in the milltask Interpreter instance (as opposed to the interpreter instance which sits in the gcode Python module and serves preview/progress display purposes but nothing else), this can be tested for by evaluating self.task.

An example use of __init__ and __delete__ can be found in configs/sim/axis/remap/cycle/python/topl initialising attributes, needed to handle cycles in ncfiles/remap_lib/python-stdglue/stdglue.py (and imported into configs/sim/axis/remap/cycle/python/remap.py).

9.6.9.4. Calling conventions: NGC to Python

Python code is called from NGC in the following situations:

- during normal program execution:
  - when an O-word call like O<proc> call is executed and the name oword.proc is defined and callable
  - cuando se ejecuta un comentario como ;py,<sentencia Python>.
  - during execution of a remapped code: any prolog=, python= and epilog= handlers.

Calling O-word Python subroutines

Arguments:

self
: La instancia del intérprete.

***args**
: The list of actual positional parameters. Since the number of actual parameters may vary, it is best to use this style of declaration:

```
# this would be defined in the oword module
def mysub(self, *args):
    print("number of parameters passed:", len(args))
    for a in args:
        print(a)
```

Return values of O-word Python subroutines Al igual que los procedimientos NGC pueden devolver valores, también lo hacen las subrutinas O-word Python. Se espera que sean uno de los siguientes

- no value (no return statement or the value None),
- a float or int value,
- a string, this means *this is an error message, abort the program*. Works like (abort, msg).

Any other return value type will raise a Python exception.

In a calling NGC environment, the following predefined named parameters are available:

#<value>
: Value returned by the last procedure called. Initialized to 0.0 on startup. Exposed in Interp as self.return_value (float).

#<value_returned>
: Indicates the last procedure called did `return` or `endsub` with an explicit value. 1.0 if true. Set to 0.0 on each `call`. Exposed in Interp was `self.value_returned` (int).

See also `tests/interp/value-returned` for an example.

Calling conventions for prolog= and epilog= subroutines Arguments are:

self
: La instancia del intérprete.

words
: Keyword parameter dictionary. If an argspec was present, words are collected from the current block accordingly and passed in the dictionary for convenience (the words could as well be retrieved directly from the calling block, but this requires more knowledge of interpreter internals). If no argspec was passed, or only optional values were specified and none of these was present in the calling block, this dict is empty. Word names are converted to lowercase.

Example call:

```
def minimal_prolog(self, **words): # in remap module
    print(len(words),” words passed”)
    for w in words:
        print(”%s: %s” % (w, words[w]))
    if words['p'] < 78: # NB: could raise an exception if p were optional
       return ”failing miserably”
    return INTERP_OK
```

Return values:

INTERP_OK
: Return this on success. You need to import this from `interpreter`.

un mensaje de texto
: Devolver una cadena desde un manejador significa 'esto es un mensaje de error, abortar el programa '. Funciona como `(abort, `__<mensaje>__`)`.

Calling conventions for python= subroutines Arguments are:

self
: La instancia del intérprete.

words
: Keyword parameter dictionary. The same kwargs dictionary as prologs and epilogs (see above).

The minimum `python=` function example:

```
def useless(self,  **words): # in remap module
    return INTERP_OK
```

Return values:

INTERP_OK
: Return this on success

un mensaje de texto
: Devolver una cadena desde un manejador significa 'esto es un mensaje de error, abortar el programa '. Funciona como `(abort, `__<mensaje>__ `)`.

Si el manejador necesita ejecutar una *operación queuebuster* (cambio de herramienta, sonda, lectura del pin HAL) se supone que se suspende la ejecución con la siguiente sentencia:

`yield INTERP_EXECUTE_FINISH`
: This signals `task` to stop read ahead, execute all queued operations, execute the *queue-buster* operation, synchronize interpreter state with machine state, and then signal the interpreter to continue. At this point the function is resumed at the statement following the `yield ..` statement.

Dealing with queue-buster: Probe, Tool change and waiting for a HAL pin Queue busters interrupt a procedure at the point where such an operation is called, hence the procedure needs to be restarted after the interpreter synch(). When this happens the procedure needs to know if it is restarted, and where to continue. The Python generator method is used to deal with procedure restart.

This demonstrates call continuation with a single point-of-restart:

```
def read_pin(self,*args):
    # wait 5secs for digital-input 00 to go high
    emccanon.WAIT(0,1,2,5.0)
    # cede control after executing the queue buster:
    yield INTERP_EXECUTE_FINISH
    # post-sync() execution resumes here:
    pin_status = emccanon.GET_EXTERNAL_DIGITAL_INPUT(0,0);
    print("pin status=",pin_status)
```



aviso
The *yield* feature is fragile. The following restrictions apply to the usage of *yield INTERP_EXECUTE_FINISH*:

- Python code executing a *yield INTERP_EXECUTE_FINISH* must be part of a remap procedure. Yield does not work in a Python oword procedure.
- A Python remap subroutine containing *yield INTERP_EXECUTE_FINISH* statement may not return a value, as with normal Python yield statements.
- Code following a yield may not recursively call the interpreter, like with `self.execute("<mdi command>")`. This is an architectural restriction of the interpreter and is not fixable without a major redesign.

9.6.9.5. Calling conventions: Python to NGC

El código NGC se ejecuta desde Python cuando

- the method `self.execute(<NGC code>[,<line number>])` is executed, or
- during execution of a remapped code, if a `prolog=` function is defined, the NGC procedure given in `ngc=` is executed immediately thereafter.

The prolog handler does not call the handler, but it prepares its call environment, for instance by setting up predefined local parameters.

Inserting parameters in a prolog, and retrieving them in an epilog Conceptually a prolog and an epilog execute at the same call level like the O-word procedure, that is after the subroutine call is set up, and before the subroutine endsub or return.

This means that any local variable created in a prolog will be a local variable in the O-word procedure, and any local variables created in the O-word procedure are still accessible when the epilog executes.

The `self.params` array handles reading and setting numbered and named parameters. If a named parameter begins with _ (underscore), it is assumed to be a global parameter; if not, it is local to the calling procedure. Also, numbered parameters in the range 1..30 are treated like local variables; their original values are restored on return/endsub from an O-word procedure.

Here is an example remapped code demonstrating insertion and extraction of parameters into/from the O-word procedure:

```
REMAP=m300 prolog=insert_param ngc=testparam epilog=retrieve_param modalgroup=10
```

```
def insert_param(self, **words): # in the remap module
    print("insert_param call level=",self.call_level)
    self.params["myname"] = 123
    self.params[1] = 345
    self.params[2] = 678
    return INTERP_OK

def retrieve_param(self, **words):
    print("retrieve_param call level=",self.call_level)
    print("#1=", self.params[1])
    print("#2=", self.params[2])
    try:
        print("result=", self.params["result"])
    except Exception,e:
  return "testparam forgot to assign #<result>"
    return INTERP_OK
```

```
o<testparam> sub
(debug, call_level=#<_call_level> myname=#<myname>)
; try commenting out the next line and run again
#<result> = [#<myname> * 3]
#1 = [#1 * 5]
#2 = [#2 * 3]
o<testparam> endsub
m2
```

`self.params()` returns a list of all variable names currently defined. Since `myname` is local, it goes away after the epilog finishes.

Calling the interpreter from Python You can recursively call the interpreter from Python code as follows:

```
self.execute(<NGC code>[,<line number>])
```

Examples:

```
  self.execute("G1 X%f Y%f" % (x,y))
  self.execute("O <myprocedure> call", currentline)
```

You might want to test for the return value being < `INTERP_MIN_ERROR`. If you are using lots of execute() statements, it is probably easier to trap InterpreterException as shown below.



atención
The parameter insertion/retrieval method described in the previous section does not work in this case. It is good enough for just

- executing simple NGC commands or a procedure call and
- advanced introspection into the procedure, and
- no es necesario pasar parámetros nombrados locales.

The recursive call feature is fragile.

Interpreter Exception during execute() if `interpreter.throw_exceptions` is nonzero (default 1), and self.execute() returns an error, the exception `InterpreterException` is raised. InterpreterException has the following attributes:

`line_number`
where the error occurred

`line_text`
the NGC statement causing the error

`error_message`
the interpreter's error message

Errors can be trapped in the following Pythonic way:

```
import interpreter
interpreter.throw_exceptions = 1
   ...
   try:
        self.execute("G3456")  #  raise InterpreterException

   except InterpreterException,e:
        msg = "%d: '%s' - %s" % (e.line_number,e.line_text, e.error_message)
        return msg  # replace builtin error message
```

Canon The canon layer is practically all free functions. Example:

```
import emccanon
def example(self,*args):
    ....
    emccanon.STRAIGHT_TRAVERSE(line,x0,y0,z0,0,0,0,0,0,0)
    emccanon.STRAIGHT_FEED(line,x1,y1,z1,0,0,0,0,0,0)
    ...
    return INTERP_OK
```

The actual canon functions are declared in `src/emc/nml_intf/canon.hh` and implemented in `src/emc/task`, The implementation of the Python functions can be found in `src/emc/rs274ncg/canonmodule.cc`.

9.6.9.6. Built in modules

The following modules are built in:

`interpreter`
Exposes internals of the Interp class. See `src/emc/rs274ngc/interpmodule.cc`, and the `tests/remap/` regression test.

`emccanon`
Exposes most calls of `src/emc/task/emccanon.cc`.

9.6.10. Adding Predefined Named Parameters

The interpreter comes with a set of predefined named parameters for accessing internal state from the NGC language level. These parameters are read-only and global, and hence cannot be assigned to.

Additional parameters may be added by defining a function in the `namedparams` module. The name of the function defines the name of the new predefined named parameter, which now can be referenced in arbitrary expressions.

To add or redefine a named parameter:

- Add a `namedparams` module so it can be found by the interpreter,
- define new parameters by functions (see below). These functions receive `self` (the interpreter instance) as parameter and so can access arbitrary state. Arbitrary Python capabilities can be used to return a value.
- Importar ese módulo desde el script `TOPLEVEL`.

```
# namedparams.py
# trivial example
def _pi(self):
    return 3.1415926535
```

```
#<circumference> = [2 * #<radius> * #<_pi>]
```

Functions in `namedparams.py` are expected to return a float or int value. If a string is returned, this sets the interpreter error message and aborts execution.

Ònly functions with a leading underscore are added as parameters, since this is the RS274NGC convention for globals.

It is possible to redefine an existing predefined parameter by adding a Python function of the same name to the `namedparams` module. In this case, a warning is generated during startup.

While the above example isn't terribly useful, note that pretty much all of the interpreter internal state is accessible from Python, so arbitrary predicates may be defined this way. For a slightly more advanced example, see `tests/remap/predefined-named-params`.

9.6.11. Standard Glue routines

Since many remapping tasks are very similar, I've started collecting working prolog and epilog routines in a single Python module. These can currently be found in *ncfiles/remap_lib/python-stdglue/stdglue.py* and provide the following routines:

9.6.11.1. T: `prepare_prolog` and `prepare_epilog`

These wrap a NGC procedure for T*x* Tool Prepare.

Actions of prepare_prolog The following parameters are made visible to the NGC procedure:

- `#<tool>` - the parameter of the `T` word
- `#<pocket>` - the corresponding pocket

If tool number zero is requested (meaning Tool unload), the corresponding pocket is passed as -1.

It is an error if:

- No tool number is given as T parameter,
- the tool cannot be found in the tool table.

Note that unless you set the `[EMCIO] RANDOM_TOOLCHANGER=1` parameter, tool and pocket number are identical, and the pocket number from the tool table is ignored. This is currently a restriction.

Actions of `prepare_epilog`

- Se espera que el procedimiento NGC devuelva un valor positivo, de lo contrario se da un mensaje de error que contiene el valor de retorno y el intérprete aborta.
- In case the NGC procedure executed the T command (which then refers to the built in T behavior), no further action is taken. This can be used for instance to minimally adjust the built in behavior be preceding or following it with some other statements.
- Otherwise, the `#<tool>` and `#<pocket>` parameters are extracted from the subroutine's parameter space. This means that the NGC procedure could change these values, and the epilog takes the changed values in account.
- Then, the Canon command `SELECT_TOOL(#<tool>)` is executed.

9.6.11.2. M6: `change_prolog` and `change_epilog`

These wrap a NGC procedure for M6 Tool Change.

Actions of `change_prolog`

- If there was no preceding T command which caused a pocket to be selected, the prolog aborts with an error message.
- If cutter radius compensation is on, the prolog aborts with an error message.

Then, the following parameters are exported to the NGC procedure:

- `#<tool_in_spindle>` : the tool number of the currently loaded tool
- `#<selected_tool>` : the tool number selected
- `#<selected_pocket>` : the selected tool's tooldata index

Actions of +change_epilog

- Se espera que el procedimiento NGC devuelva un valor positivo, de lo contrario se da un mensaje de error que contiene el valor de retorno y el intérprete aborta.
- In case the NGC procedure executed the M6 command (which then refers to the built in M6 behavior), no further action is taken. This can be used for instance to minimally adjust the built in behavior be preceding or following it with some other statements.
- Otherwise, the `#<selected_pocket>` parameter is extracted from the subroutine's parameter space, and used to set the interpreter's `current_pocket` variable. Again, the procedure could change this value, and the epilog takes the changed value in account.
- Then, the Canon command `CHANGE_TOOL(#<selected_pocket>)` is executed.
- The new tool parameters (offsets, diameter etc) are set.

9.6.11.3. G-code Cycles: `cycle_prolog` and `cycle_epilog`

These wrap a NGC procedure so it can act as a cycle, meaning the motion code is retained after finishing execution. If the next line just contains parameter words (e.g. new X,Y values), the code is executed again with the new parameter words merged into the set of the parameters given in the first invocation.

These routines are designed to work in conjunction with an argspec=<words> parameter. While this is easy to use, in a realistic scenario you would avoid argspec and do a more thorough investigation of the block manually in order to give better error messages.

The suggested argspec is as follows:

```
REMAP=G<somecode> argspec=xyzabcuvwqplr prolog=cycle_prolog ngc=<ngc procedure> epilog= ↩
    cycle_epilog modalgroup=1
```

This will permit `cycle_prolog` to determine the compatibility of any axis words give in the block, see below.

Actions of `cycle_prolog`

- Determine whether the words passed in from the current block fulfill the conditions outlined under Canned Cycle Errors.
  - Export the axis words as <x>, #<y> etc; fail if axis words from different groups (XYZ) (UVW) are used together, or any of (ABC) is given.
  - Export *L*- as #<l>; default to 1 if not given.
  - Export *P*- as #<p>; fail if p less than 0.
  - Export *R*- as #<r>; fail if r not given, or less equal 0 if given.
  - Fail if feed rate is zero, or inverse time feed or cutter compensation is on.
- Determine whether this is the first invocation of a cycle G-code, if so:
  - Add the words passed in (as per argspec) into a set of sticky parameters, which is retained across several invocations.
- Si no es así (una línea de continuación con nuevos parámetros) entonces
  - merge the words passed in into the existing set of sticky parameters.
- Export the set of sticky parameters to the NGC procedure.

Actions of `cycle_epilog`

- Determine si el código actual era en realidad un ciclo, si es así entonces
  - retain the current motion mode so a continuation line without a motion code will execute the same motion code.

9.6.11.4. S (Set Speed) : `setspeed_prolog` and `setspeed_epilog`

TBD

9.6.11.5. F (Set Feed) : `setfeed_prolog` and `setfeed_epilog`

TBD

9.6.11.6. M61 Set tool number : `settool_prolog` and `settool_epilog`

TBD

9.6.12. Remapped code execution

9.6.12.1. NGC procedure call environment during remaps

Normally, an O-word procedure is called with positional parameters. This scheme is very limiting in particular in the presence of optional parameters. Therefore, the calling convention has been extended to use something remotely similar to the Python keyword arguments model.

See LINKTO G-code/main Subroutines: sub, endsub, return, call.

9.6.12.2. Nested remapped codes

Remapped codes may be nested just like procedure calls - that is, a remapped code whose NGC procedure refers to some other remapped code will execute properly.

The maximum nesting level remaps is currently 10.

9.6.12.3. Sequence number during remaps

Sequence numbers are propagated and restored like with O-word calls. See `tests/remap/nested-remaps/w` for the regression test, which shows sequence number tracking during nested remaps three levels deep.

9.6.12.4. Debugging flags

The following flags are relevant for remapping and Python - related execution:

`EMC_DEBUG_OWORD`	`0x00002000`	traces execution of O-word subroutines
`EMC_DEBUG_REMAP`	`0x00004000`	traces execution of remap-related code
`EMC_DEBUG_PYTHON`	`0x00008000`	calls to the Python plug in
`EMC_DEBUG_NAMEDPARAM`	`0x00010000`	trace named parameter access
`EMC_DEBUG_USER1`	`0x10000000`	user-defined - not interpreted by LinuxCNC
`EMC_DEBUG_USER2`	`0x20000000`	user-defined - not interpreted by LinuxCNC

or these flags into the `[EMC]DEBUG` variable as needed. For a current list of debug flags see *src/emc/nml_intf/debugflags.h*.

9.6.12.5. Debugging Embedded Python code

Debugging of embedded Python code is harder than debugging normal Python scripts, and only a limited supply of debuggers exists. A working open-source based solution is to use the Eclipse IDE, and the PydDev Eclipse plug in and its remote debugging feature.

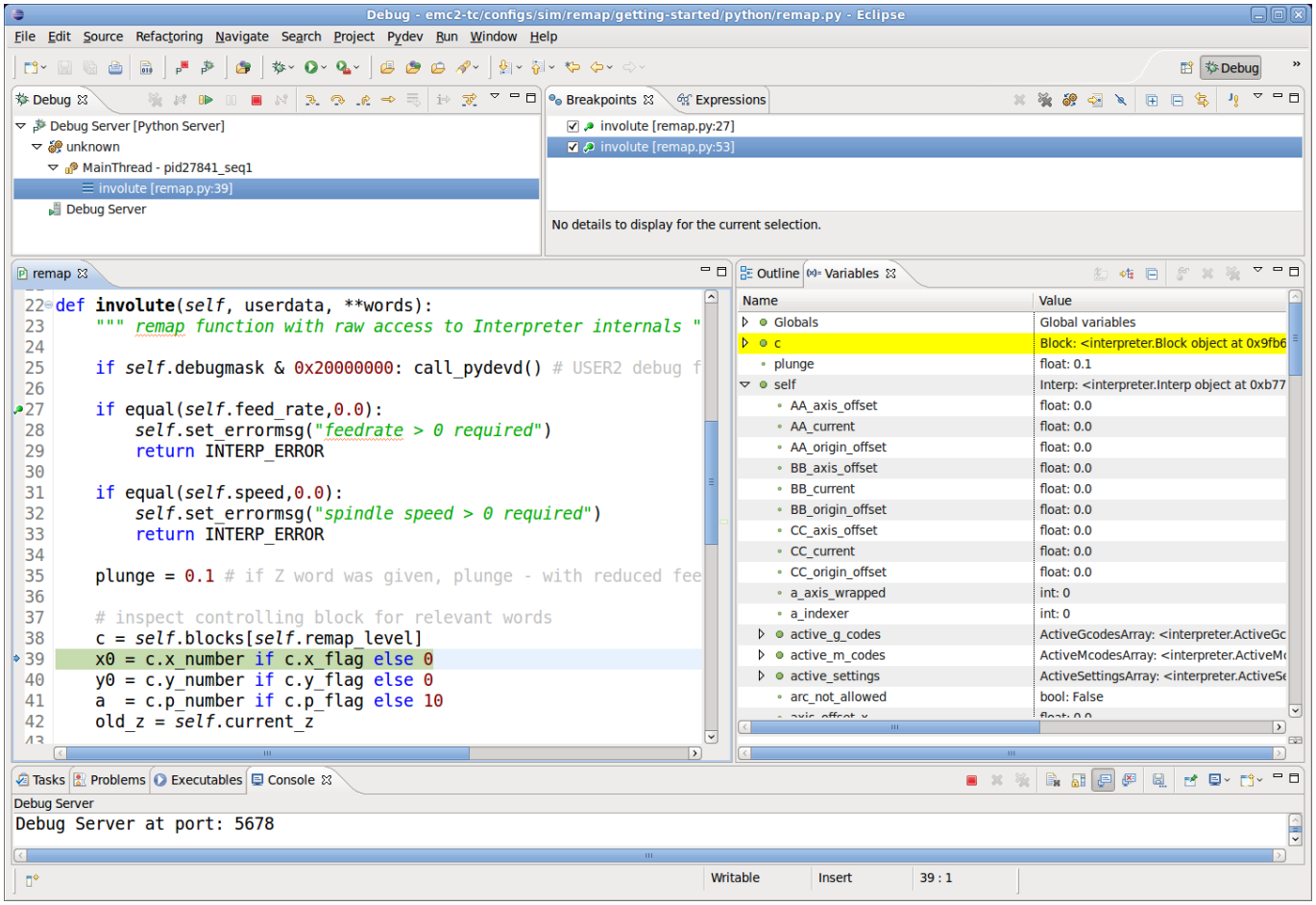
To use this approach:

- Instale Eclipse a través del *Centro de software de Ubuntu* (elija primero selección).
- Instale el plugin PyDev desde el sitio de actualización de Pydev.
- configure el árbol fuente de LinuxCNC como un proyecto de Eclipse.
- Inicie el servidor de depuración Pydev en Eclipse.
- Make sure the embedded Python code can find the `pydevd.py` module which comes with that plug in - it is buried somewhere deep under the Eclipse install directory. Set the the `pydevd` variable in `util.py` to reflect this directory location.
- Agregue `import pydevd` en su módulo Python - vea los ejemplos `util.py` y `remap.py`.
- Call `pydevd.settrace()` in your module at some point to connect to the Eclipse Python debug server - here you can set breakpoints in your code, inspect variables, step etc. as usual.

atención
`pydevd.settrace()` will block execution if Eclipse and the Pydev debug server have not been started.

To cover the last two steps: the `o<pydevd>` procedure helps to get into the debugger from MDI mode. See also the `call_pydevd` function in `util.py` and its usage in `remap.involute` to set a breakpoint.

Here's a screen-shot of Eclipse/PyDevd debugging the `involute` procedure from above:



See the Python code in `configs/sim/axis/remap/getting-started/python` for details.

9.6.13. Axis Preview and Remapped code execution

For complete preview of a remapped code's tool path some precautions need to be taken. To understand what is going on, let's review the preview and execution process (this covers the AXIS case, but others are similar):

First, note that there are **two** independent interpreter instances involved:

- Una instancia en el programa milltask, que ejecuta un programa cuando presiona el botón *Inicio* y hace que la máquina se mueva.
- A second instance in the user interface whose primary purpose is to generate the tool path preview. This one *executes* a program once it is loaded, but it doesn't actually cause machine movements.

Now assume that your remap procedure contains a G38 probe operation, for example as part of a tool change with automatic tool length touch off. If the probe fails, that would clearly be an error, so you'd display a message and abort the program.

Now, what about preview of this procedure? At preview time, of course it is not known whether the probe succeeds or fails - but you would likely want to see what the maximum depth of the probe is, and assume it succeeds and continues execution to preview further movements. Also, there is no point in displaying a *probe failed* message and aborting **during preview**.

The way to address this issue is to test in your procedure whether it executes in preview or execution mode. This can be checked for by testing the `#<_task>` predefined named parameter - it will be 1 during actual execution and 0 during preview. See *configs/sim/axis/remap/manual-toolchange-with-tool-length-switch/nc_subroutines/manual_change.ngc* for a complete usage example.

Within Embedded Python, the `task` instance can be checked for by testing *self.task* - this will be 1 in the milltask instance, and 0 in the preview instance(s).

9.6.14. Remappable Codes

9.6.14.1. Existing codes which can be remapped

The current set of **existing** codes open to redefinition is:

- T*x* (Prepare)
- M6 (Change tool)
- M61 (Set tool number)
- M0 (pause a running program temporarily)
- M1 (pause a running program temporarily if the optional stop switch is on)
- M7 (activate coolant mist)
- M8 (activate coolant flood)
- M9 (deactivate coolant mist and flood)
- M60 (exchange pallet shuttles and then pause a running program temporarily)
- M62 .. M65 (digital output control)
- M66 (wait on input)
- M67, M68 (analog output control)
- S (set spindle speed)
- F (set feed)

9.6.14.2. Currently unallocated G-codes:

Currently unallocated G-codes (for remapping) must be selected from the blank areas of the following tables. All the listed G-codes are already defined in the current implementation of LinuxCNC and may **not** be used to remap new G-codes. (Developers who add new G-codes to LinuxCNC are encouraged to also add their new G-codes to these tables.)

Cuadro 9.5: Table of Allocated G-codes 00-09

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
00	G00									
01	G01									
02	G02									
03	G03									
04	G04									
05	G05	G05.1	G05.2	G05.3						
06										
07	G07									
08	G08									
09										

Cuadro 9.6: Table of Allocated G-codes 10-19

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
10	G10									
11										
12										
13										
14										
15										
16										
17	G17	G17.1								
18	G18	G18.1								
19	G19	G19.1								

Cuadro 9.7: Table of Allocated G-codes 20-29

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
20	G20									
21	G21									
22										
23										
24										
25										
26										
27										
28	G28	G28.1								
29										

Cuadro 9.8: Table of Allocated G-codes 30-39

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
30	G30	G30.1								
31										
32										
33	G30	G30.1								
34										
35										
36										
37										
38										
39										

Cuadro 9.9: Table of Allocated G-codes 40-49

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
40	G40									
41	G41	G41.1								
42	G42	G42.1								
43	G43	G43.1								
44										
45										
46										
47										
48										
49	G40									

Cuadro 9.10: Table of Allocated G-codes 50-59

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
50										
51										
52										
53	G53									
54	G54									
55	G55									
56	G56									
57	G57									
58	G58									
59	G59	G59.1	G59.2	G59.3						

Cuadro 9.11: Table of Allocated G-codes 60-69

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
60	G60									
61	G61	G61.1								
62										

Cuadro 9.11: (continued)

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
63										
64	G64									
65										
66										
67										
68										
69										

Cuadro 9.12: Table of Allocated G-codes 70-79

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
70	G70									
71	G71	G71.1	G71.2							
72	G72	G72.1	G72.2							
73										
74										
75										
76	G76									
77										
78										
79										

Cuadro 9.13: Table of Allocated G-codes 80-89

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
80	G80									
81	G81									
82	G82									
83	G83									
84	G84									
85	G85									
86	G86									
87	G87									
88	G88									
89	G89									

Cuadro 9.14: Table of Allocated G-codes 90-99

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
90	G90	G90.1								
91	G91	G91.1								
92	G92	G92.1	G92.2	G92.3						
93	G93									
94	G94									
95	G95									

Cuadro 9.14: (continued)

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
96	G96									
97	G97									
98	G98									
99	G99									

9.6.14.3. Currently unallocated M-codes:

These M-codes are currently undefined in the current implementation of LinuxCNC and may be used to define new M-codes. (Developers who define new M-codes in LinuxCNC are encouraged to remove them from this table.)

Cuadro 9.15: Table of Unallocated M-codes 00-99

#	Mx0	Mx1	Mx2	Mx3	Mx4	Mx5	Mx6	Mx7	Mx8	Mx9
00-09										
10-19	M10	M11	M12	M13	M14	M15	M16	M17	M18	
20-29	M20	M21	M22	M23	M24	M25	M26	M27	M28	M29
30-39		M31	M32	M33	M34	M35	M36	M37	M38	M39
40-49	M40	M41	M42	M43	M44	M45	M46	M47		
50-59					M54	M55	M56	M57	M58	M59
60-69										
70-79					M74	M75	M76	M77	M78	M79
80-89	M80	M81	M82	M83	M84	M85	M86	M87	M88	M89
90-99	M90	M91	M92	M93	M94	M95	M96	M97	M98	M99

All M-codes from `M100` to `M199` are user-defined M-codes already, and should not be remapped.

All M-codes from `M200` to `M999` are available for remapping.

9.6.15. A short survey of LinuxCNC program execution

To understand remapping of codes it might be helpful to survey the execution of `task` and interpreter as far as it relates to remapping.

9.6.15.1. Interpreter state

Conceptually, the interpreter's state consist of variables which fall into the following categories:

1. *Configuration information* (typically from INI file)
2. *El 'modelo mundial'* - una representación del estado real de la máquina
3. *Modal state and settings* - refers to state which is *carried over* between executing individual NGC codes - for instance, once the spindle is turned on and the speed is set, it remains at this setting until turned off. The same goes for many codes, like feed, units, motion modes (feed or rapid) and so forth.
4. *Interpreter execution state* - Holds information about the block currently executed, whether we are in a subroutine, interpreter variables, etc. . Most of this state is aggregated in a - fairly unsystematic - `structure _setup` (see interp_internals.hh).

9.6.15.2. Task and Interpreter interaction, Queuing and Read-Ahead

The `task` part of LinuxCNC is responsible for coordinating actual machine commands - movement, HAL interactions and so forth. It does not by itself handle the RS274NGC language. To do so, `task` calls upon the interpreter to parse and execute the next command - either from MDI or the current file.

The interpreter execution generates canonical machine operations, which actually move something. These are, however, not immediately executed but put on a queue. The actual execution of these codes happens in the `task` part of LinuxCNC: canon commands are pulled off that interpreter queue, and executed resulting in actual machine movements.

This means that typically the interpreter is far ahead of the actual execution of commands - the parsing of the program might well be finished before any noticeable movement starts. This behavior is called *read-ahead*.

9.6.15.3. Predicting the machine position

To compute canonical machine operations in advance during read ahead, the interpreter must be able to predict the machine position after each line of G-code, and that is not always possible.

Let’s look at a simple example program which does relative moves (G91), and assume the machine starts at x=0,y=0,z=0. Relative moves imply that the outcome of the next move relies on the position of the previous one:

```
N10 G91
N20 G0 X10 Y-5 Z20
N30 G1 Y20 Z-5
N40 G0 Z30
N50 M2
```

Here the interpreter can clearly predict machine positions for each line:

Después N20: x=10 y=-5 z=20; después N30: x=10 y=15 z=15; después N40: x=10 y=15 z=45 y así pueden analizar todo el programa y generar operaciones canónicas con mucha antelación.

9.6.15.4. Queue-busters break position prediction

However, complete read ahead is only possible when the interpreter can predict the position impact for **every** line in the program in advance. Let’s look at a modified example:

```
N10 G91
N20 G0 X10 Y-5 Z20
N30 G38.3 Z-10
N40 O100 if [#5070 EQ 0]
N50    G1 Y20 Z-5
N60 O100 else
N70    G0 Z30
N80 O100 endif
N90 G1 Z10
N95 M2
```

To pre-compute the move in N90, the interpreter would need to know where the machine is after line N80 - and that depends on whether the probe command succeeded or not, which is not known until it is actually executed.

So, some operations are incompatible with further read-ahead. These are called *queue busters*, and they are:

- Reading a HAL pin's value with M66: value of HAL pin not predictable.
- Loading a new tool with M6: tool geometry not predictable.
- Executing a probe with G38.*n*: final position and success/failure not predictable.

9.6.15.5. Cómo tratar los busters de cola

Whenever the interpreter encounters a queue-buster, it needs to stop read ahead and wait until the relevant result is available. The way this works is:

- When such a code is encountered, the interpreter returns a special return code to `task` (*INTERP_EXECUTE*
- This return code signals to `task` to stop read ahead for now, execute all queued canonical commands built up so far (including the last one, which is the queue buster), and then *synchronize the interpreter state with the world model*. Technically, this means updating internal variables to reflect HAL pin values, reload tool geometries after an M6, and convey results of a probe.
- The interpreter's *synch()* method is called by `task` and does just that - read all the world model *actual* values which are relevant for further execution.
- At this point, `task` goes ahead and calls the interpreter for more read ahead - until either the program ends or another queue-buster is encountered.

9.6.15.6. Word order and execution order

One or several *words* may be present on an NGC *block* if they are compatible (some are mutually exclusive and must be on different lines). The execution model however prescribes a strict ordering of execution of codes, regardless of their appearance on the source line (G-code Order of Execution).

9.6.15.7. Parsing

Once a line is read (in either MDI mode, or from the current NGC file), it is parsed and flags and parameters are set in a *struct block* (struct _setup, member block1). This struct holds all information about the current source line, but independent of different ordering of codes on the current line: As long as several codes are compatible, any source ordering will result in the same variables set in the struct block. Right after parsing, all codes on a block are checked for compatibility.

9.6.15.8. Execution

After successful parsing the block is executed by execute_block(), and here the different items are handled according to execution order.

If a "queue buster" is found, a corresponding flag is set in the interpreter state (toolchange_flag, input_flag, probe_flag) and the interpreter returns an INTERP_EXECUTE_FINISH return value, signaling *stop readahead for now, and resynch* to the caller (*task*). If no queue busters are found after all items are executed, INTERP_OK is returned, signalling that read-ahead may continue.

When read ahead continues after the synch, `task` starts executing interpreter read() operations again. During the next read operation, the above mentioned flags are checked and corresponding variables are set (because the a synch() was just executed, the values are now current). This means that the next command already executes in the properly set variable context.

9.6.15.9. Procedure execution

Los procedimientos de O-word complican un poco el manejo de los generadores de colas. Un destructor de colas puede encontrarse en algún lugar en un procedimiento anidado, lo que resulta en una llamada a procedimiento semiacabada cuando es devuelto INTERP_EXECUTE_FINISH. Task se asegura de sincronizar el modelo universal y continua el análisis y ejecución siempre que haya un procedimiento en ejecución (call_level> 0).

9.6.15.10. How tool change currently works

The actions happening in LinuxCNC are a bit involved, but it is necessary to get the overall idea what currently happens, before you set out to adapt those workings to your own needs.

Note that remapping an existing code completely disables all internal processing for that code. That means that beyond your desired behavior (probably described through an NGC O-word or Python procedure), you need to replicate those internal actions of the interpreter, which together result in a complete replacement of the existing code. The prolog and epilog code is the place to do this.

Cómo se comunica la información de la herramienta Several processes are *interested* in tool information: `task` and its interpreter, as well as the user interface. Also, the *halui* process.

Tool information is held in the *emcStatus* structure, which is shared by all parties. One of its fields is the *toolTable* array, which holds the description as loaded from the tool table file (tool number, diameter, frontangle, backangle and orientation for lathe, tool offset information).

The authoritative source and only process actually *setting* tool information in this structure is the *iocontrol* process. All others processes just consult this structure. The interpreter holds actually a local copy of the tool table.

For the curious, the current emcStatus structure can be accessed by Python statements. The interpreter's perception of the tool currently loaded for instance is accessed by:

```
;py,from interpreter import *
;py,print(this.tool_table[0])
```

You need to have LinuxCNC started from a terminal window to see the results.

9.6.15.11. How T*x* (Prepare Tool) works

Interpreter action on a Tx command

All the interpreter does is evaluate the toolnumber parameter, looks up its corresponding tooldata index, remembers it in the `selected_pocket` variable for later, and queues a canon command (SELECT_TOOL). See *Interp::convert_tool_select* in *src/emc/rs274/interp_execute.cc*.

Task action on SELECT_TOOL When `task` gets around to handle a SELECT_TOOL, it sends a EMC_TOOL_PREPARE message to the `iocontrol` process, which handles most tool-related actions in LinuxCNC.

In the current implementation, `task` actually waits for `iocontrol` to complete the changer positioning operation, which is not necessary IMO since it defeats the idea that changer preparation and code execution can proceed in parallel.

Iocontrol action on EMC_TOOL_PREPARE When `iocontrol` sees the select pocket command, it does the related HAL pin wiggling - it sets the "tool-prep-number" pin to indicate which tool is next, raises the "tool-prepare" pin, and waits for the "tool-prepared" pin to go high.

Cuando el cambiador responde con "herramienta preparada", se considera completa la fase de preparación y se señala a `task` que continue. Otra vez, opino que esta *espera* no es estrictamente necesaria.

Construyendo prólogo y epílogo para Tx See the Python functions `prepare_prolog` and `prepare_epilog` in `nc_files/remap_lib/python-stdglue/stdglue.py`.

9.6.15.12. How M6 (Change tool) works

You need to understand this fully before you can adapt it. It is very relevant to writing a prolog and epilog handler for a remapped M6. Remapping an existing codes means you disable the internal steps taken normally, and replicate them as far as needed for your own purposes.

Even if you are not familiar with C, I suggest you look at the *Interp::convert_tool_change* code in *src/emc/rs274/interp_convert.cc*.

Interpreter action on a M6 command

When the interpreter sees an M6, it:

1. checks whether a `T` command has already been executed (test *settings->selected_pocket* to be >= 0) and fail with *Need tool prepared -Txx- for toolchange* message if not.
2. check for cutter compensation being active, and fail with *Cannot change tools with cutter radius compensation on* if so.
3. stop the spindle except if the ”TOOL_CHANGE_WITH_SPINDLE_ON” INI option is set.
4. generate a rapid *Z up* move if if the ”TOOL_CHANGE_QUILL_UP” INI option is set.
5. if TOOL_CHANGE_AT_G30 was set:
   a. move the A, B and C indexers if applicable
   b. generate rapid move to the G30 position
6. execute a CHANGE_TOOL canon command, with the selected pocket as a parameter. CHANGE_TOOL will:
   a. generate a rapid move to TOOL_CHANGE_POSITION if so set in INI
   b. enqueue an EMC_TOOL_LOAD NML message to `task`.
7. set the numberer parameters 5400-5413 according to the new tool
8. signal to `task` to stop calling the interpreter for readahead by returning INTERP_EXECUTE_FINISH since M6 is a queue buster.

What task does when it sees a CHANGE_TOOL command Again, not much more than passing the buck to `iocontrol` by sending it an EMC_TOOL_LOAD message, and waiting until `iocontrol` has done its thing.

Iocontrol action on EMC_TOOL_LOAD

1. it asserts the ”tool-change” pin
2. it waits for the ”tool-changed” pin to become active
3. when that has happened:
   a. deassert ”tool-change”
   b. establece los pines ”tool-prep-number” y ”tool-prep-pocket” a cero
   c. execute the *load_tool()* function with the pocket as parameter.

The last step actually sets the tooltable entries in the *emcStatus* structure. The actual action taken depends on whether the RANDOM_TOOLCHANGER INI option was set, but at the end of the process *toolTable[0]* reflects the tool currently in the spindle.

When that has happened:

1. `iocontrol` señala a `task` que puede seguir adelante.
2. `task` tells the interpreter to execute a *synch()* operation, to see what has changed.
3. The interpreter *synch()* pulls all information from the world model needed, among it the changed tool table.

From there on, the interpreter has complete knowledge of the world model and continues with read ahead.

Construyendo prólogo y epílogo para M6 See the Python functions `change_prolog` and `change_epilog` in `nc_files/remap_lib/python-stdglue/stdglue.py`.

9.6.15.13. How `M61` (Change tool number) works

`M61` requires a non-negative `Q`parameter (tool number). If zero, this means *unload tool*, else *set current tool number to Q*.

Building the replacement for M61 An example Python redefinition for `M61` can be found in the `set_tool_number` function in `nc_files/remap_lib/python-stdglue/stdglue.py`.

9.6.16. Status

1. La característica RELOAD_ON_CHANGE esta muy descompuesta. Reiniciar después de cambiar un archivo Python.

9.6.17. Changes

- El método para devolver mensajes de error y fallo solía ser *self.set_errormsg(texto)* seguido de *return INTERP_ERROR*. Esto ha sido reemplazado simplemente devolviendo una cadena desde un manejador de Python o subrutina oword. Esto establece el mensaje de error y aborta el programa. Anteriormente no había una forma clara de abortar una subrutina de palabra O de Python.

9.6.18. Debugging

In the *[EMC]* section of the INI file the *DEBUG* parameter can be changed to get various levels of debug messages when LinuxCNC is started from a terminal.

```
Debug level, 0 means no messages. See src/emc/nml_intf/debugflags.h for others
DEBUG = 0x00000002 # configuration
DEBUG = 0x7FFFDEFF # no interp,oword
DEBUG = 0x00008000 # py only
DEBUG = 0x0000E000 # py + remap + Oword
DEBUG = 0x0000C002 # py + remap + config
DEBUG = 0x0000C100 # py + remap + Interpreter
DEBUG = 0x0000C140 # py + remap + Interpreter + NML msgs
DEBUG = 0x0000C040 # py + remap + NML
DEBUG = 0x0003E100 # py + remap + Interpreter + oword + signals + namedparams
DEBUG = 0x10000000 # EMC_DEBUG_USER1 - trace statements
DEBUG = 0x20000000 # EMC_DEBUG_USER2 - trap into Python debugger
DEBUG = 0x10008000 # USER1, PYTHON
DEBUG = 0x30008000 # USER1,USER2, PYTHON # USER2 will cause involute to try to connect to ↩
    pydev
DEBUG = 0x7FFFFFFF # All debug messages
```

9.7. Componente moveoff

El componente HAL moveoff es un método exclusivo de HAL para implementar offsets. Ver la página de manual (*$ man moveoff*) para limitaciones y advertencias IMPORTANTES.

El componente moveoff se utiliza para offsets de posiciones articulares utilizando conexiones HAL personalizadas. Implementa una funcionalidad offset-mientras-programa-parado con conexiones apropiadas para los pines de entrada. Se soportan nueve articulaciones.

Los valores de los pines de offset de eje (offset-in-M) se aplican continuamente (respetando límites de valor, velocidad y aceleración) a los pines de salida (offset-current-M, pos-plusoffset-M, fb-minusoffset-M) cuando ambos pines (apply-offsets y move-enable) de habilitación de entrada son TRUE. Las dos entradas habilitadoras estan asociadas internamente con un AND. Si el pin apply-offsets va a FALSE mientras se aplican los offsets, se activa un *pin de advertencia* y se emite un mensaje. El pin de advertencia permanece TRUE hasta que se eliminen los offsets o el pin apply-offsets vuelva a TRUE.

Normalmente, el pin move-enable está conectado a controles externos y el pin apply-offsets está conectado a halui.program.is-paused (solo para offsets mientras se está en pausa) o establecido en TRUE (para offsets aplicados continuamente).

Los offsets aplicados se *devuelven automáticamente* a cero (respetando los límites) cuando cualquiera de las entradas de habilitación está desactivada. El valor cero de tolerancia se especifica mediante el valor epsilon del pin de entrada.

Los puntos de referencia se registran cuando el componente moveoff está habilitado. Los puntos de referencia son gestionados con los pines waypoint-sample-secs y waypoint-threshold. Cuando el pin backtrack-enable es TRUE, la ruta de retorno automático sigue los waypoints grabados. Cuando la memoria disponible para waypoints se agota, los offsets son congelados y el pin waypoint-limit va TRUE. Esta restricción se aplica independientemente del estado del pin backtrack-enable. Un pin de habilitación debe ser puesto a FALSE para permitir un retorno al original (posición sin offset).

Retroceder a través de puntos de ruta resulta en tasas de movimiento *más lentas* en la medida que los movimientos son punto a punto respetando la velocidad y la configuración de aceleración. Los pines de límite de velocidad y aceleración se pueden gestionar dinámicamente para controlar offsets en todo momento.

Cuando backtrack-enable es FALSE, el movimiento de retorno automático **NO** es coordinado; cada eje vuelve a cero a su propio ritmo. Si se desea una ruta controlada en esta condición, cada eje debe devolverse manualmente a cero antes de hacer FALSE un pin de habilitación.

Los pines waypoint-sample-secs, waypoint-threshold, y epsilon se evalúan solo cuando el componente está inactivo.

Se proporciona el pin de salida offsets-applied para indicar el estado actual a una GUI para que se pueda gestionar la reanudación del programa. Si el(los) offset(s) no es(son) cero cuando se desactiva el pin apply-offsets (por ejemplo, cuando se reanuda un programa tras offsets durante una pausa), los offsets se devuelven a cero (respetando límites) y se emite un mensaje de *Error*.



atención
Si los offsets están habilitados y aplicados y la máquina está apagada por cualquier razón, cualquier lógica HAL *externa* que maneje los pines habilitadores y las entradas offset-in-M es responsable de su estado cuando la máquina se vuelva a encender posteriormente.

LinuxCNC generalmente no conoce este medio de offsets solo-HAL ni está disponible en las pantallas de vista previa de las GUI. **No se proporciona protección** para movimientos con offset que exceden los límites suaves administrados por LinuxCNC. Ya que los límites suaves no se respetan, un movimiento de offset puede encontrar límites duros (o un **CHOQUE** si no hay interruptores de límite). Se recomienda limitar la carrera con el uso de las entradas offset-min-M y offset-max-M. Disparar un límite duro apagará la máquina - vea **Precaución** arriba.

Los valores offset-in-M pueden establecerse con configuraciones de archivo INI, controladas por una GUI, o gestionados por otros componentes y conexiones HAL. Los valores fijos pueden ser apropiados en casos simples donde la dirección y la cantidad de offset es bien definida pero se requiere un método de control para desactivar un pin de habilitación para devolver los offsets a cero. Las GUI pueden proporcionar medios para que los usuarios puedan establecer, incrementar, disminuir y acumular valores de offset para cada eje y puedan establecer los valores de offset-in-M en cero antes de hacer FALSE un pin de habilitación.

Los valores predeterminados para accel, vel, min, max, epsilon, waypoint-sample-secs y waypoint-threshold pueden inadecuados para alguna aplicación en particular. Este componente HAL desconoce los límites impuestos por LinuxCNC en otros lugares. Los usuarios deben probar el uso en una aplicación de simulador y comprender todos los riesgos antes de su uso en hardware.

Las configuraciones Sim que demuestran el componente y una GUI (moveoff_gui) se encuentran en:

- configs/sim/axis/moveoff (axis-ui)
- configs/sim/touchy/ngcgui (touchy-ui)

9.7.1. Modificar una configuración existente

Se puede utilizar un archivo HAL proporcionado por el sistema (LIB:hookup_moveoff.tcl) para adaptar una configuración existente para usar el componente moveoff. Configuraciones adicionales del archivo INI admiten el uso de una interfaz gráfica de usuario simple (moveoff_gui) para controlar los offsets.

Cuando el archivo HAL del sistema (LIB:hookup_moveoff.tcl) se especifica correctamente en un archivo INI de configuración, podrá:

1. Desconectar los pines de conexión joint.N.motor-pos-cmd y joint.N.motor-pos-fb originales
2. Cargar (loadrt) el componente moveoff (usando el nombre mv) con una personalidad configurada para acomodar todos los ejes identificados en el archivo INI
3. Agregar (addf) las funciones del componente moveoff en la secuencia requerida
4. Volver a conectar los pines joint.N.motor-pos-cmd y joint.N.motor-pos-fb para usar el componente moveoff
5. Establecer los parámetros operativos y los límites del componente moveoff para cada eje de acuerdo con la configuración adicional del archivo INI

Nota: La aplicación moveoff_gui admite configuraciones que utilizan módulos de cinemática conocidos con KINEMATICS_TYPE=KINEMATICS_IDENTITY. Los módulos soportados incluyen: trivkins. Con cinemática de identidad, moveoff_gui asigna cada nombre de eje especificado con el parámetro de línea de comando *-axes axisnames* a la articulación correspondiente.

Modifique una configuración existente de la siguiente manera:

Asegúrese de que haya una entrada de archivo INI para `[HAL]HALUI` y cree una nueva entrada `[HAL]HALFILE` para LIB:hookup_moveoff.tcl. La entrada para LIB:hookup_moveoff.tcl debe seguir todas las entradas `HALFILE=` para archivos HAL que conectan los pines para `joint.N.motor-pos-cmd`, `joint.N.motor-pos-fb`, y cualquier componente conectado a estos pines (componentes pid y codificadores en un sistema servo, por ejemplo).

```
[HAL]
HALUI = halui
MEDIO = archivo_configuracion_hal_existente_1
...
MEDIO = archivo_configuracion_hal_existente_n
HALFILE = LIB:hookup_moveoff.tcl
```

Agregue entradas de archivo INI para la configuración por eje para cada eje en uso (si una entrada no está definida, la entrada correspondiente de la sección [AXIS_n] será utilizada; si no se encuentra ninguna entrada, se utilizará lo predeterminado del componente moveoff).

nota
NO se recomienda usar los valores predeterminados de componente o la configuración de valores de sección [AXIS_n] para offset por eje.

```
[MOVEOFF_n]
MAX_LIMIT =
MIN_LIMIT =
MAX_VELOCITY =
MAX_ACCELERATION =
```

Agregar entradas de archivo INI para la configuración del componente moveoff (omitir para usar los valores predeterminados):

```
[MOVEOFF]
EPSILON =
WAYPOINT_SAMPLE_SECS =
WAYPOINT_THRESHOLD =
```

moveoff_gui se usa para hacer conexiones requeridas adicionales y proporcionar una GUI emergente para:

1. Proporcionar un botón de control para habilitar/deshabilitar los offsets
2. Proporcionar un botón de control para activar/desactivar el seguimiento
3. Proporcionar botones de control para aumentar/disminuir/Cero cada offset de eje
4. Mostrar el valor actual de offset de cada eje
5. Mostrar el estado de offset actual (deshabilitado, activo, eliminando, etc.)

Los botones de control provistos son opcionales dependiendo del estado del pin move-enable del componente. Se proporcionan tanto una pantalla como controles para habilitar offsets si el pin mv.move-enable NO está conectado cuando se inicia moveoff_gui. Para este caso, moveoff_gui maneja el pin de habilitación de movimiento del componente (denominado mv.move-enable), así como los offsets (mv.move-offset-in-M) y la habilitación de backtracking (mv.backtrack-enable)

Si el pin mv.move-enable está conectado cuando moveoff_gui se inicia, moveoff_gui proporcionará una pantalla pero NO controles. Este modo admite configuraciones que usan una perilla de trote u otros métodos de controlar las entradas de offsets y los pines de habilitación (mv.offset-in-M, mv.move-enable, mv.backtrack-enable).

Moveoff_gui realiza las conexiones necesarias para los pines del componente moveoff: mv.power_on y mv.apply-offsets. El pin mv.power_on está conectado al pin motion.motion-enabled (una nueva señal se crea automáticamente si es necesario). mv.apply-offsets está conectado a halui.program.is-paused o se establece en 1 dependiendo de la opción de línea de comando -mode [onpause | always]. Una nueva señal se crea automáticamente si es necesario.

Para usar moveoff_gui, agregar una entrada en la sección del archivo INI [APPLICATIONS] de la siguiente manera:

```
[APPLICATIONS]
# Nota: puede ser requerido un retraso (especificado en segundos) si las conexiones
# se hacen usando archivos HAL postgui ([HAL]POSTGUI_HALFILE=)
DELAY = 0
APP = moveoff_gui opción1 opción2 ...
```

Cuando el archivo HAL LIB:hookup_moveoff.tcl se usa para cargar y conectar el componente moveoff, el pin mv.move-enable no se conectará y se utilizarán los controles locales proporcionados por moveoff_gui. Este es el método más simple para probar o demostrar el componente moveoff cuando se ha modificado una configuración INI existente.

Para habilitar controles externos mientras se usa la pantalla moveoff_gui para valores de offsets y estado, los archivos HAL que siguen a LIB:hookup_moveoff.tcl debe hacer conexiones adicionales. Por ejemplo, las demostraciones proporcionadas (configs/sim/axis/moveoff/*.ini) usan un archivo HAL de sistema simple (llamado LIB:moveoff_external.hal) para conectar los pines mv.move-enable, mv.offset-in-M y mv.bactrack-enable a señales:

```
[HAL]
HALUI = halui
...
HALFILE = LIB:hookup_moveoff.tcl
HALFILE = LIB:moveoff_external.hal
```

Las conexiones realizadas por LIB:moveoff_external.hal (para una configuración de tres ejes) son:

```
net external_enable mv.move-enable

net external_offset_0 mv.offset-in-0
net external_offset_1 mv.offset-in-1
net external_offset_2 mv.offset-in-2

net external_backtrack_en mv.backtrack-enable
```

Estas señales (external_enable, external_offset_M, external_backtrack_en) pueden ser administradas por HALFILES posteriores (incluidos POSTGUI_HALFILEs) para proporcionar control personalizado del componente mientras se usa la pantalla moveoff_gui para valores de offset y estado actuales.

Moveoff_gui se configura con opciones de línea de comando. Para detalles sobre la operación de moveoff_gui, ver la página del manual:

```
$ man moveoff_gui
```

Para obtener una breve lista de las opciones de línea de comando para moveoff_gui, use la opción de ayuda de línea de comando:

```
$ moveoff_gui --help

Uso:
moveoff_gui [Opciones]

Opciones:
    [--help | -? | -- -h] (Este texto)

    [-mode [onpause | always]] (predeterminado: onpause)
                                   (onpause: muestra la gui cuando el programa está en pausa ↩
                                       )
                                   (always: muestra gui siempre)

    [-axes nombresdeejes]        (predeterminado: xyz (sin espacios))
                                (letras del conjunto de: x y z a b c u v w)
                                (ejemplo: -axes z)
                                (ejemplo: -axes xz)
                                (ejemplo: -axes xyz)
    [-inc valordeincremento]    (predeterminado: 0.001 0.01 0.10 1.0)
                                (especifique uno por cada -inc (hasta 4))
                                (ejemplo: -inc 0.001 -inc 0.01 -inc 0.1)
    [-size entero]       (predeterminado: 14)
                                (El tamaño general emergente de la gui se basa en el tamaño  ↩
                                    de la fuente)
```

```
    [-loc center|+x+y]     (predeterminado: center)
                               (ejemplo: -loc +10+200)
    [-autoresume]            (predeterminado: notused)
                               (reanudar el programa cuando move-enable sea false)
    [-delay segundos]      (predeterminado: 5 (retraso de reanudación))

Opciones para casos especiales:
    [-noentry]               (predeterminado: notused)
                              (no crear widgets de entrada)
    [-no_resume_inhibit]     (predeterminado: notused)
                              (no usar un pin de reanudar inhibición)
    [-no_pause_requirement]  (predeterminado: notused)
                              (sin verificación de halui.program.is-paused)
    [-no_cancel_autoresume]  (predeterminado: notused)
                              (útil para retractaciones de offset con simple)
                              (control externo)
    [-no_display]            (predeterminado: notused)
                              (Usar cuando tanto los controles externos como las pantallas)
                              (están en uso (ver Nota))

Nota: Si el pin de habilitación de movimiento de moveoff (mv.move-enable) está conectado ↩
    cuando
      moveoff_gui se inicia, se requieren controles externos y solo
      se proporcionan pantallas.
```

9.8. Intérprete independiente

El intérprete autónomo rs274 está disponible para su uso a través de la línea de comandos.

9.8.1. Utilización

```
Uso: rs274 [-p interp.so] [-t tool.tbl] [-v var-file.var] [-n 0 | 1 | 2]
          [-b] [-s] [-g] [archivo de entrada [archivo de salida]]

    -p: especifica el intérprete conectable a utilizar
    -t: especifica el archivo .tbl (tabla de herramientas) a usar
    -v: especifica el archivo .var (parámetros) a usar
    -n: especifica el modo de continuación:
           0: continuar
           1: ingresar al modo MDI
           2: detener (predeterminado)
    -b: Activa o desactiva el indicador de ”eliminación de bloque” (predeterminado: OFF)
    -s: Activa o desactiva el indicador ’pila de impresión’ (predeterminado: OFF)
    -g: Activa o desactiva el indicador ’ir (modo por lotes)’ (predeterminado: OFF)
    -i: especifica el archivo .ini (predeterminado: sin archivo ini)
    -T: llamar a task_init()
    -l: especifica el log_level (predeterminado: -1)
```

9.8.2. Ejemplo

Para ver la salida de un bucle, por ejemplo, podemos ejecutar rs274 en el siguiente archivo y ver que el ciclo nunca termina. Para salir del bucle, use Ctrl Z. Se necesitan los dos archivos siguientes para ejecutar el ejemplo.

test.ngc

```
#<test> = 123.352

o101 while [[#<test> MOD 60 ] NE 0]
(debug,#<test>)
    #<test> = [#<test> + 1]
o101 endwhile

M2
```

test.tbl

```
T1 P1 Z0.511 D0.125; fresa 1/8
T2 P2 Z0.1 D0.0625; fresa de extremo 1/16
T3 P3 Z1.273 D0.201; taladro de rosca n°7
```

comando

```
rs274 -g test.ngc -t test.tbl
```

9.9. External Axis Offsets

External axis offsets are supported during teleop (world) jogs and coordinated (G-code) motion. External axis offsets are enabled on a per-axis basis by INI file settings and controlled dynamically by INI input pins. The INI interface is similar to that used for wheel jogging. This type of interface is typically implemented with a manual-pulse-generator (mpg) connected to an encoder INI component that counts pulses.

9.9.1. INI File Settings

For each axis letter (**L** in xyzabcuvw):

```
[AXIS_L]OFFSET_AV_RATIO = value (controls accel/vel for external offsets)
```

1. Allowed values: 0 <= value <= 0.9
2. Disallowed values are replaced with 0.1 with message to stdout
3. Valor por defecto: 0 (desactiva el offset externo).
   Consecuencia: omitiendo [AXIS_L]OFFSET_AV_RATIO se desactiva el offset externo para el eje.
4. Si no es cero, OFFSET_AV_RATIO (**r**), ajusta la velocidad y aceleración máxima convencional (planificada) para preservar las restricciones en [AXIS_L]:

```
planning max velocity        = (1-r) * MAX_VELOCITY
external offset velocity     = (  r) * MAX_VELOCITY

planning max acceleration    = (1-r) * MAX_ACCELERATION
external offset acceleration = (  r) * MAX_ACCELERATION
```

9.9.2. HAL Pins

9.9.2.1. Per-Axis Motion HAL Pins

For each axis letter (***L*** in xyzabcuvw)

1. **axis.*L*.eoffset-enable** Entrada(bit): habilitado
2. **axis.*L*.eoffset-scale** Entrada(float): factor de escala
3. **axis.*L*.eoffset-count** Entrada (s32): entrada al registro de conteo
4. **axis.*L*.eoffset-clear** Entrada(bit): borrar el offset solicitado
5. **axis.*L*.eoffset** Salida(float): offset externo actual
6. **axis.*L*.eoffset-request** Salida(float): offset externo solicitado

9.9.2.2. Other Motion HAL Pins

1. **motion.eoffset-active** Salida(bit): se aplican offsets externos distintos de cero
2. **motion.eoffset-limited** Salida(bit): movimiento inhibido debido a límite suave

9.9.3. Utilización

The axis input HAL pins (enable,scale,counts) are similar to the pins used for wheel jogging.

9.9.3.1. Offset Computation

At each servo period, the *axis*.L.*eoffset-counts* pin is compared to its value in the prior period. The increase or decrease (positive or negative delta) of the *axis*.L.*eoffset-counts* pin is multiplied by the current *axis*.L.*eoffset-scale* pin value. This product is accumulated in an internal register and exported to the *axis*.L.*eoffset-request* HAL pin. The accumulation register is reset to zero at each machine-on.

The requested offset value is used to plan the movement for the offset that is applied to the L coordinate and represented by the *axis*.L.*eoffset* HAL pin. The planned motion respects the allocated velocity and acceleration constraints and may be limited if the net motion (offset plus teleop jogging or coordinated motion) reaches a soft limit for the L coordinate.

For many applications, the *axis*.L.*eoffset-scale* pin is constant and the net *axis*.L.*eoffset-request* response to *axis*.L.*eoffset-counts* is equivalent to the product of the accumulated value of *axis*.L.*eoffset-counts* and the (constant) *axis*.L.*eoffset-scale* pin values.

9.9.3.2. Machine-off/Machine-on

When the machine is turned off, the **current position** with **external offsets** is **maintained** so that there is no unexpected motion at turn off or turn on.

At each startup (machine-on), the internal counts register for each HAL pin *axis*.L.*eoffset-counts* is zeroed and the corresponding HAL output pin *axis*.L.*eoffset* is reset to zero.

In other words, external offsets are **defined as ZERO at each startup** (machine-on) regardless of the value of the *axis*.L.*eoffset-counts* pins. To avoid confusion, it is recommended that all *axis*.L.*eoffset-counts* pins are set to zero when the machine is off.

9.9.3.3. Límites de software

External axis offset movements are independently planned with velocity and acceleration settings specified by the *[AXIS_L]OFFSET_AV_RATIO*. The offsetting motion is not coordinated with teleop jogs nor with coordinated (G-code) motion. During teleop jogging and coordinated (G-code) motion, axis soft limits (*[AXIS_*L*]MIN_LIMIT,MAX_LIMIT*) restrict movement of the axis.

When external offsets are applied and motion reaches a soft limit (by external offset increases or teleop jogging or coordinated motion), the HAL pin *motion.eoffset-limited* is asserted and the axis value is held nominally to the soft limit. This HAL pin can be used by associated HAL logic to truncate additional eoffset counts or to stop the machine (connect to *halui.machine.off* for instance). If the axis is moved within the soft limit, the *motion.eoffset-limited* pin is reset.

When operating at a soft limit during coordinated motion that continues to change the planned axis value, the HAL output pin *axis*.L.*eoffset* will indicate the current offset — the distance needed to reach the limit instead of the computed offset request. This indicated value will change as the planned axis value changes.

The HAL pin *axis*.L.*eoffset-request* indicates the current requested offset that is the product of the internal counts register and the eoffset-scale. In general, the *axis*.L.*eoffset* pin value lags the *axis*.L.*eoffset-request* value since the external offset is subject to an acceleration limit. When operating at a soft limit, additional updates to the *axis*.L.*eoffset-counts* will continue to affect the requested external offset as reflected in the *axis*.L.*eoffset-request* HAL pin.

When teleop jogging with external offsets enabled **and** non-zero values applied, encountering a soft limit will stop motion in the offending axis **without a deacceleration interval**. Similarly, during coordinated motion with external offsets enabled, reaching a soft limit will stop motion with no deacceleration phase. For this case, it does not matter if the offsets are zero.

When motion is stopped with no deacceleration phase, system **acceleration limits may be violated** and lead to: 1) a following error (and/or a thump) for a servo motor system, 2) a loss of steps for a stepper motor system. In general, it is recommended that external offsets are applied in a manner to avoid approaching soft limits.

9.9.3.4. Notes

External offsets apply to axis coordinate letters (xyzabcuvw). All joints must be homed before external axis offsets are honored.

If an *axis*.L.*eoffset-enable* HAL pin is reset when its offset is non-zero, the offset is maintained. The offset may be cleared by:

1. a *Machine-off/Machine on* toggle
2. reactivating the enable pin and incrementing/decrementing the *axis*.L.*eoffset-counts* HAL pin to return the offset to zero.
3. pulsing the *axis*.L.*eoffset-clear* HAL pin

External-offsets are intended for use with *small* offsets that are applied within the soft-limit bounds.

Soft limits are respected for both teleop jogging and coordinated motion when external offsets are applied. However, when a soft limit is reached during coordinated motion, reducing the offending external offset **may not move away** from the soft limit **if planned motion continues in the same direction**. This circumstance can occur since the rate of correcting offset removal (as set by *[AXIS_L]OFFSET_AV_RATIO*) may be less than the opposing planned rate of motion. In such cases, **pausing** (or stopping) the planned, coordinated motion will allow movement away from the soft limit when correcting changes are made in the offending external offset.

9.9.3.5. Warning

The use of external offsets can alter machine motion in a significant manner. The control of external offsets with HAL components and connections and any associated user interfaces should be carefully designed and tested before deployment.

9.9.4. Related HAL Components

9.9.4.1. eoffset_per_angle.comp

Component to compute an external offset from a function based on a measured angle (rotary coordinate or spindle). See the man page for details (**$ man eoffset_per_angle**).

9.9.5. Pruebas

The external axis offset capability is enabled by adding an *[AXIS_L]* setting for each candidate axis. For example:

```
[AXIS_Z]
OFFSET_AV_RATIO = 0.2
```

For testing, it is convenient to simulate a jog wheel interface using the **sim_pin** GUI. For example, in a terminal:

```
$ sim_pin axis.z.eoffset-enable axis.z.eoffset-scale axis.z.eoffset-counts
```

The use of external offsets is aided by displaying information related to the current offsets: the current eoffset value and the requested eoffset value, the axis pos-cmd, and (for an identity kinematics machine) the corresponding joint motor pos-cmd and motor-offset. The provided sim configuration (see below) demonstrates an example PyVCP panel for the AXIS GUI.

In the absence of a custom display, **halshow** can be started as an auxiliary application with a custom watch list.

Example INI file settings to simulate the HAL pin eoffset connections and display eoffset information for the z axis (for identity kinematics with z==joint2):

```
[APPLICATIONS]
APP = sim_pin \
      axis.z.eoffset-enable \
      axis.z.eoffset-scale \
      axis.z.eoffset-counts \
      axis.z.eoffset-clear

APP = halshow --fformat ”%0.5f” ./z.halshow
```

Where the file z.halshow (in the configuration directory) is:

```
pin+joint.2.motor-pos-cmd
pin+joint.2.motor-offset
pin+axis.z.pos-cmd
pin+axis.z.eoffset
pin+axis.z.eoffset-request
pin+motion.eoffset-limited
```

9.9.6. Ejemplos

Provided simulation configurations demonstrate the use of external offsets in order to provide a starting point for user customization for real hardware.

The sim configurations utilize the INI setting *[HAL]HALFILE = LIB:basic_sim.tcl* to configure all routine HAL connections for the axes specified in the INI file *[TRAJ]COORDINATES*= setting. The HAL logic needed to demonstrate external offset functionality and the GUI HAL pin connections for a PyVCP panel are made in separate HAL files. A non-simulation configuration should replace the *LIB:basic_sim.tcl* item HALFILEs appropriate to the machine. The provided PyVCP files (.hal and .xml) could be a starting point for application-specific GUI interfaces.

9.9.6.1. eoffsets.ini

La configuración sim *sim/configs/axis/external_offsets/eoffsets.ini* demuestra una máquina cartesiana XYZ con controles para habilitar offsets externos en cualquier eje.

Displays are provided to show all important position and offset values.

A sim_pin GUI provides controls for the axis offset pins: eoffset-scale & eoffset-counts (via signal e:<L>counts), eoffset-clear (via signal e:clearall)

A script (eoffsets_monitor.tcl) is used to set *axis.L.counts* pins to zero at Machine-off.

9.9.6.2. jwp_z.ini

La configuración sim *sim/configs/axis/external_offsets/jwp_z.ini* demuestra la implementacion de una capacidad *jog-while-pause* en una sola coordenada (Z):

Panel LEDs are provided to show important status items.

Controls are provided to set the eoffset scale factor and to increment/decrement/clear the eoffset counts.

9.9.6.3. dynamic_offsets.ini

Esta configuración sim *sim/configs/axis/external_offsets/dynamic_offsets.ini* demuestra los offsets aplicados dinámicamente mediante la conexión de una forma de onda senoidal a las entradas externas de offset de la coordenada z.

Panel LEDs are provided to show important status items.

Controls are provided to alter INI file settings for the Z axis max velocity and max acceleration.

Controls are provided to set the waveform generator parameters.

A halscope app is started to show the applied waveform, the offset response, and the motor cmd response.

---

nota
changes to the z coordinate max-acceleration and max-velocity are not acknowledged while a program is running.

9.9.6.4. opa.ini (eoffset_per_angle)

The opa.ini configuration uses the INI component eoffset_per_angle (**$ man eoffset_per_angle**) to demonstrate an XZC machine with functional offsets computed from the C coordinate (angle) and applied to the transverse (X) coordinate. Offset computations are based on a specified reference radius typically set by a program (or MDI) M68 command to control a **motion.analog-out-NN** pin.

Panel LEDs are provided to show important status items.

Functions are provided for inside and outside polygons (nsides >= 3), sine waves and square waves. The functions can be multiplied in frequency using the fmul pin and modified in amplitude using the rfrac pin (fraction of reference radius).

Controls are provided to start/stop offset waveforms and to set the function type and its parameters.

9.10. Tool Database Interface

Tool data is conventionally described by a tool table file specified by an inifile setting: [EMCIO]TOOL_TABLE=
A tool table file consists of a text line for each available tool describing the tool's parameters, see Tool Table Format.

The tool database interface provides an alternative method for obtaining tool data via a separate program that manages a database of tools.

9.10.1. Interface

9.10.1.1. INI file Settings

INI file settings enable the (optional) operation of a user-provided tool database program:

```
[EMCIO]
DB_PROGRAM = db_program [args]
```

When included, **db_program** specifies the path to a user-provided executable program that provides tooldata. Up to 10 space-separated args may be included and passed to the **db_program** at startup.

---

nota
INI file settings for [EMCIO]TOOL_TABLE are ignored when a **db_program** is specified.

---

nota
The **db_program** may be implemented in any language currently supported in LinuxCNC (e.g., BASH scripts, Python or Tcl scripts, C/C++ programs) as long as it conforms to the interface messages received on stdin and replied on stdout. A **db_program** could manage data from a flat file, a relational database (SQLite for example), or other data sources.

9.10.1.2. db_program operation (v2.1)

When a **db_progam** is specified, operation is as follows:

1. At startup, LinuxCNC starts the **db_program** and connects to its stdin and stdout.

2. The **db_program** must respond by writing a single line acknowledgement consisting of a version string (e.g., ”v2.1”). No tools will be available if the version is not compatible with the LinuxCNC database interface version.
3. Upon a successful acknowledgement, LinuxCNC issues a *g* (**get**) command to request all tools. The **db_program** must respond with a sequence of replies to identify each available tool. The textual reply format is identical to the text line format used in conventional tool table files. A final response of ”FINI” terminates the reply.
4. The **db_program** then enters an event wait loop to receive commands that indicate that tool data has been changed by LinuxCNC. Tool data changes include:
   - a) spindle loading(T*n* M6)/unloading(T0 M6)
   - b) tool parameter changes (G10L1P*n* for example)
   - c) tool substitutions (M61Q*n*).

When a tool data change occurs, LinuxCNC sends a command to the **db_program** consisting of an identifying command letter followed by a full or abbreviated tool data line. The **db_program** must respond with a reply to confirm receipt. If the reply includes the text ”*NAK*”, a message is printed to stdout but execution continues. The ”*NAK*” message signifies a lack of synchronization between the **db_program** and LinuxCNC — accompanying text should give an indication for the cause of the fault.

The commands issued for tool data changes are:

- ”p” put data changes caused by G10L1, G10L10, G10L11 G-codes. The tool data line will include all elements of a tool table text line.
- ”l” spindle_load (T*n*M6). The tool data line includes only the *T* and *P* items identifying the relevant tool number and pocket number.
- ”u” spindle_unload (T0M6). The tool data line includes only the *T* and *P* items identifying the relevant tool number and pocket number.

---

nota
When a NON_RANDOM tool changer is specified using [EMCIO]RANDOM_TOOL_CHANGER=0 (the default), the spindle_load command issued for T*n*M6 (or M61Q*n*) is: *l T*n *P0* (pocket 0 is the spindle). The spindle_unload command issued for T0M6 is *u T0 P0*.

---

nota
When a RANDOM tool changer is specified using [EMCIO]RANDOM_TOOL_CHANGER=1, a pair of spindle_unload/spindle_load commands are issued at each tool exchange. The pair of commands issued for T*n*M6 (or M61Q*n*) are *u T*u *P*m followed by *l T*n *P0*, where *u* is the current tool to be sent to pocket *m* and *n* is the new tool to load in the spindle (pocket *0*). By convention, a tool number of 0 is used to specify an empty tool,

---

9.10.1.3. Utilización

Using a **db_program** does not change the way LinuxCNC operates but provides support for new database functionality for tool management.

For example, a **db_program** database application can maintain the operating hours for all tools by tracking each load/unload of a tool. A machine could then have three 6 mm endmills in pockets 111, 112, and 113 with the database application programmed to assign tool number 110 to the 6 mm endmill with the fewest operating hours. Then, when a LinuxCNC program requests tool 110, the database would specify the appropriate pocket based on tool usage history.

Tool data changes made within LinuxCNC (*p*,*u*,*l* commands) are pushed immediately to the **db_program** which is expected to synchronize its source data. By default, LinuxCNC requests for tool data (*g* commands) are made at startup only. A database program may update tool usage data on a continuous basis so long-lived LinuxCNC applications may benefit by refreshing the tool data provided by the **db_program**. The G-code command **G10L0** can be used to request a tool data reload (*g* command) from within G-code programs or by MDI. A reload operation is also typically provided by a Graphical User Interface (GUI) so that on-demand reloads can be requested. For example, a Python GUI application can use:

```
#!/usr/bin/env python3
from  linuxcnc import command
command().load_tool_table()
```

Alternatively, a **db_program** may push its local data changes to synchronize its data with LinuxCNC by using the load_tool_table() interface command. Commands which push changes to LinuxCNC may be rejected if the interpreter is running. The interpreter state can be checked before issuing a load_tool_table() command. Example:

```
#! /usr/bin/env python3
import linuxcnc
s = linuxcnc.stat()
s.poll()
if s.interp_state == linuxcnc.INTERP_IDLE:
    linuxcnc.command().load_tool_table()
else: # defer loading until interp is idle
   ...
```

If the database application adds or removes tools after initialization, a call to tooldb_tools() must be issued with an updated user_tools list. The updated list of tools will be used on subsequent get commands or load_tool_table() requests.

---

nota
Un número de herramienta sólo debería eliminarse si no se encuentra actualmente cargado en el husillo.

Exporting the environmental variable DB_SHOW enables LinuxCNC prints (to stdout) that show tool data retrieved from the **db_program** at startup and at subsequent reloading of tool data.

Exporting the environmental variable DB_DEBUG enables LinuxCNC prints (to stdout) for additional debugging information about interface activity.

9.10.1.4. Programa de ejemplo

An example **db_program** (implemented as a Python script) is provided with the simulation examples. The program demonsrates the required operations to:

1. acknowledge startup version
2. receive tool data requests: *g* (**get** command)
3. receive tool data updates: *p* (**put** command)
4. receive tool load updates: *l* (**load_spindle** command)
5. receive tool unload updates: *u* (**unload_spindle** command)

9.10.1.5. Python tooldb module

The example program uses a LinuxCNC provided Python module (*tooldb*) that manages the low-level details for communication and version verification. This module uses callback functions specified by the **db_program** to respond to the *g* (get) command and the commands that indicate tool data changes (*p*, *l*, *u*).

The **db_program** uses the *tooldb* module by implementing the following Python code:

```
user_tools = list(...)   # list of available tool numbers

def user_get_tool(toolno):
    # function to respond to 'g' (get) commands
    # called once for each toolno in user_tools
    ...
def user_put_tool(toolno,params):
    # function to respond to 'p' (put) commands
    ...
def user_load_spindle(toolno,params):
    # function to respond to 'l' (put) commands
    ...
def user_unload_spindle(toolno,params):
    # function to respond to 'u' (put) commands
    ...

#-----------------------------------------------------------
# Begin:
from tooldb import tooldb_tools     # identify known tools
from tooldb import tooldb_callbacks # identify functions
from tooldb import tooldb_loop      # main loop

tooldb_tools(user_tools)
tooldb_callbacks(user_get_tool,
                 user_put_tool,
                 user_load_spindle,
                 user_unload_spindle,
                )
tooldb_loop()
```

nota
Use of *tooldb* is not required — it is provided as a demonstration of the required interface and as a convenience for implementing Python-based applications that interface with an external database.

9.10.2. Simulation configs

Simulación de configuraciones usando la IGU AXIS:

1. configs/sim/axis/db_demo/**db_ran**.ini (random_toolchanger)
2. configs/sim/axis/db_demo/**db_nonran**.ini (nonrandom_toolchanger)

Each sim config simulates a **db_program** implementing a database with 10 tools numbered 10—19.

The **db_program** is provided by a single script (db.py) and symbolic links to it for alternative uses: db_ran.py and db_nonran.py. By default, the script implements random_toolchanger functionality. Nonrandom toolchanger functions are substituted if the link name includes the text "*nonran*".

The sim configs demonstrate the use of the Python *tooldb* interface module and implement a basic flat-file database that tracks tool time usage for multiple tools having equal diameters. The database rules support selection of the tool having the lowest operating time.

The sim configs use a primary task to monitor and respond to tool updates initiated from within LinuxCNC. A periodic task updates tool time usage at reguar intervals. Separate, concurrent tasks are implemented as threads to demonstrate the code required when changes are initiated by the **db_program** and demonstrate methods for synchronizing LinuxCNC internal tooldata. Examples include:

1. updates of tool parameters
2. addition and removal of tool numbers

A mutual exclusion lock is used to protect data from inconsistencies due to race conditions between LinuxCNC tooldata updates and the database application updates.

9.10.2.1. Notes

When a **db_program** is used in conjunction with a random tool changer ([EMCIO]RANDOM_TOOLCHANGE
LinuxCNC maintains a file (*db_spindle.tbl* in the configuration directory) that consists of a single tool table line identifying the current tool in the spindle.

Parte II

Utilización

Capítulo 10

Interfaces de usuario

10.1. GUI AXIS

10.1.1. Introducción

AXIS es una interfaz gráfica para LinuxCNC que cuenta con vista previa y trazo posterior. Está escrito en Python y utiliza Tk y OpenGL para mostrar la interfaz de usuario.

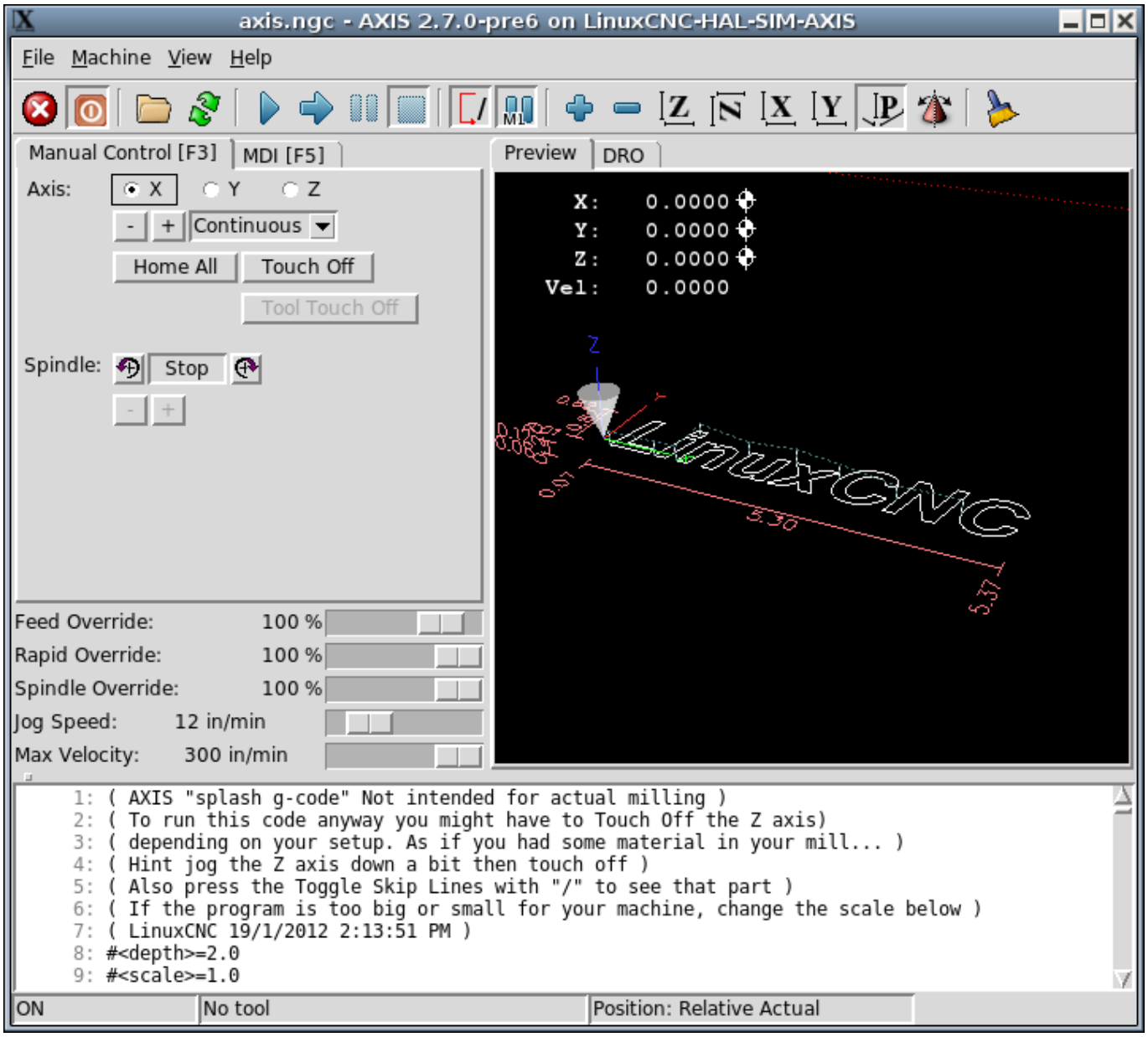


Figura 10.1: La ventana de AXIS

10.1.2. Primeros pasos

Si la configuración no está preparada actualmente para usar AXIS, se puede cambiar editando el archivo .ini (archivo INI). En la sección `[DISPLAY]` cambiar la línea a `DISPLAY = axis`.

La configuracion de ejemplo *sim/axis.ini* ya está configurada para usar AXIS como su interfaz de usuario.

Al iniciar AXIS se abrirá una ventana como la de la figura Figura 10.1 de arriba.

10.1.2.1. Configuraciones de INI

Para más información sobre las configuraciones del archivo INI que modifican el funcionamiento de AXIS ver la Sección DISPLAY y la Sección AXIS del capítulo Configuración de INI.

- *CYCLE_TIME* - Ajusta la velocidad de respuesta de la IGU en milisegundos. Típicamente 100, rango sable 50 - 200
  (acepta tiempo en segundos (.05 - .2) por razones de compatibilidad hacia atrás - se prefieren milisegundos para coincidir con otras pantallas).

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

- *PREVIEW_TIMEOUT* - Establece el tiempo de espera en segundos para cargar la vista previa de código G. Si el análisis gramatical del código G tarda más que este valor, se mostrará una advertencia y solo se mostrará en el área de visualización gráfica la parte inicial del programa. Especificar 0 o nada significa sin tiempo de espera.

```
[DISPLAY]
PREVIEW_TIMEOUT = 5
```

10.1.2.2. Una sesión típica

1. Iniciar LinuxCNC y seleccionar un archivo de configuración.
2. Liberar el botón E-STOP (F1) y encender la alimentación de la máquina (F2).
3. Llevar a casa todos los ejes.
4. Cargar el archivo de código G.
5. Utilizar el gráfico de vista previa para verificar que el programa está correcto.
6. Montar el material.
7. Establecer el offset adecuado para cada eje trotando y usando la tecla `Touch Off` según sea necesario.
8. Ejecutar el programa.
9. Para maquinar el mismo archivo otra vez regresar al paso 6. Para maquinar otro archivo regresar al paso 4.
10. Al terminar el trabajo salir AXIS.

---

nota
Ejecutar el mismo programa de nuevo dependerá de su configuración y requisitos. Es posible que se deba cargar más material y establecer offsets o desplazarse y establecer un nuevo offset y luego ejecutar el programa de nuevo. Si la ubicación del material es siempre la misma, entonces es posible que solo se tenga que volver a ejecutar el programa. Ver la sección Menu Máquina para obtener más información sobre el comando de ejecución.

10.1.3. Pantalla AXIS

La ventana de AXIS contiene los elementos siguientes:

- Un área de visualización que muestra uno de los siguientes elementos:

  - Una vista previa del archivo cargado (en este caso, *axis.ngc*), así como la ubicación actual del *punto controlado* de la máquina CNC. Más adelante, esta área mostrará el camino por el que la máquina CNC se ha movido, llamado *trazo posterior* (*backplot*).
  - Un lector de salida grande que muestra la posición actual y todos los offsets.
- Una barra de menús y una barra de herramientas que permiten realizar varias acciones
- Pestaña de *Control manual (F3)* - que permite mover la máquina, encender o apagar el husillo, y el refrigerante si se incluye en el archivo INI.
- Pestaña *MDI* - donde los programas en código G se pueden ingresar manualmente, una línea a la vez. También muestra los *Códigos G activos* que son los códigos G modales en vigor.
- *Ajustar avance* - que permite escalar la velocidad de los movimientos programados. El máximo predeterminado es 120 % y se puede configurar a un valor diferente en el archivo INI. Consultar la Sección DISPLAY del archivo INI para más información.
- *Ajustar Husillo* - que permite escalar la velocidad del husillo hacia arriba o hacia abajo.
- *Velocidad Jog* - que permite configurar la velocidad de trote dentro de los límites establecidos en el archivo INI. Ver la Sección DISPLAY del archivo INI para obtener más información.
- *Velocidad máxima* - que permite restringir la velocidad máxima de todos los movimientos programados (excepto movimiento sincronizado del husillo).
- Una zona de visualización de texto que muestra el código G cargado.
- Una barra de estado que muestra el estado de la máquina. En la pantalla mostrada, la máquina está encendida, no tiene una herramienta insertada y la posición mostrada es *Relativa* (mostrando todos los offsets) y *Actual* (mostrando la posición retroalimentada).

10.1.3.1. Elementos del menú

Algunos elementos del menú pueden estar en gris dependiendo de cómo este configurado el archivo INI. Para más información sobre configuración ver el Capítulo INI.

- *Abrir ...* - Abre un cuadro de diálogo estándar para abrir un archivo de código G para cargar en AXIS. Si ha configurado LinuxCNC para usar un programa de filtro, también puede abrirlo. Consulte la Sección FILTRO de la configuración INI para más información.
- *Archivos recientes* - muestra una lista de los archivos abiertos recientemente.
- *Editar ...* - abre el archivo de código G actual para editarlo si se tiene un editor configurado en el archivo INI. Consultar la sección sección DISPLAY para obtener más información sobre la especificación del editor a usar.
- *Recargar* - vuelve a cargar el archivo de código G actual. Si se editó, se debe recargar para que los cambios surtan efecto. Si se detiene un archivo y se quiere empezar desde el principio, volver a cargar el archivo. La recarga de la barra de herramientas es la misma que la del menú.
- *Guardar código G como ...* - Guarda el archivo actual con un nombre nuevo.
- *Propiedades* - La suma de los movimientos rápidos y de avance. No tiene en cuenta modos de aceleración, fusión o ruta para que el tiempo reportado nunca sea menor que el tiempo de ejecución real.
- *Editar datos de herramienta ...* - Igual que Editar, si se ha definido un editor, se puede abrir la tabla de herramientas y editarla.
- *Recargar datos de herramienta* - Después de editar la tabla de herramientas se debe volver a cargarla.

- *Editor Ladder* - Si se ha cargado ClassicLadder, se puede editar desde aquí. Ver el capítulo Classicladder para más información.
- *Salir* - Termina la sesión actual de LinuxCNC.

- *Conmutar parada de Emergencia F1*- Cambia el estado de la parada de emergencia.
- *Conmutar encendido de Máquina F2* - Cambia el estado de encendido de la máquina si la parada de emergencia no está encendida.
- *Ejecutar programa*- Ejecuta el programa actualmente cargado desde el principio.
- *Ejecutar desde línea seleccionada* - Selecciona la línea desde la que desea comenzar. Usarla con precaución ya que esto moverá primero la herramienta a la posición esperada antes de la línea y luego se ejecutará el resto del código.



aviso
No usar *Ejecutar desde línea seleccionada* si el programa de código G contiene subrutinas.

- *Paso de programa* - Un solo paso a través de un programa.
- *Pausa* - Pausa un programa.
- *Reanudar* - Reanuda la ejecución pausada.
- *Detener* - Detiene un programa en ejecución. Cuando se selecciona ejecutar después de una parada, el programa comenzará desde el principio.
- *Parar en M1* - Si se alcanza un M1, y este esta activo, la ejecución del programa parará en la línea M1. Presionar Reanudar para continuar.
- *Saltar líneas con "/"* - Si una línea comienza con / y esto está activo, la línea se saltará.
- *Borrar historial MDI* - Borra la ventana del historial de MDI.
- *Copiar desde historial MDI* - Copia el historial MDI al portapapeles
- *Pegar en historial MDI* - Pegar desde el portapapeles a la ventana del historial MDI
- *Calibración* - Inicia el asistente de calibración (emccalib.tcl). La calibración lee el archivo HAL y para cada *setp* que usa una variable del archivo INI que se encuentra en una de las secciones [AXIS_L], [JOINT_N], [SPINDLE_S] o [TUNE], crea una entrada que puede ser editada y probada.
- *Mostrar configuración HAL* - Abre la ventana de configuración HAL donde se pueden monitorear componentes HAL, pines, parámetros, señales, funciones e hilos.
- *HAL Meter* - Abre una ventana donde se puede monitorear un solo pin, señal o parámetro HAL.
- *Osciloscopio HAL* - Abre un osciloscopio virtual que permite seguir valores HAL en función del tiempo.
- *Mostrar estado de LinuxCNC* - Abre una ventana que muestra el estado de LinuxCNC.
- *Establecer nivel de depuración* - Abre una ventana donde se pueden ver los niveles de depuración y se pueden configurar algunos.
- *Homing* - Lleva a casa uno o todos los ejes.
- *Unhoming* - Deshacer la ida a casa de uno o todos los ejes.

- *Sistema de coordenadas cero* - Establece todos los offsets a cero en el sistema de coordenadas elegido.
- Toque inicial de la herramienta
  - *Touch off herramienta -pieza* - Al realizar Touch off, el valor ingresado es relativo al sistema de coordenadas de la pieza actual (*G5x*), modificado por el offset del eje (*G92*). Cuando se completa el Touch off, la coordenada relativa para el eje elegido se convertirá en el valor ingresado. Consultar <<gcode:g10-l10,G10 L10 en el capítulo de código G.
  - *Touch off herramienta - amarre* - Al realizar Touch off, el valor ingresado es relativo al noveno (*G59.3*) sistema de coordenadas, con el offset del eje (*G92*) ignorado. Esto es útil cuando hay un accesorio para Tool touch off en una ubicación fija en la máquina, con el noveno (*G59.3*) sistema de coordenadas establecido de tal manera que la punta de una herramienta de longitud cero esté en el origen del montaje cuando las coordenadas relativas son 0. Consultar G10 L11 en el capítulo de código G.
- *Vista superior* - La vista superior (o vista Z) muestra la previsualización del código G mirando en dirección del eje Z de positivo a negativo. Esta vista es la mejor para mirar el plano X y Y.
- *Vista superior girada* - La vista superior girada (o vista Z girada) también muestra el código G mirando a lo largo del eje Z de positivo a negativo. Pero a veces es conveniente mostrar los ejes X e Y girados 90 grados para que quede mejor la visualización. Este punto de vista es también mejor para mirar X y Y.
- *Vista lateral* - La vista lateral (o vista X) muestra el código G mirando a lo largo del eje X de positivo a negativo. Esta vista es mejor para mirar a Y & Z.
- *Vista frontal* - La vista frontal (o vista en Y) muestra el código G mirando hacia adelante el eje Y de negativo a positivo. Esta vista es mejor para mirar X y Z.
- *Vista en perspectiva* - La vista en perspectiva (o vista P) muestra el código G mirando la pieza desde un punto de vista ajustable, por defecto a X+, Y-, Z+. La posición es ajustable usando el mouse y el selector de arrastrar/rotar. Esta vista es una vista de compromiso, y si bien hace un buen trabajo al tratar de mostrar tres (¡hasta nueve!) ejes en una pantalla bidimensional, a menudo habrá alguna característica que es difícil de ver, que requiera un cambio en el punto de vista. Esta vista es la mejor cuando se quiera ver los tres (a nueve) ejes a la vez.

Punto de vista

El menú de selección de pantalla AXIS *Ver* se refiere a las vistas *Superior*, *Frontal* y *Lateral*. Estos términos son correctos si la máquina CNC tiene su eje Z vertical, con Z positivo hacia arriba. Esto es cierto para las fresadoras verticales, que es probablemente la aplicación más popular, y también es cierto para casi todas las máquinas EDM, e incluso tornos verticales de torreta, donde la pieza gira debajo de la herramienta.

Los términos *Superior*, *Frontal* y *Lateral* pueden ser confusos en otras máquinas CNC; como un torno estándar, donde el eje Z es horizontal, o una fresa horizontal, donde también el eje Z es horizontal, o incluso un torno vertical de torreta invertido, donde la pieza gira sobre la herramienta y la dirección positiva del eje Z ¡es hacia abajo!

Solo recuerde que el eje Z positivo está (casi) siempre *alejandose* de la pieza. Familiarícese con el diseño de su máquina e interprete la pantalla según sea necesario.

- *Mostrar pulgadas* - Establece la escala de la pantalla AXIS en pulgadas.
- *Mostrar MM* - Establece la escala de la pantalla AXIS en milímetros.
- *Mostrar programa* - La vista previa del programa de código G cargado puede ser completamente desactivada si lo desea.

- *Mostrar Rápidos del programa* - La vista previa del programa de código G cargado siempre mostrará la velocidad de avance (G1, G2, G3) en color blanco. Pero la visualización de movimientos rápidos (G0) en cian se puede desactivar si se desea.
- *Vista previa Alpha-blend* - Esta opción hace que la vista previa de programas complejos sea más fácil de ver, pero puede hacer que la vista previa se muestre más lentamente.
- *Mostrar plot en vivo* - El resaltado de las rutas de avance (G1, G2, G3) a medida que la herramienta se mueve se puede desactivar si se desea.
- *Mostrar herramienta* - La visualización del cono/cilindro de la herramienta se puede desactivar si se desea.
- *Mostrar extensiones* - La visualización de las extensiones (recorrido máximo en cada dirección del eje) del programa de código G cargado se puede desactivar si se desea.
- *Mostrar offsets* - Se puede mostrar la ubicación de origen del offset del montaje seleccionado (G54-G59.3) como un conjunto de tres líneas ortogonales, roja, azul y verde. Esta visualización de origen de offset (o punto cero) se puede desactivar si se desea.
- *Mostrar límites de máquina* - Los límites máximos de desplazamiento de la máquina para cada eje, según lo establecido en el archivo INI, se muestran como una caja rectangular dibujada en líneas discontinuas rojas. Esto es útil cuando se carga un nuevo programa de código G, o cuando se comprueba cuánto offset de montaje se necesita para llevar el programa de código G dentro de los límites de recorrido de tu máquina. Puede apagarse si no es necesario.
- *Mostrar velocidad* - Una visualización de la velocidad a veces es útil para ver qué tan cerca está funcionando su máquina de la velocidad de diseño. Puede ser desactivado si se desea.
- *Mostrar distancia restante* - Distancia a recorrer es un elemento muy útil que conocer cuando se ejecuta un programa de código G desconocido por primera vez. En combinación con los controles de porcentaje de velocidad rápida y de avance, se puede evitar el daño a herramientas y/o a la máquina. Una vez que el programa de código G se ha depurado y se ejecuta sin problemas, se puede desactivar la visualización si se desea.
- *Fuente grande de coordenadas* - Las coordenadas de los ejes y la velocidad de avance se mostrarán con una fuente mas grande en la vista de ruta de la herramienta.
- *Limpiar plot en vivo* - A medida que la herramienta se desplaza en la pantalla AXIS, se resalta la ruta del código G. Para repetir el programa, o para ver mejor un área de interés, se pueden borrar las rutas previamente resaltadas.
- *Mostrar posición ordenada* - Esta es la posición a la que intentará ir LinuxCNC. Una vez que el movimiento se ha detenido, esta es la posición que intentará mantener LinuxCNC.
- *Mostrar posición actual* - La posición real es la posición medida, leída desde los codificadores del sistema o simulada por los generadores de pasos. Esta puede diferir ligeramente de la posición ordenada por muchas razones, incluyendo afinación del PID, restricciones físicas, o cuantización de la posición.
- *Mostrar posición de la máquina* - Esta es la posición en coordenadas sin offset, según lo establecido por Homing.
- *Mostrar posición relativa* - Esta es la posición de la máquina modificada por los offsets *G5x*, *G92* y *G43*.

- *Acerca de AXIS* - Todos sabemos lo que es esto.
- *Referencia rápida* - Muestra las teclas de método abreviado del teclado.

10.1.3.2. Botones de la barra de herramientas

De izquierda a derecha en la pantalla de AXIS, los botones de la barra de herramientas (atajos de teclado mostrados [entre corchetes]) son:

-  Activar parada de emergencia [F1] (también llamado E-Stop)
-  Conmutar encendido de máquina [F2]
-  Abrir archivo de código G [O]
-  Reabrir el archivo actual [Ctrl-R]
-  Comenzar a ejecutar el archivo actual [R]
-  Ejecutar línea siguiente [T]
-  Pausar ejecución [P] Reanudar ejecución [S]
-  Detener la ejecución del programa [ESC]
-  Conmutar salto de líneas con ”/” [Alt-M-/]
-  Alternar pausa opcional [Alt-M-1]
-  Acercar
-  Alejar
-  Vista superior
-  Vista superior girada
-  Vista lateral
-  Vista frontal
-  Vista en perspectiva
-  Alternar entre los modos de arrastrar/rotar [D]
-  Limpiar trazo posterior en vivo [Ctrl-K]

10.1.3.3. Área de visualización gráfica

Visualización de coordenadas En la esquina superior izquierda de la pantalla del programa está la visualización de las coordenadas de posicion para cada eje. A la derecha del número, un símbolo de origen que se muestra si el eje ha sido llevado a casa.

Una símbolo de límite se muestra en el lado derecho del número de coordenada de posición, si el eje está en uno de sus interruptores de límite.

Para interpretar correctamente los números de coordenadas de posición, consulte el indicador *Posición:* en la barra de estado. Si la posición es *Máquina actual*, entonces el número mostrado está en el sistema de coordenadas de la máquina. Si se muestra *Relative Actual*, entonces el número mostrado está en la coordenada del sistema con desplazamiento. Cuando las coordenadas mostradas son relativas y se ha establecido un desplazamiento, la pantalla incluirá un marcador origen de máquina cian.

Si la posición es *Comandada*, entonces la coordenada exacta dada en un comando de código G es la mostrada. Si es *Actual*, entonces es la posición real a la que la máquina se ha movido. Estos valores pueden ser diferentes de la posición ordenada debido al error de seguimiento, banda muerta, resolución del codificador o tamaño de paso. Por ejemplo, si ordena un movimiento a X 0.0033 en su fresadora, pero el paso de su motor paso a paso o un conteo de codificador es 0.00125, la posición *Comandada* podría ser 0.0033, pero la posición *Actual* será 0.0025 (2 pasos) o 0.00375 (3 pasos).

Trazo de vista previa Cuando se carga un archivo, se muestra una vista previa en el área de visualización. Los movimientos rápidos (como los producidos por el comando *G0*) se muestran como lineas cian. Los movimientos a velocidad de avance (como los producidos por el comando *G1*) se muestran como líneas blancas sólidas. Las esperas (como las producidos por el comando *G4*) se muestran como pequeñas marcas *X* rosadas.

Movimientos G0 (rápido), antes de un movimiento de alimentación no se mostrarán en el trazo de vista previa. Los movimientos rápidos después de un T<n> (Cambio de herramienta) no se mostrarán en la vista previa hasta después del primer movimiento de alimentación. Para desactivar cualquiera de estas funciones programe un G1 sin ningún movimiento antes de los movimientos G0.

Dimensiones (físicas necesarias) del programa Se muestran las *dimensiones* resultantes del programa en cada eje. En los extremos, se indican los valores de coordenadas mínimo y máximo. En el medio, se muestra la diferencia entre las coordenadas (dimensión).

Cuando algunas coordenadas exceden los *límites de software* del archivo INI, la dimensión *culpable* se muestra en un color diferente y encerrada en un cuadro. En la figura de abajo se sobrepasa el límite de software máximo en el eje X, que se indica en el cuadro que rodea el valor de la coordenada. El mínimo recorrido X del programa es -1.95, el recorrido máximo de X es 1.88, y el programa requiere 3.83 pulgadas de recorrido X. Para que el movimiento programado esté dentro de la carrera de la máquina en este caso, trotar a la izquierda y volver a hacer touch off X.

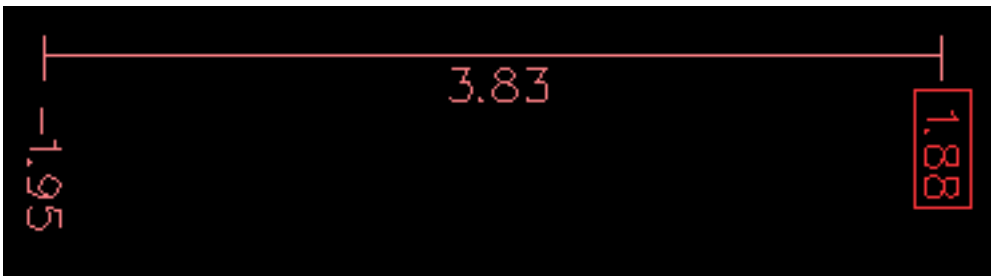


Figura 10.2: Límites de software

Cono de herramienta Cuando no se ha cargado ninguna herramienta, la ubicación de la punta de

la herramienta esta indicada por el *cono de herramienta*. El *cono de herramienta* no proporciona orientación sobre la forma, longitud, o radio de una herramienta real.

Cuando se carga una herramienta (por ejemplo, con el comando MDI *T1 M6*), el cono cambia a un cilindro que muestra el diámetro de la herramienta, dado en el archivo de tabla de herramientas.

Trazo posterior Cuando la máquina se mueve, deja un rastro en pantalla llamado trazo posterior (backplot). El color de la línea indica el tipo de movimiento: amarillo para trotes, verde claro para movimientos rápidos, rojo para movimientos rectos a velocidad de avance y magenta para movimientos circulares a velocidad de avance.

Cuadrícula Axis puede, opcionalmente, mostrar una cuadrícula en las vistas ortogonales. Habilite o deshabilite la cuadrícula usando *Gratícula* en el menu *Ver*. Cuando esta habilitada, la cuadrícula se muestra en las vistas superior y superior girada. Cuando el sistema de coordenadas no esta girado, la cuadrícula se muestra también en las vistas frontal y lateral. Los preajustes en el menú *Gratícula* están controlados por el elemento del archivo INI `[DISPLAY]GRIDS`. Si no se especifica, el valor predeterminado es `10mm 20mm 50mm 100mm 1in 2in 5in 10in`.

Especificar una cuadrícula muy pequeña puede disminuir el rendimiento.

Interacción Al hacer clic izquierdo en una parte del trazo de vista previa, la línea se resaltará en las visualizaciones tanto gráfica como de texto. Al hacer clic izquierdo en un área vacía, se quitará el resaltado.

Al arrastrar presionando el botón izquierdo del ratón, el trazo de vista previa se desplazará (panorámico).

Al arrastrar con Mayús y el botón izquierdo del ratón presionado, o arrastrando con la rueda del ratón presionada, el trazo de vista previa se rotará. Cuando una línea está resaltada, el centro de rotación es el centro de la línea, de lo contrario, el centro de rotación es el centro de todo el programa.

Al girar la rueda del ratón, o arrastrando con el botón derecho del ratón presionado, o arrastrando con Control y presionando el botón izquierdo del ratón, el trazo de vista previa se acercará o alejará.

Al hacer clic en uno de los iconos de *Vista predefinida*, o presionando *V*, se pueden seleccionar varias vistas preestablecidas.

10.1.3.4. Área de visualización de texto

Al hacer clic izquierdo en una línea del programa, la línea se resaltará en ambas visualizaciones, tanto la gráfica como la de texto.

Cuando el programa está en ejecución, la línea que se está ejecutando actualmente es resaltada en rojo. Si el usuario no ha seleccionado ninguna línea, la pantalla de texto se desplazará automáticamente para mostrar la línea actual.

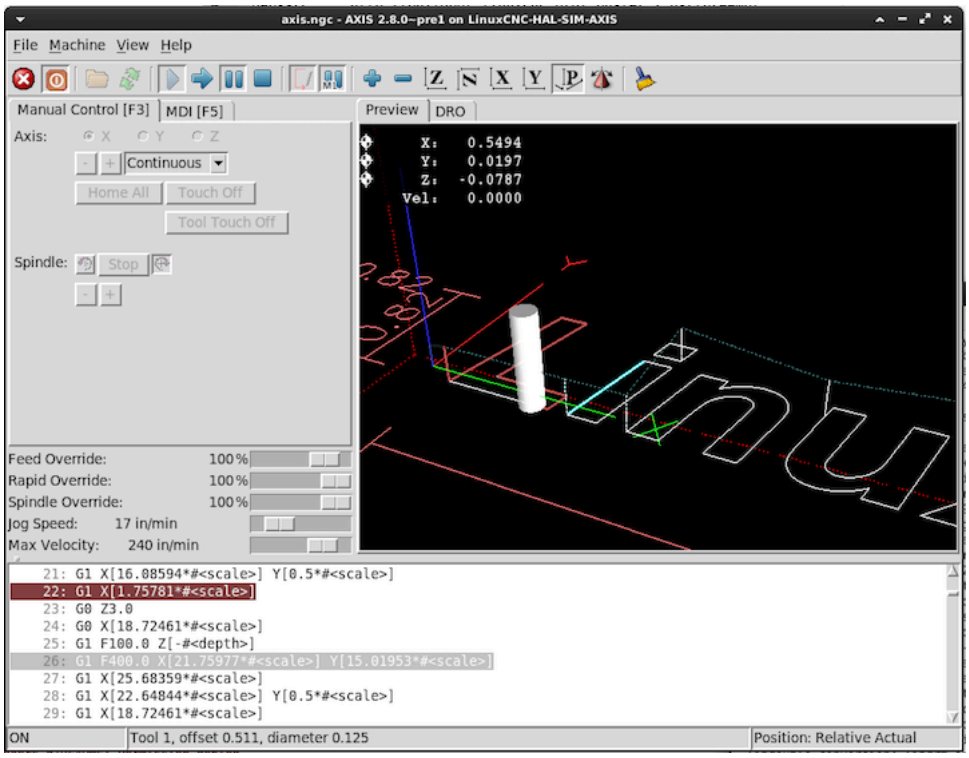


Figura 10.3: Líneas actuales y seleccionadas

10.1.3.5. Control manual

Mientras la máquina este encendida sin ejecutar un programa, los elementos de la pestaña *Control manual* se pueden utilizar para mover la máquina o controlar su husillo y el refrigerante.

Cuando la máquina no está encendida, o cuando se está ejecutando un programa, los controles manuales no están disponibles.

Muchos de los elementos descritos a continuación no son útiles en todas las máquinas. Cuando AXIS detecta que un pin en particular no está conectado en HAL, se elimina el elemento correspondiente en la pestaña Control manual. Por ejemplo, si el pin HAL *spindle.0.brake* no está conectado, entonces el botón *Freno* no aparecerá en la pantalla. Si la variable de entorno *AXIS_NO_AUTOCONFIGURE* está establecida, este comportamiento está deshabilitado y todos los elementos aparecerán.

El grupo Ejes *Ejes* permite mover manualmente la máquina. Esta acción se conoce como *trotar*. Primero, seleccione el eje a mover haciendo clic en él. Luego, haga clic y manténga presionado el botón + o - dependiendo de la dirección de movimiento deseada. Los primeros cuatro ejes también pueden ser movidos por las teclas de flecha (X e Y), teclas Pág. arriba y Pág. abajo (Z), y las teclas [y] (A).

Si se selecciona *Continuo*, el movimiento continuará mientras se presiona el botón o la tecla. Si se selecciona otro valor, la máquina se moverá exactamente la distancia mostrada cada vez que se hace clic en el botón o se presiona la tecla. Por defecto, los valores disponibles son *0.1000, 0.0100, 0.0010, 0.0001*.

Consultar la sección DISPLAY para obtener más información sobre la configuración los incrementos.

Homing (cinemáticas identidad) La configuración de archivo INI [KINS]JOINTS define el número total de articulaciones para el sistema. Una articulación puede configurarse con un interruptor de casa o con un homing *inmediato*. Las articulaciones pueden especificar una secuencia de casa que organice el orden de grupos de homing de articulaciones.

Si **todas** las articulaciones están configuradas para homing y tienen secuencias válidas para homing, el botón homing mostrará *Home Todos*. Presionar el botón *Home Todos* (o las teclas Ctrl-HOME) llevará a casa todas las articulaciones utilizando sus secuencias de casa definidas. La tecla HOME llevará a casa la articulación correspondiente al eje seleccionado actual, incluso si no esta definida una secuencia de homing.

Si no todos los ejes tienen una secuencia de casa válida, el botón mostrará *Home Eje* y llevará a casa solo a la articulación del eje seleccionado actualmente. Cada eje debe ser seleccionado y llevado a casa por separado.

El menú desplegable Máquina/Homing proporciona un método alterno para llevar a casa los ejes. El menú desplegable Máquina/Unhoming proporciona los medios para quitar de casa a los ejes.

Si la máquina no tiene interruptores de casa definidos en la configuración, el botón *Home* establecerá la posición actual del eje seleccionado como la posición absoluta 0 para ese eje y activará el bit *is-homed* para ese eje.

Consultar el capítulo de Configuración de homing para obtener más información.

Homing (cinemáticas sin identidad) Operación similar a la de cinemáticas identidad, pero antes del homing, los botones radio de selección seleccionan las articulaciones por número. El botón homing mostrará *Home Todos* si todas las articulaciones están configuradas para homing y tienen secuencias de casa válidas. De lo contrario, el botón homing mostrará *Home Joint*.

Consultar el capítulo de Configuración de homing para obtener más información.

Touch off

Al presionar *Touch Off* o la tecla END, el *offset G5x* para el eje actual se cambia para que el valor del eje actual sea el valor especificado. Las expresiones se pueden ingresar usando las reglas para los programas rs274ngc, excepto que las variables no pueden ser referidas. El valor resultante se muestra como un número.

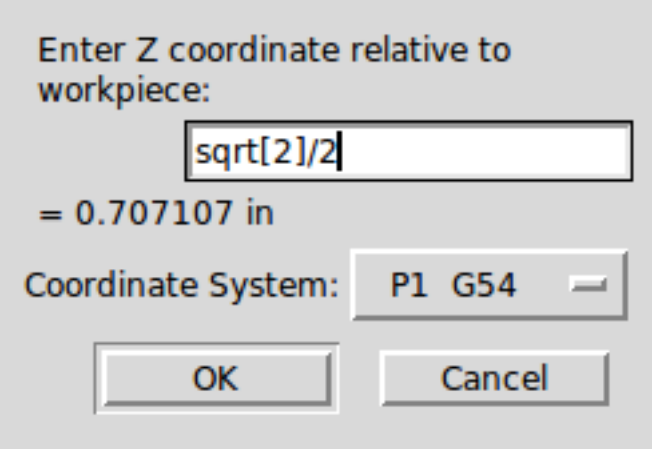


Figura 10.4: Ventana Touch off

Ver también las opciones del menú Máquina: *Touch pieza* y *Touch amarre*.

Touch off con posición real Se puede configurar un eje en el archivo INI para incorporar el valor de la posición real para el cálculo en un touch off, ya sea sumando o restando este valor. Esto es principalmente útil en máquinas que tienen un eje no motorizado como un cañón con un codificador. Cuando se habilita esta característica en un eje, la barra de título de una ventana touch off indicará **(sistema REAL)**.

Ver la sección Sección 10.1.12.11 para obtener más información.

Toque inicial de la herramienta Al presionar el botón *Touch off herramienta* cambiará la longitud y los offsets de la herramienta cargada actualmente para que la posición de la punta de la herramienta actual coincida con la coordenada introducida.

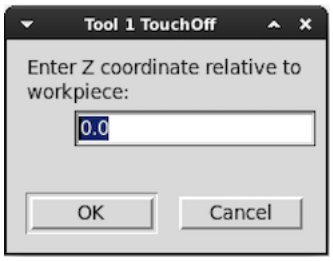


Figura 10.5: Ventana touch off herramienta

Consultar también las opciones *Touch off herramienta - pieza* y *Touch off herramienta - amarre* en el menú Máquina.

Anulación de límites Pulsando Anulación de límites se permitirá trotar temporalmente en la máquina más allá de un interruptor de límite físico. Esta casilla solo está disponible cuando se dispara un interruptor de límite. La anulación desaparece después de un trote. Si el eje está configurado con interruptores de límite positivo y negativo separados, LinuxCNC permitirá el trote solo en la dirección correcta. *La anulación de límites no permite un trote más allá de un límite de software. La única manera de deshabilitar un límite de software en un eje es con unhome.*

El grupo Husillo Los botones de la primera fila seleccionan la dirección de giro del husillo; en sentido contrario a las agujas del reloj, detenido y en el sentido de las agujas del reloj. El sentido antihorario solo aparece si el pin *spindle.0.reverse* está en el archivo HAL (puede ser *net trick-axis spindle.0.reverse*). Los botones en la siguiente fila aumentan o disminuyen la velocidad de rotación. La casilla de verificación en la tercera fila permite que el freno del husillo sea accionado o liberado. Dependiendo de la configuración de su máquina, no todos los elementos en este grupo pueden aparecer. Presionando el botón de arranque del husillo se establece la velocidad *S* en 1.

El grupo Refrigerante Dos botones permiten encender los refrigerantes *Niebla* e *Inundación* y apagarlos. Dependiendo de la configuración de la máquina, puede que no todos los elementos de este grupo aparezcan.

10.1.3.6. MDI

MDI permite que los comandos de código G se ingresen manualmente. Cuando la máquina no está encendida, o cuando un programa está en ejecución, los controles MDI no están disponibles.

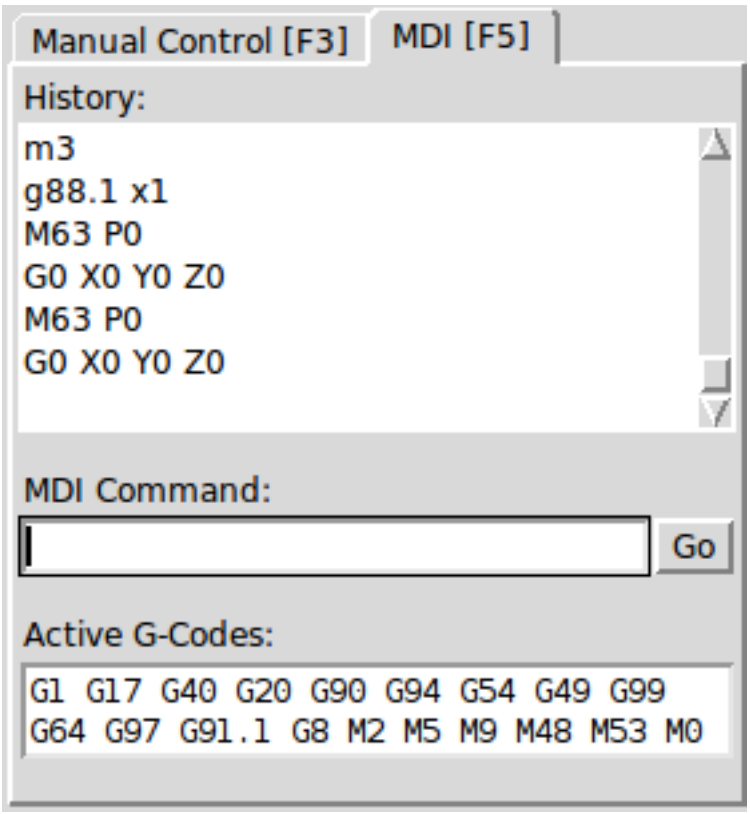


Figura 10.6: La pestaña MDI

- *Historial* - Muestra los comandos MDI que se han escrito anteriormente en esta sesión.
- *Comando MDI* - Esto permite ingresar un comando de código G para ser ejecutado. Ejecute el comando pulsando Intro o haciendo clic en *Ir*.
- *Códigos G activos* - Se muestran los *códigos modales* que están activos en el intérprete. Por ejemplo, *G54* indica que el *offset G54* se aplica a todas las coordenadas que se introduzcan. En modo Auto los Códigos G activos representan los códigos después cualquier lectura anticipada del intérprete.

10.1.3.7. Porcentaje de alimentación

Al mover este control deslizante, se puede modificar la velocidad de alimentación programada. Por ejemplo, si un programa solicita *F60* y el control deslizante se establece en 120 %, entonces la velocidad de alimentación resultante será 72.

10.1.3.8. Porcentaje de velocidad del husillo

Al mover este control deslizante, la velocidad del husillo programada puede ser modificada. Por ejemplo, si un programa solicita S8000 y el control deslizante es establecido en 80 %, entonces la velocidad del husillo resultante será 6400. Este elemento solo aparece cuando el pin HAL *spindle.0.speed-out* está conectado.

10.1.3.9. Velocidad de trote

Al mover este control deslizante, se puede modificar la velocidad de trote. Por ejemplo, si el deslizante se establece en 1 pulgada/min, entonces un trote de .01 pulgadas tardará aproximadamente .6

segundos, o 1/100 de minuto. Cerca del lado izquierdo (trotes lentos) los valores están espaciados muy cerca, mientras que cerca del lado derecho (trotes rápidos) están espaciados mucho más separados, permitiendo una amplia gama de velocidades de trote con control fino cuando sea importante.

En las máquinas con un eje giratorio, se muestra un segundo control deslizante de velocidad de trote. Este deslizante establece la velocidad de trote de los ejes giratorios (A, B y C).

10.1.3.10. Velocidad máxima

Al mover este control deslizante, se puede establecer la velocidad máxima. Esto limita la velocidad máxima para todos los movimientos programados, excepto en movimientos sincronizados con el husillo.

10.1.4. Controles del teclado

Casi todas las acciones en AXIS se pueden realizar con el teclado. La lista completa de atajos de teclado se puede encontrar en la Referencia rápida de AXIS, que se puede mostrar seleccionando Ayuda > Referencia rápida. Muchos de los accesos directos no están disponibles cuando se está en modo MDI.

10.1.4.1. Teclas de porcentaje de alimentación

nota
En el teclado *Español de España*, el simbolo ” ’ ” se refiere al signo de acentuacion junto a la tecla ”P”. Este signo precisa doble pulsacion de dicha tecla.

Las teclas de porcentaje de alimentación se comportan de manera diferente cuando están en modo manual. Las teclas ’,1,2,3,4,5,6,7 y 8 seleccionarán un eje si está programado. Si se tienen 3 ejes ’ seleccionará el eje 0, 1 seleccionará el eje 1, y 2 seleccionará eje 2. El resto de las teclas numéricas establecerán el porcentaje de alimentación. Al ejecutar un programa, ’,1,2,3,4,5,6,7,8,9,0 establecerán el porcentaje de alimentación a 0 % - 100 %.

Los métodos abreviados de teclado más utilizados se muestran en la siguiente tabla:

Cuadro 10.1: Atajos de teclado más comunes

Teclas	Acción	Modo
F1	Alterna parada de emergencia	Cualquiera
F2	Enciende/apaga la máquina	Cualquiera
`, 1 .. 9, 0	Ajusta el porcentaje de alimentación de 0 % a 100 %	Varía
X, `	Activa el primer eje	Manual
Y, 1	Activa el segundo eje	Manual
Z, 2	Activa el tercer eje	Manual
A, 3	Activa el cuarto eje	Manual
I	Selecciona incremento de trote	Manual
C	Trote continuo	Manual

Cuadro 10.1: (continued)

Teclas	Acción	Modo
Control-Home	Realiza secuencia de homing	Manual
Fin	Touch off: Establece offset G5x en el eje activo	Manual
Izquierda, Derecha	Trota el primer eje	Manual
Arriba, Abajo	Trota el segundo eje	Manual
Pág. arriba, Pág. abajo	Trota el tercer eje	Manual
[,]	Trota el cuarto eje	Manual
O	Abrir archivo	Manual
Control-R	Recargar archivo	Manual
R	Ejecutar archivo	Manual
P	Pausar ejecución	Auto
S	Reanuda ejecución	Auto
ESC	Detiene la ejecución	Auto
Control-K	Limpiar trazo posterior	Auto/Manual
V	Cambia cíclicamente entre las vistas preestablecidas	Auto/Manual
Mayús. Izq, Der	Movimiento rápido eje X	Manual
Mayús.-Arriba, Abajo	Movimiento rápido eje Y	Manual
Mayús.-Pág. Arriba, Pág. Abajo	Movimiento rápido eje Z	Manual
@	Alternar Real/Ordenada	Cualquiera
#	Alternar Relativo/Máquina	Cualquiera

10.1.5. Mostrar estado de LinuxCNC (`linuxcnctop`)

AXIS incluye un programa llamado *linuxcnctop* que muestra algunos de los detalles del estado de LinuxCNC. Puedes ejecutar este programa mediante Máquina > Mostrar estado de LinuxCNC

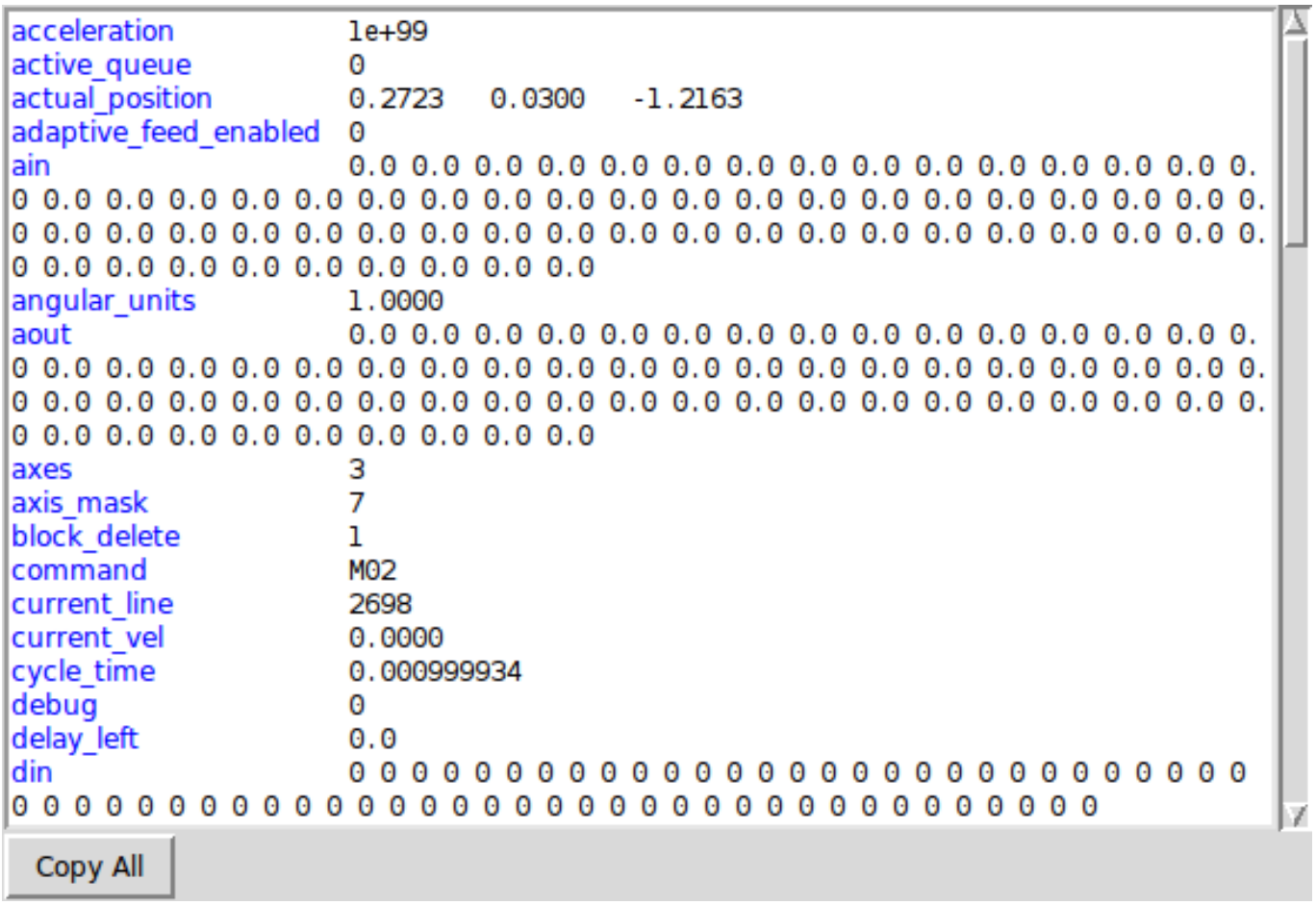


Figura 10.7: Ventana de estado de LinuxCNC

El nombre de cada elemento se muestra en la columna izquierda. El valor actual se muestra en la columna derecha. Si el valor ha cambiado recientemente, se muestra sobre un fondo rojo.

10.1.6. Interfaz MDI

AXIS incluye un programa llamado `mdi` que permite la entrada en modo texto de comandos MDI a una sesión de LinuxCNC en ejecución. Este programa se puede ejecutar directamente desde la línea de comandos abriendo una terminal y escribiendo:

```
mdi
```

Una vez que se está ejecutando, muestra el mensaje *MDI>*. Cuando se ingresa una linea en blanco se muestra la posición actual de la máquina. Cuando se ingresa un comando, se envía a LinuxCNC para ser ejecutado.

Esta es una sesión de muestra de mdi:

```
$ mdi
MDI>
(0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
MDI> G1 F5 X1
MDI>
(0.5928500000000374, 0.0, 0.0, 0.0, 0.0, 0.0)
MDI>
```

```
(1.0000000000000639, 0.0, 0.0, 0.0, 0.0, 0.0)
```

10.1.7. axis-remote

AXIS incluye un programa llamado *axis-remote* que puede enviar ciertos comandos a un AXIS en ejecución. Los comandos disponibles se muestran ejecutando *axis-remote --help*, que incluyen el de verificación de ejecución de AXIS (*--ping*), el de carga de un archivo por nombre, recarga del archivo cargado actualmente (*--reload*) y el de salir de AXIS (*--quit*).

10.1.8. Cambio manual de herramienta

LinuxCNC incluye un componente HAL en tiempo no-real llamado *hal_manualtoolchange*, que muestra una ventana que le indica qué herramienta se espera cuando se emite el comando *M6*. Después de presionar el botón Aceptar, la ejecución del programa continuará.

El componente hal_manualtoolchange incluye un pin HAL para un botón que se puede conectar a un botón físico para completar el cambio de herramienta y eliminar la ventana de captura (hal_manualtoolchang

El archivo de configuración de HAL *lib/hallib/axis_manualtoolchange.hal* muestra los comandos HAL necesarios para usar este componente.

hal_manualtoolchange se puede usar incluso cuando AXIS no se usa como GUI. Este componente es más útil si tiene herramientas predefinidas y usa la tabla de herramientas.

---

nota
Nota importante: los rápidos no se mostrarán en la vista previa después de emitir un T<n> hasta el siguiente movimiento de alimentación después de M6. Esto puede ser muy confuso para la mayoría de los usuarios. Para desactivar esta función para el cambio de herramienta actual, programe un G1 sin movimiento después de T<n>.

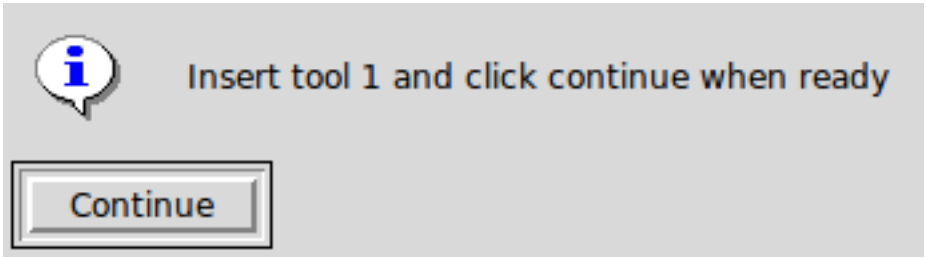


Figura 10.8: Ventana de cambio de herramientas manual

10.1.9. Módulos Python

AXIS incluye varios módulos Python que pueden ser útiles para otros. Para más información sobre uno de estos módulos, use *pydoc <nombre del módulo>* o lea el código fuente. Estos módulos incluyen:

- *emc* proporciona acceso a los canales de comando, estado y error de LinuxCNC
- *gcode* proporciona acceso al intérprete rs274ngc
- *rs274* proporciona herramientas adicionales para trabajar con archivos rs274ngc

- *hal* permite la creación de componentes HAL en tiempo no-real escritos en Python
- *_togl* proporciona un widget OpenGL que puede usarse en aplicaciones Tkinter

Para utilizar estos módulos en sus propios scripts, debe asegurarse que el directorio donde residen esté en la ruta de módulos de Python. Cuando se ejecuta una versión instalada de LinuxCNC, esto debería suceder automáticamente. Cuando ejecutando una RIP, esto se puede hacer mediante el guion *scripts/rip-environment*.

10.1.10. AXIS en modo torno

Incluyendo la línea *LATHE = 1* en la sección [DISPLAY] del archivo INI, AXIS selecciona el modo de torno. El eje *Y* no se muestra en las lecturas de coordenadas, la vista se cambia mostrando el eje Z extendido hacia la derecha y el eje X extendido hacia la parte inferior de la pantalla, y se quitan varios controles (como los de vistas preestablecidas). Las lecturas de coordenadas para X se reemplazan con diámetro y radio.

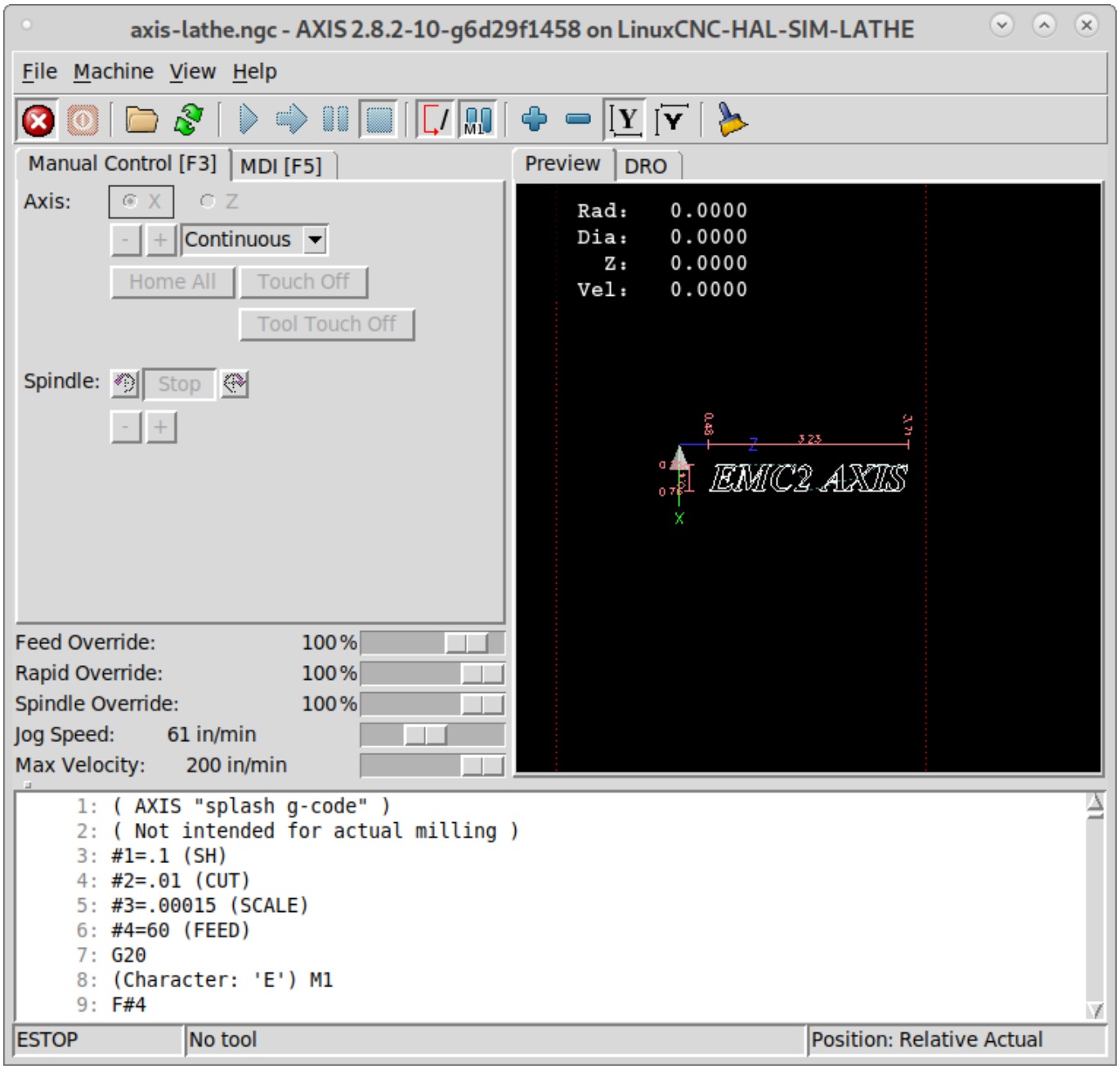


Figura 10.9: AXIS en modo torno

Al presionar *V* se hace zoom para mostrar el archivo completo, si hay uno cargado.

En el modo de torno, se muestra la forma de la herramienta cargada (si existe).

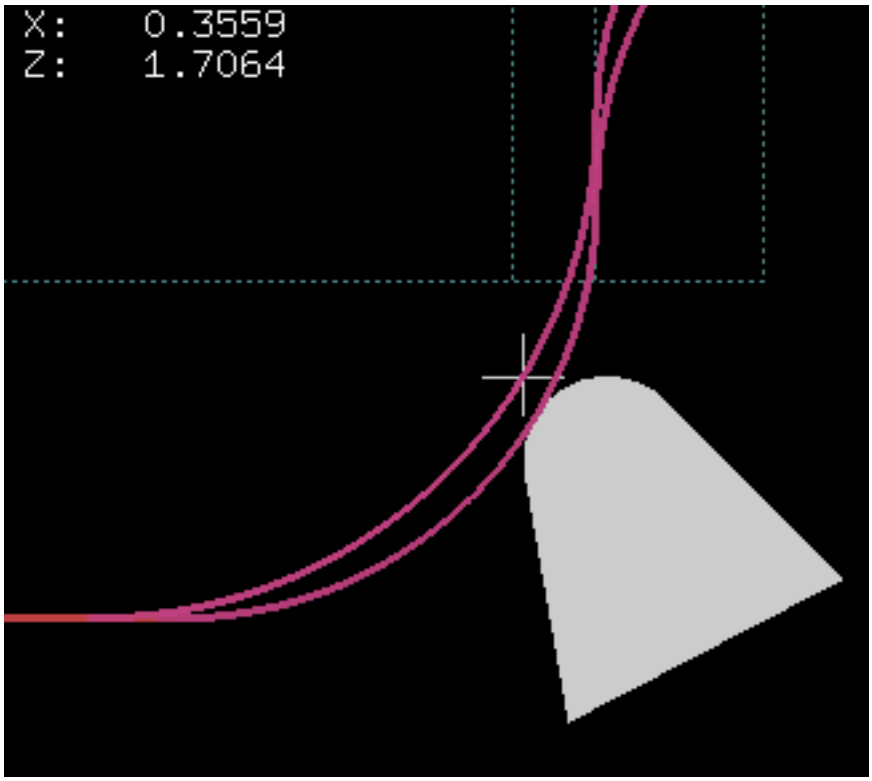


Figura 10.10: Forma de herramienta de torno

Para cambiar la visualización a un torno de herramienta trasera necesitas tener las dos configuraciones *LATHE = 1* y *BACK_TOOL_LATHE = 1* en la sección [DISPLAY]. Esto invertirá la vista y colocará la herramienta en la parte trasera del eje Z.

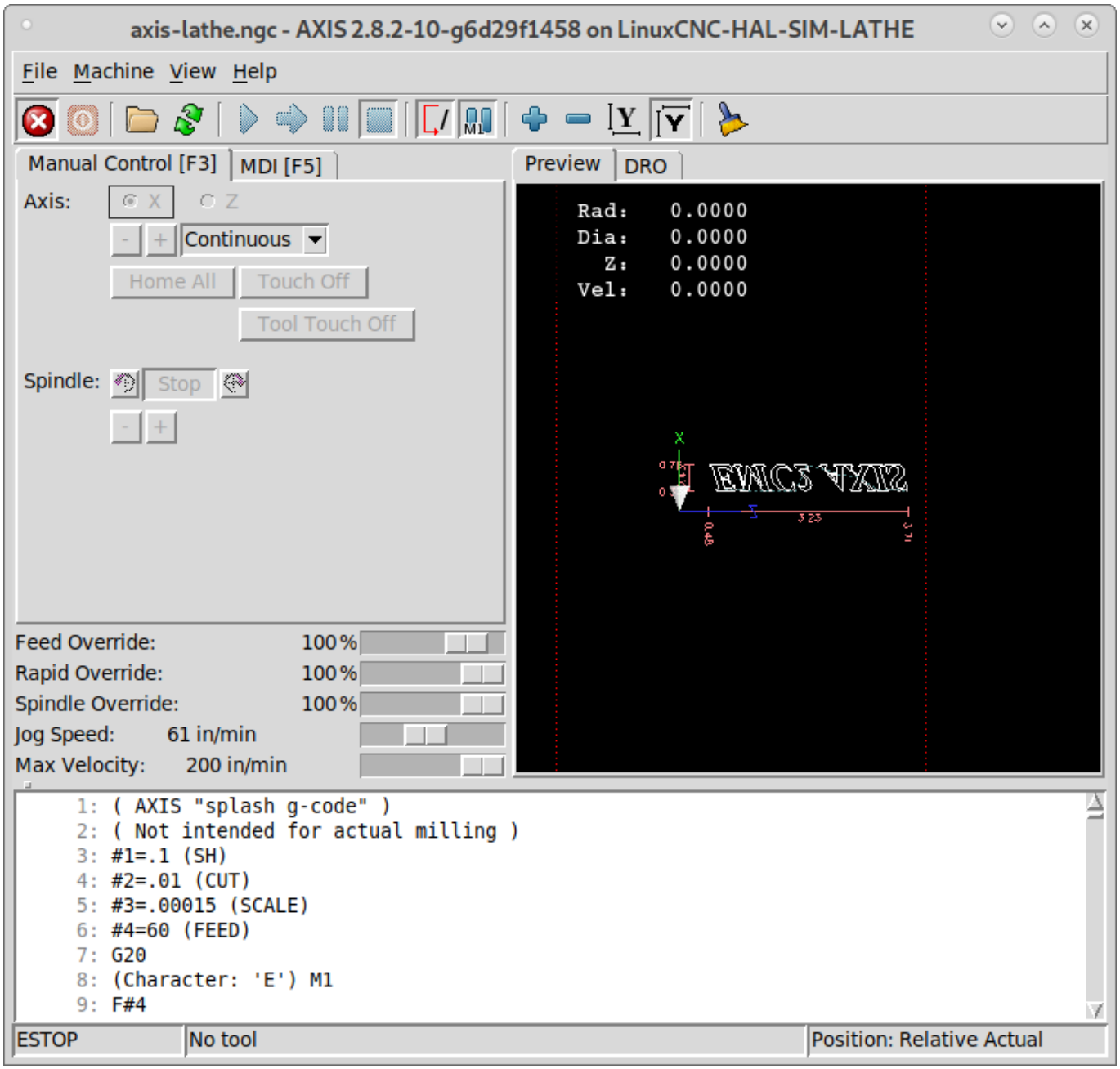


Figura 10.11: Forma de herramienta trasera de torno

10.1.11. Usando AXIS en el modo de corte de espuma

Incluyendo la línea *FOAM = 1* En la sección [DISPLAY] del archivo INI, AXIS selecciona el modo de corte de espuma. En la vista previa del programa, los movimientos XY se muestran en un plano, y los movimientos UV en otro. En el trazo en vivo, se dibujan líneas entre los puntos correspondientes en el plano XY y el plano UV. Los comentarios especiales (XY_Z_POS) y (UV_Z_POS) establecen las coordenadas Z de estos planos, que por defecto son 0 y 1.5 unidades de máquina.

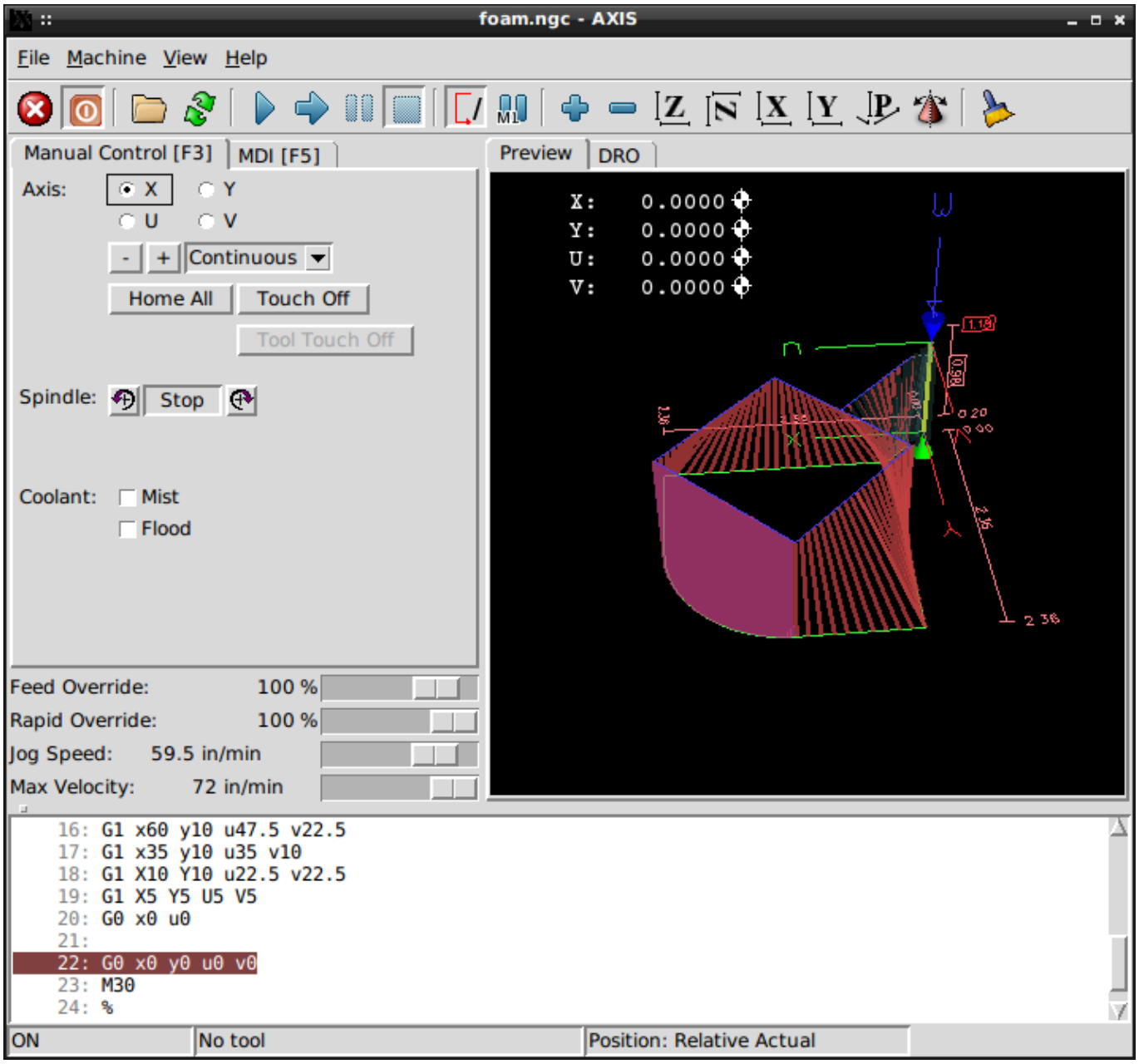


Figura 10.12: Modo de corte de espuma

10.1.12. Configuración avanzada

Cuando se inicia AXIS, se crean los pines HAL para la GUI y se ejecuta el archivo HAL nombrado *[HAL]POSTGUI_HALFILE=<nombre de archivo>* en el archivo INI. Típicamente *<nombre de archivo>* sería el nombre base de configuraciones + *_postgui* + *.hal*, p.ej. *torno_postgui.hal*, pero puede ser cualquier nombre válido. Estos comandos se ejecutan después de construir la pantalla, garantizando que los pines HAL del widget estén disponibles. Se pueden tener varias líneas de *POSTGUI_HALFILE=<nombre de archivo>* en el INI; cada uno será ejecutado uno tras de otro en el orden en que aparezcan.

Para obtener más información sobre la configuración del archivo INI que puede cambiar la forma en que AXIS trabaja, ver la Sección DISPLAY del capítulo de configuración INI.

10.1.12.1. Filtros de programa

AXIS tiene la capacidad de enviar archivos cargados a través de un *programa filtro*. Este filtro puede realizar cualquier tarea deseada: algo tan simple como asegurarse el archivo termina con *M2*, o algo tan complicado como generar código G a partir de una imagen.

La sección *[FILTER]* del archivo INI controla cómo funcionan los filtros. Primero, para cada tipo de archivo, se escribe una línea *PROGRAM_EXTENSION*. Luego, se especifica el programa a ejecutar para cada tipo de archivo. Este programa recibe el nombre del archivo de entrada como su primer argumento, y debe escribir código rs274ngc en la salida estándar. Esta salida es lo que se mostrará en el área de texto, se previsualizará en el área de visualización y será ejecutado por LinuxCNC con *Ejecutar*. Las líneas siguientes agregan soporte para el convertidor *image-to-gcode* incluido en LinuxCNC:

```
[FILTER]
PROGRAM_EXTENSION = .png,.gif Imagen de profundidad en escala de grises
png = image-to-gcode
gif = image-to-gcode
```

También es posible especificar un intérprete:

```
PROGRAM_EXTENSION = .py Script Python
py = python
```

De esta manera, cualquier script de Python se puede abrir, y su salida es tratada como código G. Un ejemplo de este script está disponible en *nc_files/holecircle.py*. Este script crea código G para perforar una serie de agujeros a lo largo de la circunferencia de un círculo.

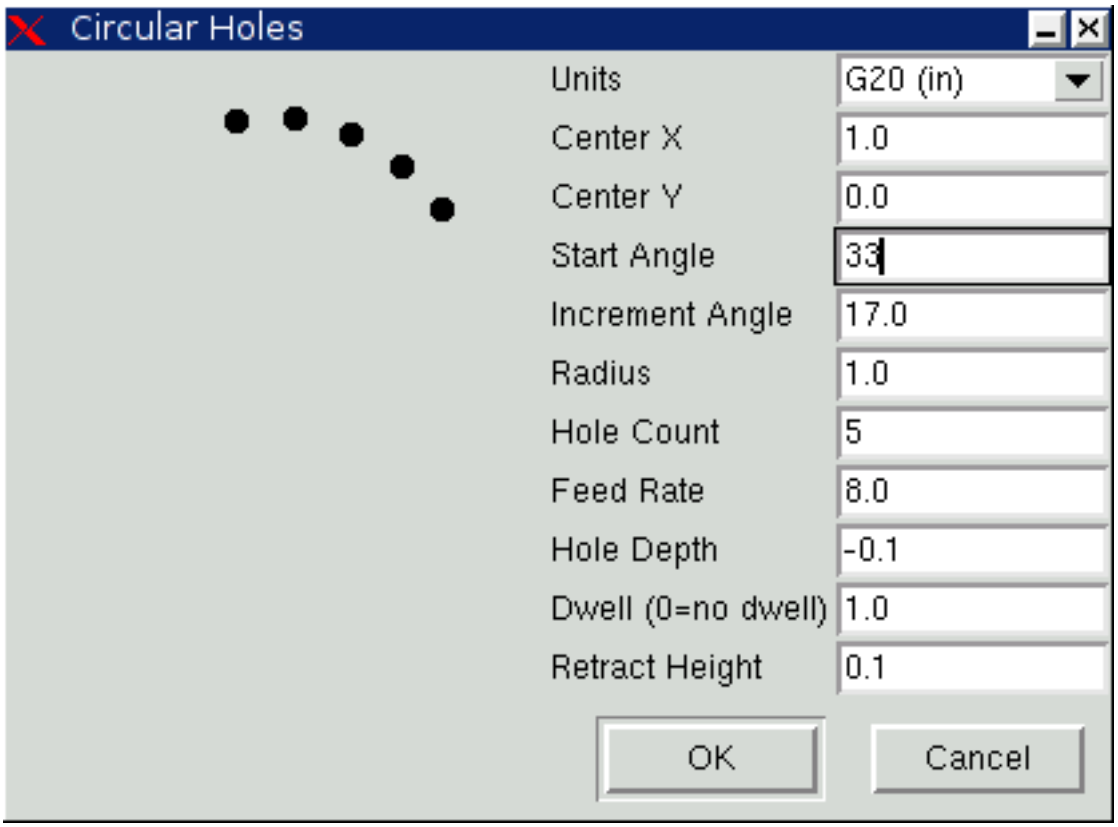


Figura 10.13: Agujeros circulares

Si la variable de entorno AXIS_PROGRESS_BAR está establecida, entonces las líneas escriben al stderr del formulario

```
FILTER_PROGRESS=%d
```

establecerá la barra de progreso de AXIS en el porcentaje dado. Esta característica debe ser utilizada por cualquier filtro que se ejecute durante mucho tiempo.

10.1.12.2. La base de datos de recursos X

Los colores de la mayoría de los elementos de la interfaz de usuario AXIS pueden ser personalizados a través de la base de datos de recursos X. El archivo de ejemplo *axis_light_background* cambia los colores de la ventana de trazo posterior a un esquema *líneas oscuras en fondo blanco*, y también sirve como una referencia para elementos configurables en el área de visualización. El archivo de ejemplo *axis_big_dro* cambia la posición de lectura a una fuente de tamaño más grande. Para utilizar estos archivos:

```
xrdb -merge /usr/share/doc/emc2/axis_light_background

xrdb -merge /usr/share/doc/emc2/axis_big_dro
```

Para obtener información sobre los otros elementos que se pueden configurar en aplicaciones Tk, ver las páginas del manual de Tk.

Dado que los entornos de escritorio modernos hacen algunas configuraciones automáticamente en la base de datos de recursos X que afectan adversamente a AXIS, estos ajustes son ignorados por defecto. Para hacer que los elementos de la base de datos de recursos X anulen los valores predeterminados de AXIS, incluya la siguiente línea en sus Recursos X:

```
*AXIS*optionLevel: widgetDefault
```

esto hace que las opciones integradas se creen en el nivel de opción *widgetDefault*, de modo que X Resources (que son nivel *userDefault*) pueda anularlas.

10.1.12.3. Perilla de trote

Para mejorar la interacción de AXIS con una perilla de trote, el eje actualmente seleccionado en la GUI también se reporta en un *pin HAL* con un nombre como *axisui.jog.x*. A excepción de un corto tiempo después de cambiar el eje actual, solo uno de estos pines es Verdadero a la vez, y el resto permanecen en Falso.

Después de que AXIS ha creado estos *pines HAL*, ejecuta el archivo HAL con: [HAL]POSTGUI_HALFILE. Lo que difiere de [HAL]HALFILE, el cual solo puede usarse una vez.

10.1.12.4. ~/.axisrc

Si existe, el contenido de `~/.axisrc` se ejecuta como código fuente Python justo antes de mostrar la interfaz gráfica de usuario de AXIS. Los detalles de lo que se puede escribir en `~/.axisrc` están sujetos a cambios durante el ciclo de desarrollo.

Lo siguiente agrega Control-Q como método abreviado de teclado para Salir.

Ejemplo de archivo .axisrc

```
root_window.bind("<Control-q>", "destroy .")
help2.append(("Control-Q", "Quit"))
```

Lo siguiente detiene el cuadro de diálogo "¿Realmente desea salir?".

```
root_window.tk.call("wm","protocol",".","WM_DELETE_WINDOW","destroy .")
```

10.1.12.5. USER_COMMAND_FILE

Puede especificarse un archivo Python específico de configuración en un archivo INI configurando *[DISLAY]USER_COMMAND_FILE=filename.py*. Al igual que un archivo `~/.axisrc`, este archivo corre justo antes de que se muestre la GUI de AXIS. Este archivo es específico de una configuración de archivo INI, no del directorio de inicio del usuario.

10.1.12.6. user_live_update()

La GUI AXIS incluye una función no-op (marcador de posición) llamada *user_live_update()* que se ejecuta al final de la función update() de su clase LivePlotter. Esta función puede ser implementada dentro de scripts Python en `~/.axisrc` o en *[DISPLAY]USER_COMMAND_FILE* para realizar acciones personalizadas periódicas. Los detalles de lo que puede lograrse con esta función dependerán de la implementación de la GUI AXIS y sujetos a cambios durante el ciclo de desarrollo.

10.1.12.7. user_hal_pins()

La GUI AXIS incluye una función de no-operación (marcador de posición) llamada *user_hal_pins()*.
Se ejecuta justo después de llamar al archivo .axisrc y justo antes inicializar cualquier panel GladeVCP / pestaña incrustada.
Esta función puede implementarse dentro de un script Python `~/.axisrc` o *[DISPLAY]USER_COMMAND_FI*
para hacer que los pines HAL personalizados usen el prefijo *axisui.*.
Use *comp* como la referencia de instancia del componente HAL.
Se llama a HAL comp.ready() justo después de regresar de esta función.

10.1.12.8. Editor externo

Las opciones de menú Archivo > Editar ... y Archivo > Editar tabla de herramientas ... estarán disponibles después de definir el editor en la sección INI [DISPLAY]. Los valores útiles incluyen `EDITOR=gedit` y `EDITOR=gnome-terminal -e vim`. Para obtener más información, ver la Sección display del capítulo de Configuración INI.

10.1.12.9. Panel de control virtual

AXIS puede mostrar un panel de control virtual personalizado en la columna derecha o en la última fila. Además, se pueden mostrar uno o más paneles como pestañas incrustadas. Se pueden programar botones, indicadores, visualizadores de datos y más cosas. Para más información, consultar los capítulos PyVCP y GladeVCP.

10.1.12.10. Control de vista previa

Se pueden insertar comentarios especiales en el archivo de código G para controlar cómo se comporta la vista previa de AXIS. En caso de querer limitar el dibujo de la vista previa, utilizar estos comentarios especiales. Cualquier cosa entre (AXIS,hide) y (AXIS,show) no se dibujará durante la vista previa. (AXIS,hide) y (AXIS,show) deben usarse en pares, con (AXIS,hide) primero. Cualquier cosa después de (AXIS,stop) no se dibujará durante la vista previa.

Estos comentarios son útiles para despejar la visualización de vista previa (por ejemplo, mientras se depura un archivo de código G más grande, se puede deshabilitar la vista previa de ciertas partes que ya están trabajando bien).

- (AXIS,hide) Detiene la vista previa (debe estar primero)

- (AXIS,show) Reanuda la vista previa (debe seguir un hide)
- (AXIS,stop) Detiene la vista previa desde aquí hasta el final del archivo.
- (AXIS,notify,el_texto) Muestra el_texto como una visualización de información

Esta visualización puede ser útil en la vista previa de AXIS cuando no se muestran los comentarios (debug,mensaje).

10.1.12.11. Touch off usando posición real

La característica de touch off puede incorporar opcionalmente el valor de la posición real del eje en los cálculos del offset. Esto se usa primordialmente en casos donde un eje no motorizado, como el del cañón en una máquina fresadora, proporciona retroalimentación a LinuxCNC por medio de un codificador, pero no sin haber motor para controlar su movimiento. Esto permite a AXIS proporcionar una visualización DRO para tal eje con capacidad touch off funcional.

Esta característica se activa en un eje alterando la sección *[AXIS_x]* apropiada en el archivo INI. Agregando una opción llamada *TOUCHOFF_ACTUAL* y estableciendo el valor a *PLUS* o *MINUS* dependiendo de cómo se quiera aplicar la posición actual al offset.

Ejemplo:

```
[AXIS_Z]
TOUCHOFF_ACTUAL = MINUS
```

Ordinariamente, solo se usa la posición ordenada de un eje para establecer este offset, lo que significa que no funciona como debería por que a los ejes no motorizados nunca se les ordena moverse, y por lo tanto su posición ordenada siempre es 0.

Touch off envía un comando *G10 L20* al MDI para establecer el nuevo valor de offset. El valor aplicado es normalmente solo el valor introducido en la caja de diálogo. Cuando se activa esta característica, sumará o restará el valor de la posición actual al valor introducido en el diálogo, dependiendo de cómo se haya configurado.

10.1.13. Axisui

Para mejorar la interacción de AXIS con perillas de trote físicas, el eje actualmente seleccionado en la GUI también se reporta en un pin con un nombre como *axisui.jog.x*. Uno de estos pines es verdadero y el resto son falso. Están diseñados para controlar los pines que habilitan el movimiento de trote.

Pines axisui AXIS tiene pines HAL para indicar qué botón de selección de trote está seleccionado en el pestaña *Control manual*.

```
Tipo Dir Nombre
bit  OUT axisui.jog.x
bit  OUT axisui.jog.y
bit  OUT axisui.jog.z
bit  OUT axisui.jog.a
bit  OUT axisui.jog.b
bit  OUT axisui.jog.c
bit  OUT axisui.jog.u
bit  OUT axisui.jog.v
bit  OUT axisui.jog.w
```

AXIS tiene un pin HAL para indicar el incremento de trote seleccionado en *Pestaña manual*.

```
Tipo  Dir Nombre
float OUT axisui.jog.increment
```

AXIS tiene un pin de salida HAL para indicar que ha ocurrido un aborto. El pin *axisui.abort* será *TRUE* y regresará a *FALSE* después de 0.3ms.

```
Tipo  Dir    Nombre
bit   OUT    axisui.abort
```

AXIS tiene un pin de salida HAL para indicar que ha ocurrido un error. El pin *axisui.error* permanecerá *TRUE* hasta que se hayan confirmado todas las notificaciones de error.

```
Tipo  Dir    Nombre
bit   OUT    axisui.error
```

AXIS tiene pines de entrada HAL para borrar las notificaciones emergentes de errores e información.

```
Tipo  Dir Nombre
bit   IN  axisui.notifications-clear
bit   IN  axisui.notifications-clear-error
bit   IN  axisui.notifications-clear-info
```

AXIS tiene un pin de entrada HAL que deshabilita/habilita la función *Pausa/Reanudar*.

```
Tipo  Dir Nombre
bit   IN  axisui.resume-inhibit
```

10.1.14. Sugerencias de personalización de AXIS

AXIS es un código bastante grande y difícil de penetrar. Esto es útil para mantener el código estable pero dificulta la personalización.
Aquí mostraremos fragmentos de código para modificar comportamientos o elementos visuales de la pantalla. Hay que tener en cuenta que el código interno de AXIS puede cambiar de vez en cuando. No se garantiza que estos fragmentos continúen funcionando - pueden necesitar ajustes.

10.1.14.1. La función de actualización

Hay una función en AXIS nombrada user_live_update que se llama cada vez que AXIS se actualiza solo. Puedes usar esto para actualizar tus propias funciones.

```
# función de actualización continua
def user_live_update():
    print('impreso en cada actualización...')
```

10.1.14.2. Deshabilitar el cuadro de diálogo Cerrar

```
# deshabilita el diálogo de confirmación de cierre
root_window.tk.call("wm","protocol",".","WM_DELETE_WINDOW","destroy .")
```

10.1.14.3. Cambiar la fuente del texto

```
# cambiar la fuente

font = 'sans 11'
fname,fsize = font.split()
root_window.tk.call('font','configure','TkDefaultFont','-family',fname,'-size',fsize)
```

```
# rehace el texto en pestañas para que cambien al tamaño de la nueva fuente predeterminada

root_window.tk.call('.pane.top.tabs','itemconfigure','manual','-text',' Manual - F3 ')
root_window.tk.call('.pane.top.tabs','itemconfigure','mdi','-text',' MDI - F5 ')
root_window.tk.call('.pane.top.right','itemconfigure','preview','-text',' Preview ')
root_window.tk.call('.pane.top.right','itemconfigure','numbers','-text',' DRO ')


# la fuente código G es independiente

root_window.tk.call('.pane.bottom.t.text','configure','-foreground','blue')
#root_window.tk.call('.pane.bottom.t.text','configure','-foreground','blue','-font',font)
#root_window.tk.call('.pane.bottom.t.text','configure','-foreground','blue','-font',font,'- ↩
    height','12')
```

10.1.14.4. Modificar velocidad rápida con atajos de teclado

```
# use control + ‘ o 1-0 como atajos de teclado para acelerar y mantener ‘ o 1-0 para avance
# también agrega texto a la referencia rápida en la ayuda

help1.insert(10,("Control+ ‘,1..9,0", _("Establecer ajuste de rápidos de 0% a 100%")),)

root_window.bind('<Control-Key-quoteleft>',lambda event: set_rapidrate(0))
root_window.bind('<Control-Key-1>',lambda event: set_rapidrate(10))
root_window.bind('<Control-Key-2>',lambda event: set_rapidrate(20))
root_window.bind('<Control-Key-3>',lambda event: set_rapidrate(30))
root_window.bind('<Control-Key-4>',lambda event: set_rapidrate(40))
root_window.bind('<Control-Key-5>',lambda event: set_rapidrate(50))
root_window.bind('<Control-Key-6>',lambda event: set_rapidrate(60))
root_window.bind('<Control-Key-7>',lambda event: set_rapidrate(70))
root_window.bind('<Control-Key-8>',lambda event: set_rapidrate(80))
root_window.bind('<Control-Key-9>',lambda event: set_rapidrate(90))
root_window.bind('<Control-Key-0>',lambda event: set_rapidrate(100))
root_window.bind('<Key-quoteleft>',lambda event: set_feedrate(0))
root_window.bind('<Key-1>',lambda event: set_feedrate(10))
root_window.bind('<Key-2>',lambda event: set_feedrate(20))
root_window.bind('<Key-3>',lambda event: set_feedrate(30))
root_window.bind('<Key-4>',lambda event: set_feedrate(40))
root_window.bind('<Key-5>',lambda event: set_feedrate(50))
root_window.bind('<Key-6>',lambda event: set_feedrate(60))
root_window.bind('<Key-7>',lambda event: set_feedrate(70))
root_window.bind('<Key-8>',lambda event: set_feedrate(80))
root_window.bind('<Key-9>',lambda event: set_feedrate(90))
root_window.bind('<Key-0>',lambda event: set_feedrate(100))
```

10.1.14.5. Leer el archivo INI

```
# leer un elemento del archivo INI
machine = inifile.find('EMC','MACHINE')
print('nombre de máquina =',machine)
```

10.1.14.6. Lectura del estado de LinuxCNC

```
# El estado de LinuxCNC se puede leer desde s.
print(s.actual_position)
print(s.paused)
```

10.1.14.7. Cambiar la vista actual

```
# establecer la vista de la vista previa
# las vistas válidas son view_x view_y view_y2 view_z view_z2 view_p
command.set_view_z()
```

10.1.14.8. Crear nuevos pines AXISUI HAL

```
def user_hal_pins():
    comp.newpin('my-new-in-pin', hal.HAL_BIT, hal.HAL_IN)
    comp.ready()
```

10.1.14.9. Crear nuevos componentes y pines HAL

```
# crear un componente

mycomp = hal.component('my_component')
mycomp.newpin('idle-led',hal.HAL_BIT,hal.HAL_IN)
mycomp.newpin('pause-led',hal.HAL_BIT,hal.HAL_IN)
mycomp.ready()

# conectar pines

hal.new_sig('idle-led',hal.HAL_BIT)
hal.connect('halui.program.is-idle','idle-led')
hal.connect('my_component.idle-led','idle-led')

# establecer un pin

hal.set_p('my_component.pause-led','1')

# obtener un pin de rama 2,8+

value = hal.get_value('halui.program.is-idle')
print 'value is a',type(value),'value of',value
```

10.1.14.10. Cambiar pestañas con pines HAL

```
# los pines HAL de un panel GladeVCP no estarán listos cuando se ejecute user_live_update
# para leerlos necesita ponerlos en un bloque try/except

# el siguiente ejemplo supone 5 botones HAL en un panel GladeVCP utilizado para cambiar
# las pestañas en la pantalla AXIS.
# los nombres de los botones son 'manual-tab', 'mdi-tab', 'preview-tab', 'dro-tab', 'user0- ↩
    tab'
# la pestaña user_0, si existe, sería la primera pestaña GladeVCP incrustada

# para la rama LinuxCNC 2.8+
```

```
def user_live_update():
    try:
        if hal.get_value('gladevcp.manual-tab'):
            root_window.tk.call('.pane.top.tabs','raise','manual')
        elif hal.get_value('gladevcp.mdi-tab'):
            root_window.tk.call('.pane.top.tabs','raise','mdi')
        elif hal.get_value('gladevcp.preview-tab'):
            root_window.tk.call('.pane.top.right','raise','preview')
        elif hal.get_value('gladevcp.numbers-tab'):
            root_window.tk.call('.pane.top.right','raise','numbers')
        elif hal.get_value('gladevcp.user0-tab'):
            root_window.tk.call('.pane.top.right','raise','user_0')
    except:
        pass
```

10.1.14.11. Agregar un botón para ir a casa

```
def goto_home(axis):
    if s.interp_state == linuxcnc.INTERP_IDLE:
        home = inifile.find('JOINT_' + str(inifile.find('TRAJ', 'COORDINATES').upper(). ↩
            index(axis)), 'HOME')
        mode = s.task_mode
        if s.task_mode != linuxcnc.MODE_MDI:
            c.mode(linuxcnc.MODE_MDI)
        c.mdi('G53 G0 ' + axis + home)

# hacer un botón para llevar a casa el eje Y
root_window.tk.call('button','.pane.top.tabs.fmanual.homey','-text','Home Y','-command',' ↩
    goto_home Y','-height','2')

# colocar el botón
root_window.tk.call('grid','.pane.top.tabs.fmanual.homey','-column','1','-row','7','- ↩
    columnspan','2','-padx','4','-sticky','w')

# cualquier función llamada desde Tcl debe agregarse a TclCommands
TclCommands.goto_home = goto_home
commands = TclCommands(root_window)
```

10.1.14.12. Agregar botón al marco manual

```
# crea un nuevo botón y lo pone en el marco manual

root_window.tk.call('button','.pane.top.tabs.fmanual.mybutton','-text','My Button','- ↩
    command','mybutton_clicked','-height','2')
root_window.tk.call('grid','.pane.top.tabs.fmanual.mybutton','-column','1','-row','6','- ↩
    columnspan','2','-padx','4','-sticky','w')

# lo anterior envía el comando "mybutton_clicked" cuando se hace clic
# otras opciones son vincular un comando de pulsar o soltar (o ambos) al botón
# estos pueden ser adicionales o en lugar del comando seleccionado
# si en lugar de eliminar '-command', 'mybutton_clicked', de la primera línea

# Button-1 = botón izquierdo del mouse, 2 = derecho o 3 = medio

root_window.tk.call('bind','.pane.top.tabs.fmanual.mybutton','<Button-1>','mybutton_pressed ↩
    ')
```

```
root_window.tk.call('bind','.pane.top.tabs.fmanual.mybutton','<ButtonRelease-1>',' ↩
    mybutton_released')

# funciones llamadas desde los botones

def mybutton_clicked():
    print('mybutton fue clicado')
def mybutton_pressed():
    print('mybutton fue presionado')
def mybutton_released():
    print('mybutton fue liberado')

# cualquier función llamada desde Tcl debe agregarse a TclCommands

TclCommands.mybutton_clicked = mybutton_clicked
TclCommands.mybutton_pressed = mybutton_pressed
TclCommands.mybutton_released = mybutton_released
commands = TclCommands(root_window)
```

10.1.14.13. Lectura de variables internas

```
# las siguientes variables pueden leerse desde la instancia vars

print(vars.machine.get())
print(vars.emcini.get())

    active_codes            = StringVar
    block_delete            = BooleanVar
    brake                   = BooleanVar
    coord_type              = IntVar
    display_type            = IntVar
    dro_large_font          = IntVar
    emcini                  = StringVar
    exec_state              = IntVar
    feedrate                = IntVar
    flood                   = BooleanVar
    grid_size               = DoubleVar
    has_editor              = IntVar
    has_ladder              = IntVar
    highlight_line          = IntVar
    interp_pause            = IntVar
    interp_state            = IntVar
    ja_rbutton              = StringVar
    jog_aspeed              = DoubleVar
    jog_speed               = DoubleVar
    kinematics_type         = IntVar
    linuxcnctop_command     = StringVar
    machine                 = StringVar
    max_aspeed              = DoubleVar
    max_maxvel              = DoubleVar
    max_queued_mdi_commands = IntVar
    max_speed               = DoubleVar
    maxvel_speed            = DoubleVar
    mdi_command             = StringVar
    metric                  = IntVar
    mist                    = BooleanVar
    motion_mode             = IntVar
    on_any_limit            = BooleanVar
    optional_stop           = BooleanVar
    override_limits         = BooleanVar
```

```
    program_alpha           = IntVar
    queued_mdi_commands     = IntVar
    rapidrate               = IntVar
    rotate_mode             = BooleanVar
    running_line            = IntVar
    show_distance_to_go     = IntVar
    show_extents            = IntVar
    show_live_plot          = IntVar
    show_machine_limits     = IntVar
    show_machine_speed      = IntVar
    show_program            = IntVar
    show_pyvcppanel         = IntVar
    show_rapids             = IntVar
    show_tool               = IntVar
    show_offsets            = IntVar
    spindledir              = IntVar
    spindlerate             = IntVar
    task_mode               = IntVar
    task_paused             = IntVar
    task_state              = IntVar
    taskfile                = StringVar
    teleop_mode             = IntVar
    tool                    = StringVar
    touch_off_system        = StringVar
    trajcoordinates         = StringVar
    tto_g11                 = BooleanVar
    view_type               = IntVar
```

10.1.14.14. Ocultar widgets

```
# ocultar un widget
# use 'grid' o 'pack' dependiendo de cómo se colocó originalmente
root_window.tk.call('grid','forget','.pane.top.tabs.fmanual.jogf.zerohome.tooltouch')
```

10.1.14.15. Cambiar una etiqueta

```
# cambiar la etiqueta de un widget
root_window.tk.call('setup_widget_accel','.pane.top.tabs.fmanual.mist','Downdraft')

# asegurar que aparezca (solo es necesario en este caso si el botón de niebla estaba oculto ↩
    )
root_window.tk.call('grid','.pane.top.tabs.fmanual.mist','-column','1','-row','5','- ↩
    columnspan','2','-padx','4','-sticky','w')
```

10.1.14.16. Redirigir un comando existente

```
# secuestrar un comando existente
# originalmente el botón de niebla llama a la función de niebla
root_window.tk.call('.pane.top.tabs.fmanual.mist','configure','-command','hijacked_command' ↩
    )

# La nueva función
def hijacked_command():
    print('comando de niebla secuestrado')
```

```
# agrega la función a TclCommands
TclCommands.hijacked_command = hijacked_command
commands = TclCommands(root_window)
```

10.1.14.17. Cambiar el color DRO

```
# cambiar la pantalla dro
root_window.tk.call('.pane.top.right.fnumbers.text','configure','-foreground','green','- ↩
    background','black')
```

10.1.14.18. Cambiar los botones de la barra de herramientas

```
# cambiar los botones de la barra de herramientas

buW = '3'
buH = '2'
boW = '3'

root_window.tk.call('.toolbar.machine_estop','configure','-image','','-text','ESTOP','- ↩
    width',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.machine_power','configure','-image','','-text','POWER','- ↩
    width',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.file_open','configure','-image','','-text','OPEN','-width', ↩
    buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.reload','configure','-image','','-text','RELOAD','-width',buW ↩
    ,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_run','configure','-image','','-text','RUN','-width', ↩
    buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_step','configure','-image','','-text','STEP','-width' ↩
    ,buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_pause','configure','-image','','-text','PAUSE','- ↩
    width',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_stop','configure','-image','','-text','STOP','-width' ↩
    ,buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_blockdelete','configure','-image','','-text','Skip /' ↩
    ,'-width',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_optpause','configure','-image','','-text','M1','- ↩
    width',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.view_zoomin','configure','-image','','-text','Zoom+','-width' ↩
    ,buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.view_zoomout','configure','-image','','-text','Zoom-','-width ↩
    ',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.view_z','configure','-image','','-text','Top X','-width',buW, ↩
    '-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.view_z2','configure','-image','','-text','Top Y','-width',buW ↩
    ,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.view_x','configure','-image','','-text','Right','-width',buW, ↩
    '-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.view_y','configure','-image','','-text','Front','-width',buW, ↩
    '-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.view_p','configure','-image','','-text','3D','-width',buW,'- ↩
    height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.rotate','configure','-image','','-text','Rotate','-width',buW ↩
    ,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.clear_plot','configure','-image','','-text','Clear','-width', ↩
    buW,'-height',buH,'-borderwidth',boW)
```

10.1.14.19. Cambiar colores del plóter

En formato RGBA, en este orden: trote, rápidos, avance, arco, cambio de herramienta, sonda

```
# cambiar colores del plóter
try:
    live_plotter.logger.set_colors((255,0,0,255),
                                   (0,255,0,255),
                                   (0,0,255,255),
                                   (255,255,0,255),
                                   (255,255,255,255),
                                   (0,255,255,255))
except Exception as e:
    print(e)
```

10.2. GMOCCAPY

10.2.1. Introducción

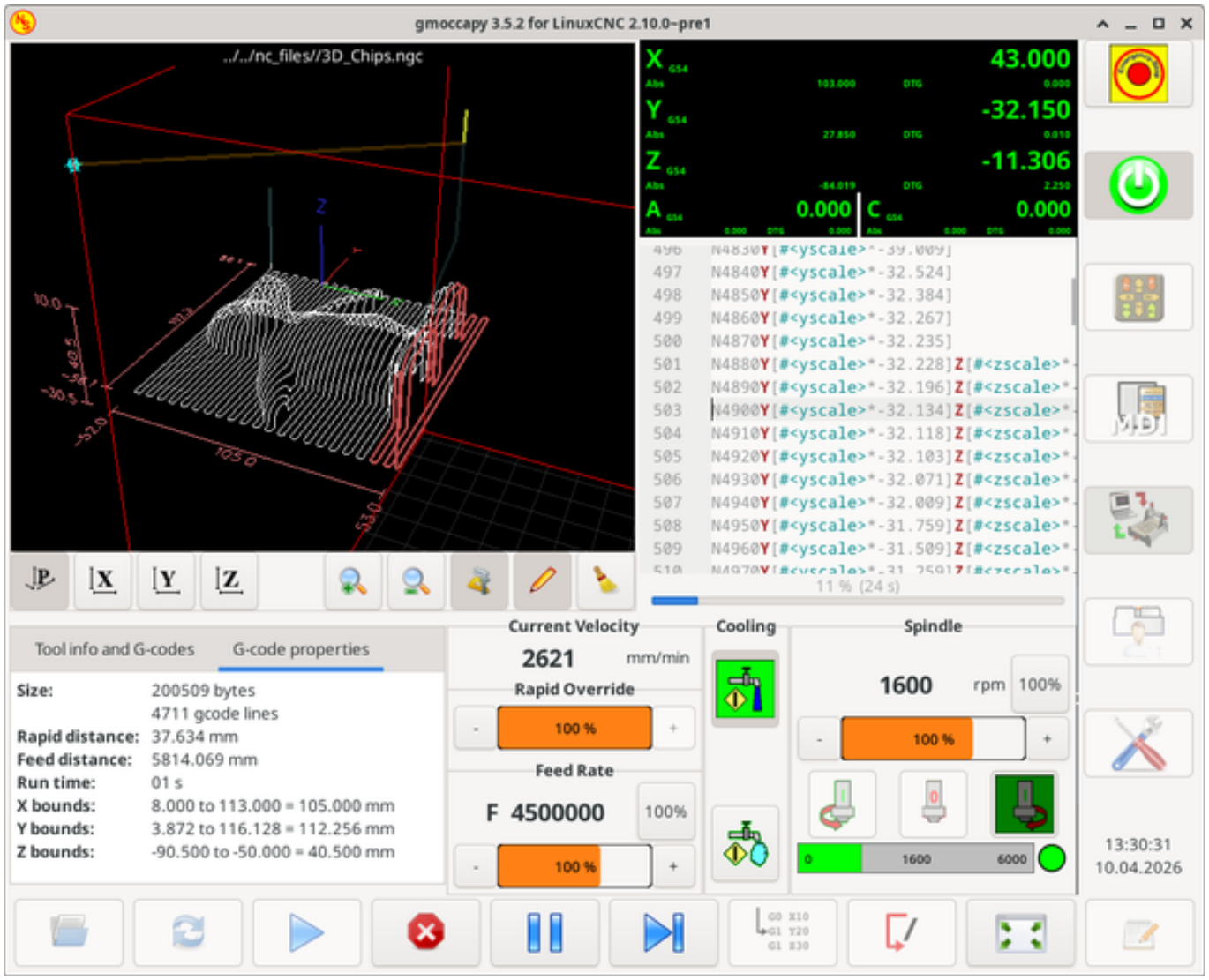
GMOCCAPY is a GUI for LinuxCNC, designed to be used with a touch screen, but can also be used on normal screens with a mouse or hardware buttons and MPG wheels, as it presents HAL Pins for the most common needs. Please find more information in the following.

Ofrece la posibilidad de mostrar hasta 4 ejes, admite un modo de torno para herramienta normal y posterior, que puede adaptarse a casi cualquier necesidad, ya que GMOCCAPY soporta pestañas y paneles laterales incrustados. Como un buen ejemplo de esto, puede verse gmoccapy_plasma.

GMOCCAPY 3 does support up to 9 axes and 9 joints. As GMOCCAPY 3 has been changed in code to support the joint / axis changes in LinuxCNC it does not work on 2.7 or 2.6 branch!

It has support for integrated virtual keyboard (onboard or matchbox-keyboard), so there is no need for a hardware keyboard or mouse, but it can also be used with that hardware. GMOCCAPY offers a separate settings page to configure most settings of the GUI without editing files.

GMOCCAPY can be localized very easy, because the corresponding files are separated from the linuxcnc.po files, so there is no need to translate unneeded stuff. If you want to contribute a translation, please use the web based translation editor Weblate. For more information see the section Translations



10.2.2. Requisitos

GMOCCAPY 3 se ha probado en Debian Jessie, Debian Stretch y MINT 18 con la rama master versión 2.8 de LinuxCNC. Es totalmente compatible con los cambios articulaciones/ejes de LinuxCNC, haciéndolo adecuado como GUI para Scara, Robots o cualquier otra configuración con más articulaciones que ejes. También soporta configuraciones de pórtico. Si usas otras versiones, por favor, informar sobre problemas y/o soluciones en foro LinuxCNC o en Foro CNC Ecke Alemán o en Lista de correo de usuarios de LinuxCNC.

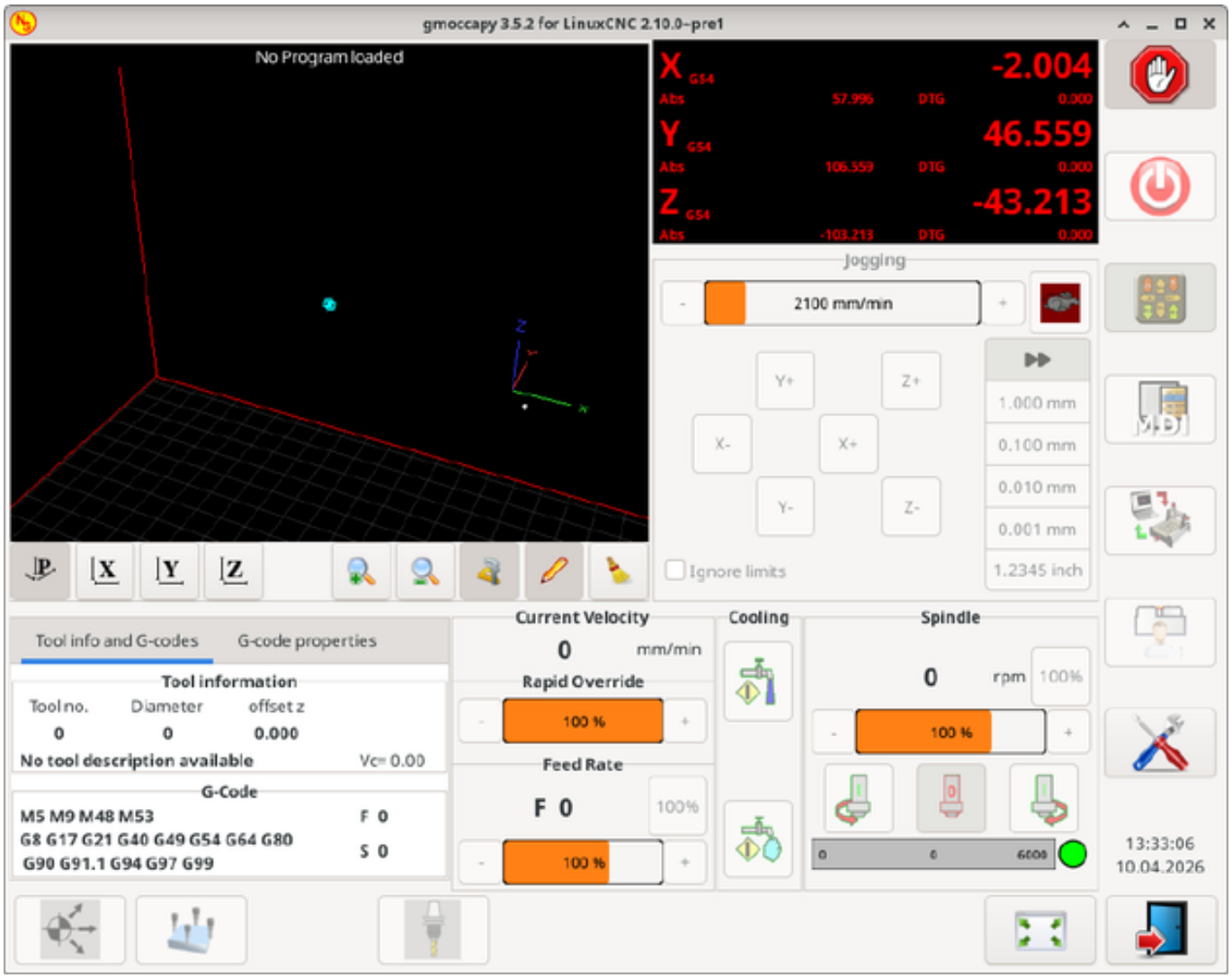
The minimum screen resolution for GMOCCAPY for the normal layout (without side panels) is **980 x 750 Pixel**, so it should fit to every standard screen. It is recommended to use screens with minimum resolution of 1024x768. There is also a configuration which fits for 800x600 screens (introduced in GMOCCAPY 3.4.8).

10.2.3. How to get GMOCCAPY

GMOCCAPY 3 is included in the standard distribution of LinuxCNC since release 2.7. So the easiest way to get GMOCCAPY on your controlling PC is just to download the ISO and install it from the CD/DVD/USB-stick. This allows you to receive updates with the regular Debian packages.

In the **release notes** aka changelist you can track the latest bugfixes and features.

Obtendrá una pantalla similar a la siguiente (el diseño puede variar dependiendo de su configuración.):



10.2.4. Configuración básica

GMOCCAPY 3 supports the following command line options:

- *-user_mode*: si está configurado, el botón de configuración se deshabilitará, por lo que los operadores normales de la máquina no podrán editar la configuración.
- *-logo <ruta al archivo de logotipo>*: si se proporciona, el logotipo ocultará la pestaña del botón de jog en modo manual, esto solo es útil para máquinas con botón de hardware para jogging y selección de incrementos.

There is really not to much to configure just to run GMOCCAPY, but there are some points you should take care off if you want to use all the features of the GUI.

You will find a lot of simulation configurations (INI files) included, just to show the basics:

- gmoccapy.ini
- gmoccapy_4_axis.ini

- lathe_configs/gmoccapy_lathe.ini
- lathe_configs/gmoccapy_lathe_imperial.ini
- gmoccapy_left_panel.ini
- gmoccapy_right_panel.ini
- gmoccapy_messages.ini
- gmoccapy_pendant.ini
- gmoccapy_sim_hardware_button.ini
- gmoccapy_tool_sensor.ini
- gmoccapy_with_user_tabs.ini
- gmoccapy_XYZAB.ini
- gmoccapy_XYZAC.ini
- gmoccapy_XYZCW.ini
- gmoccapy-JA/Gantry/gantry_mm.ini
- gmoccapy-JA/scara/scara.ini
- gmoccapy-JA/table-rotary-tilting/xyzac-trt.ini
- and a lot more ...

The names should explain the main intention of the different INI files.

If you use an existing configuration of your machine, just edit your INI according to this document.

Echemos un vistazo más de cerca al archivo INI y lo que necesita incluir para usar GMOCCAPY en su máquina:

10.2.4.1. The DISPLAY Section

```
[DISPLAY]
DISPLAY = gmoccapy
PREFERENCE_FILE_PATH = gmoccapy_preferences
MAX_FEED_OVERRIDE = 1.5
MAX_SPINDLE_OVERRIDE = 1.2
MIN_SPINDLE_OVERRIDE = 0.5
DEFAULT_SPINDLE_SPEED = 500
LATHE = 1
BACK_TOOL_LATHE = 1
PROGRAM_PREFIX = ../../nc_files/
```

- *DISPLAY = gmoccapy* - This tells LinuxCNC to use GMOCCAPY.
- *PREFERENCE_FILE_PATH* - Gives the location and name of the preferences file to be used. In most cases this line will not be needed, it is used by GMOCCAPY to store your settings of the GUI, like themes, DRO units, colors, and keyboard settings, etc., see settings page for more details.

> **nota**
> If no path or file is given, GMOCCAPY will use as default <your_machinename>.pref, if no machine name is given in your INI File it will use gmoccapy.pref. The file will be stored in your config directory, so the settings will not be mixed if you use several configs. If you only want to use one file for several machines, you need to include PREFERENCE_FILE_PATH in your INI.

- *MAX_FEED_OVERRIDE = 1.5* - Sets the maximum feed override, in the example given, you will be allowed to override the feed by 150 %.

  nota
  Si no se da ningún valor, se establecerá en 1.0.

- *MIN_SPINDLE_OVERRIDE = 0.5* and *MAX_SPINDLE_OVERRIDE = 1.2* - Will allow you to change the spindle override within a limit from 50 % to 120 %.

  nota
  Si no se dan valores, MIN se establecerá en 0.1 y MAX en 1.0.

- *LATHE = 1* - Set the screen layout to control a lathe.
- *BACK_TOOL_LATHE = 1* - Is optional and will switch the X axis in a way you need for a back tool lathe. Also the keyboard shortcuts will react in a different way. It is allowed with GMOCCAPY to configure a lathe also with additional axes, so you may use also a XZCW config for a lathe.

  sugerencia
  Consulte también la sección específica del torno.

- *PROGRAM_PREFIX = ../../nc_files/* - Is the entry to tell LinuxCNC/GMOCCAPY where to look for the NGC files.

  nota
  If not specified, GMOCCAPY will look in the following order for NGC files: First `linuxcnc/nc_files` and then the users home directory.

- *DEFAULT_SPINDLE_SPEED* - Start value for "Starting RPM" if value not present in preferences file or file is not present. Will have no effect with valid preferences file.
- *MIN_ANGULAR_VELOCITY* - Establece la velocidad de trote mínima de la máquina para ejes rotatorios.
- *MAX_ANGULAR_VELOCITY* - Establece la velocidad máxima de trote de la máquina para ejes rotatorios.
- *DEFAULT_ANGULAR_VELOCITY* - Sets the default jog velocity of the machine for rotary axes.

10.2.4.2. The TRAJ Section

- *DEFAULT_LINEAR_VELOCITY = 85.0* - Sets the default jog velocity of the machine.

  nota
  Si no se proporciona, se utilizará la mitad de *MAX_LINEAR_VELOCITY*. Si ese valor tampoco se da, se predeterminará a 180.

- *MAX_LINEAR_VELOCITY = 230.0* - Establece la velocidad máxima de la máquina. Este valor también será la velocidad de trote lineal máxima.

  nota
  Predeterminado a 600 si no se asigna.

10.2.4.3. Macro Buttons

You can add macros to GMOCCAPY, similar to Touchy's way. A macro is nothing else than a NGC file. You are able to execute complete CNC programs in MDI mode by just pushing one button. To do so, you first have to specify the search path for macros:

```
[RS274NGC]
SUBROUTINE_PATH = macros
```

Establece la ruta para buscar macros y otras subrutinas. Si quiere usar varias rutas de subrutinas, simplemente sepárelas con ":".

Then you just have to add a section like this:

Configuration of Five Macros to be Shown in the MDI Button List

```
[MACROS]
MACRO = i_am_lost
MACRO = hello_world
MACRO = jog_around
MACRO = increment xinc yinc
MACRO = go_to_position X-pos Y-pos Z-pos
```

Then you have to provide the corresponding NGC files which have to follow these rules:

- El nombre del archivo debe ser exactamente el mismo que el nombre mencionado en la linea MACRO, con la extensión ".ngc" (sensible a mayúsculas).
- El archivo debe contener una subrutina como: ***O<i_am_lost> sub***; el nombre de la subrutina debe coincidir exactamente (distingue entre mayúsculas y minúsculas) con el nombre de la macro.
- El archivo debe terminar con un endsub ***O<i_am_lost> endsub*** seguido de un comando ***M2***.
- The files need to be placed in a folder specified in your INI file by *SUBROUTINE_PATH* in the RS274NGC section

The code between sub and endsub will be executed by pushing the corresponding macro button.

---

nota
A maximum of 16 macros will be shown in the GUI. Due to space reasons you may need to click on an arrow to switch the page and display hidden macro buttons. The macro buttons will be displayed in the order of the INI entries. It is no error placing more than 16 macros in your INI file, they will just not be shown.

---

nota
You will find the sample macros in a folder named *macros* placed in the GMOCCAPY sim folder. If you have given several subroutine paths, they will be searched in the order of the given paths. The first file found will be used.

GMOCCAPY will also accept macros asking for parameters like:

```
[MACROS]
MACRO = go_to_position X-pos Y-pos Z-pos
```

The parameters must be separated by spaces. This example calls a file *go_to_position.ngc* with the following content:

```
; Test file ”go to position”
; will jog the machine to a given position

O<go_to_position> sub

G17
G21
G54
G61
G40
G49
G80
G90

;#1 = <X-Pos>
;#2 = <Y-Pos>
;#3 = <Z-Pos>

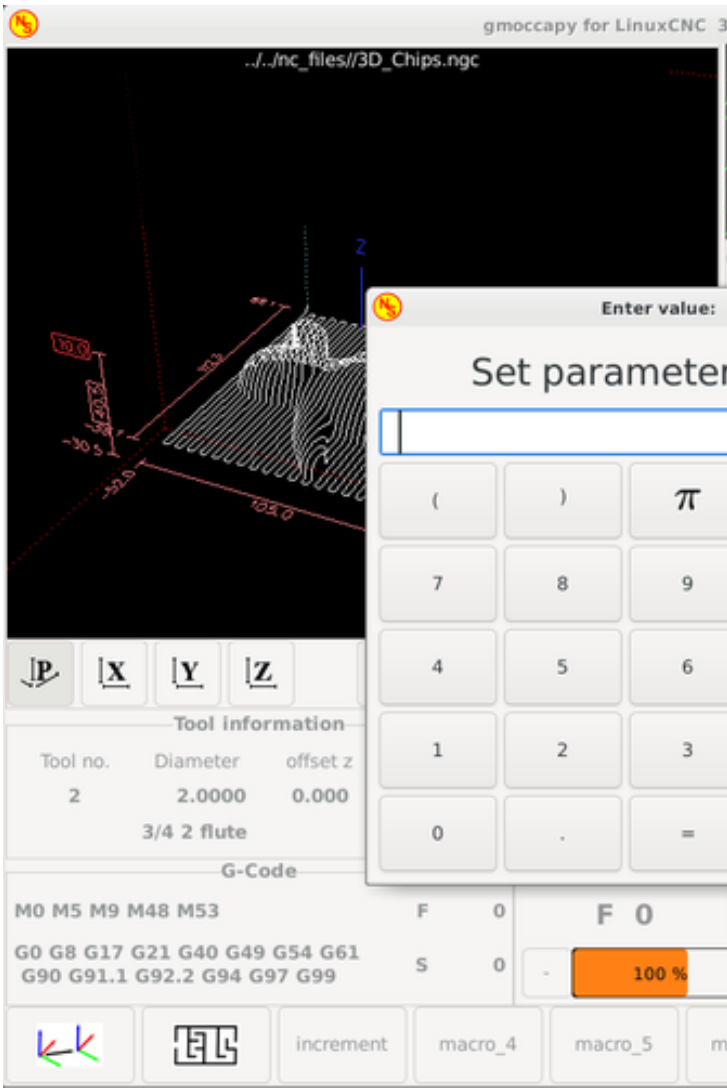
(DEBUG, Will now move machine to X = #1 , Y = #2 , Z = #3)
G0 X #1 Y #2 Z #3

O<go_to_position> endsub
M2
```

After pushing the ***execute macro button***, you will be asked to enter the values for ***X-pos Y-pos Z-pos*** and the macro will only run if all values have been given.

---

nota
Si desea usar una macro sin ningún movimiento, vea también las notas en problemas conocidos.



Macro example using the ”go to position”-macro

10.2.4.4. Pestañas y paneles incrustados

You can add embedded programs to GMOCCAPY like you can do in AXIS, Touchy and Gscreen. All is done by GMOCCAPY automatically if you include a few lines in your INI file in the DISPLAY section.

Si nunca ha usado un panel Glade, se recomienda leer la excelente documentación en Glade VCP.

Embedded Tab Example

```
EMBED_TAB_NAME = DRO
EMBED_TAB_LOCATION = ntb_user_tabs
EMBED_TAB_COMMAND = gladevcp -x {XID} dro.glade

EMBED_TAB_NAME = Second user tab
EMBED_TAB_LOCATION = ntb_preview
EMBED_TAB_COMMAND = gladevcp -x {XID} vcp_box.glade
```

Todo lo que debe tener en cuenta es incluir para cada pestaña o panel lateral las tres líneas mencionadas:

- EMBED_TAB_NAME = Representa el nombre de la pestaña o el panel lateral. Depende de usted qué nombre usar, pero ¡debe estar presente!

- EMBED_TAB_LOCATION = Es el lugar donde se colocará su programa en la GUI, ver figura Ubicaciones de pestañas incrustadas. Son valores válidos:
  - **ntb_user_tabs** (como pestaña principal, cubriendo la pantalla completa **(7)**)
  - **ntb_preview** (como pestaña en el lado de vista previa **(1)**)
  - **hbox_jog** (ocultará los botones de trote e introducirá su archivo glade aquí **(2)**)
  - **box_left** (a la izquierda, altura completa de la pantalla **(10)**)
  - **box_right** (a la derecha, entre la pantalla normal y la lista de botones **(11)**)
  - **box_tool_and_code_info** (ocultará los marcos de Información de la herramienta y Código G e introducirá su archivo glade ahí **(3)**)
  - **box_tool_info** (ocultará el cuadro de Información de la herramienta e introducirá su archivo glade ahí **(3a)**)
  - **box_code_info** (ocultará el cuadro de Información de código G e introducirá tu archivo glade ahí **(3b)**)
  - **box_vel_info** (ocultará los marcos de velocidad e introducirá tu archivo glade **(4)**)
  - **box_coolant_and_spindle** (ocultará los marcos del refrigerante y del husillo e introducirá ahí su archivo glade **(5)+(6)**)
  - **box_cooling** (ocultará el marco de refrigerante e introducirá su archivo glade **(5)**)
  - **box_spindle** (ocultará el marco del husillo e introducirá su archivo glade **(6)**)
  - **box_custom_1** (presentará tu archivo glade a la izquierda de vel_frame)
  - **box_custom_2** (presentará tu archivo glade a la izquierda de cooling_frame)
  - **box_custom_3** (presentará tu archivo glade a la izquierda de spindle_frame)
  - **box_custom_4** (presentará su archivo glade a la derecha de spindle_frame)
  - **box_dro_side** (presentará su archivo glade a la derecha de DRO **(8)**)
  - **ntb_setup** (como pestaña en la página de configuración **(9)**)

---

nota
See also the included sample INI files to see the differences.

- EMBED_TAB_COMMAND = El comando a ejecutar, p. ej.

```
gladevcp -x {XID} dro.glade
```

includes a custom glade file called dro.glade in the mentioned location. The file must be placed in the config folder of your machine.

```
gladevcp h_buttonlist.glade
```

will just open a new user window called h_buttonlist.glade note the difference. This one is stand alone, and can be moved around independent from GMOCCAPY window.

```
gladevcp -c gladevcp -u hitcounter.py -H manual-example.hal manual-example.ui
```

will add a the panel manual-example.ui, include a custom Python handler, hitcounter.py and make all connections after realizing the panel according to manual-example.hal.

```
hide
```

will hide the chosen box.

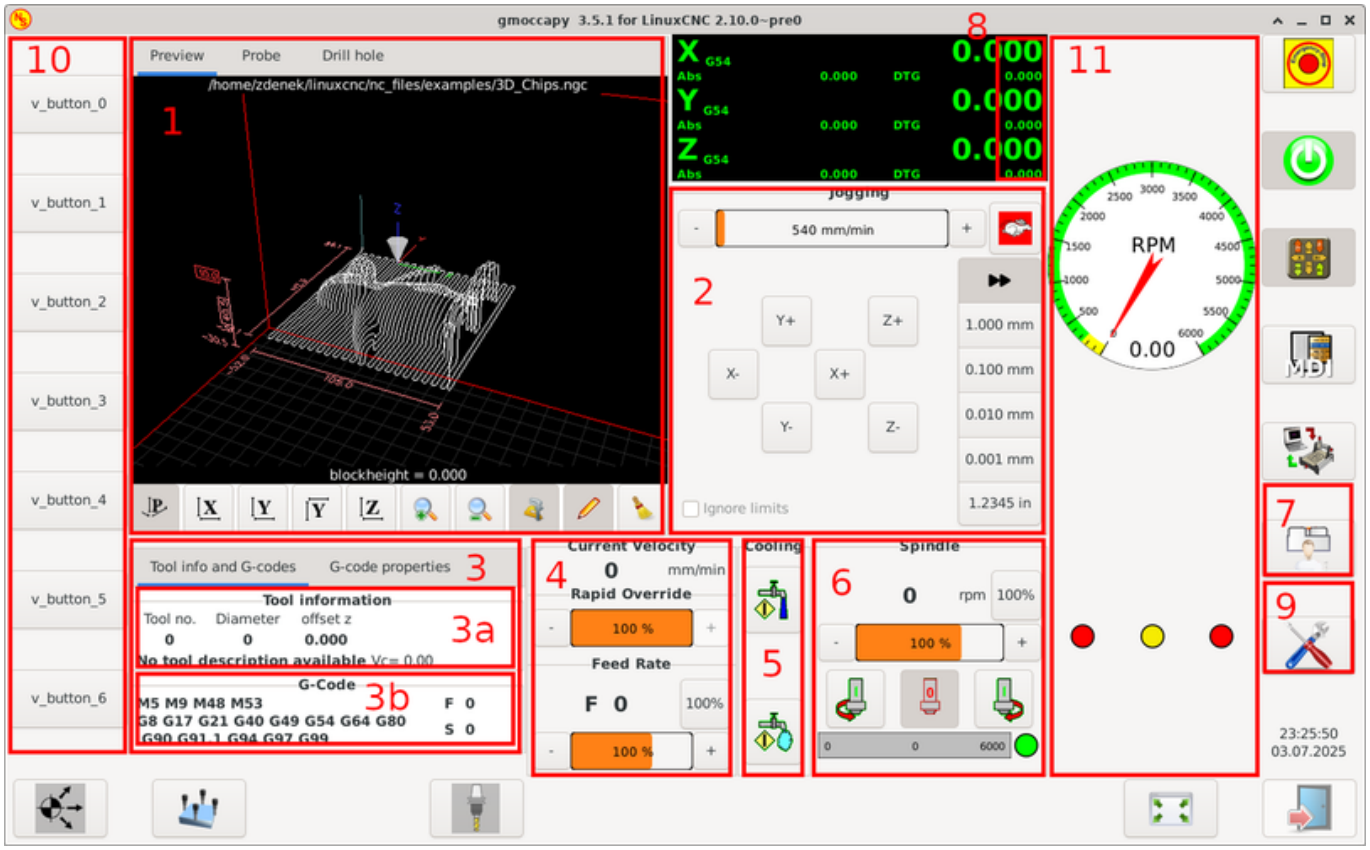


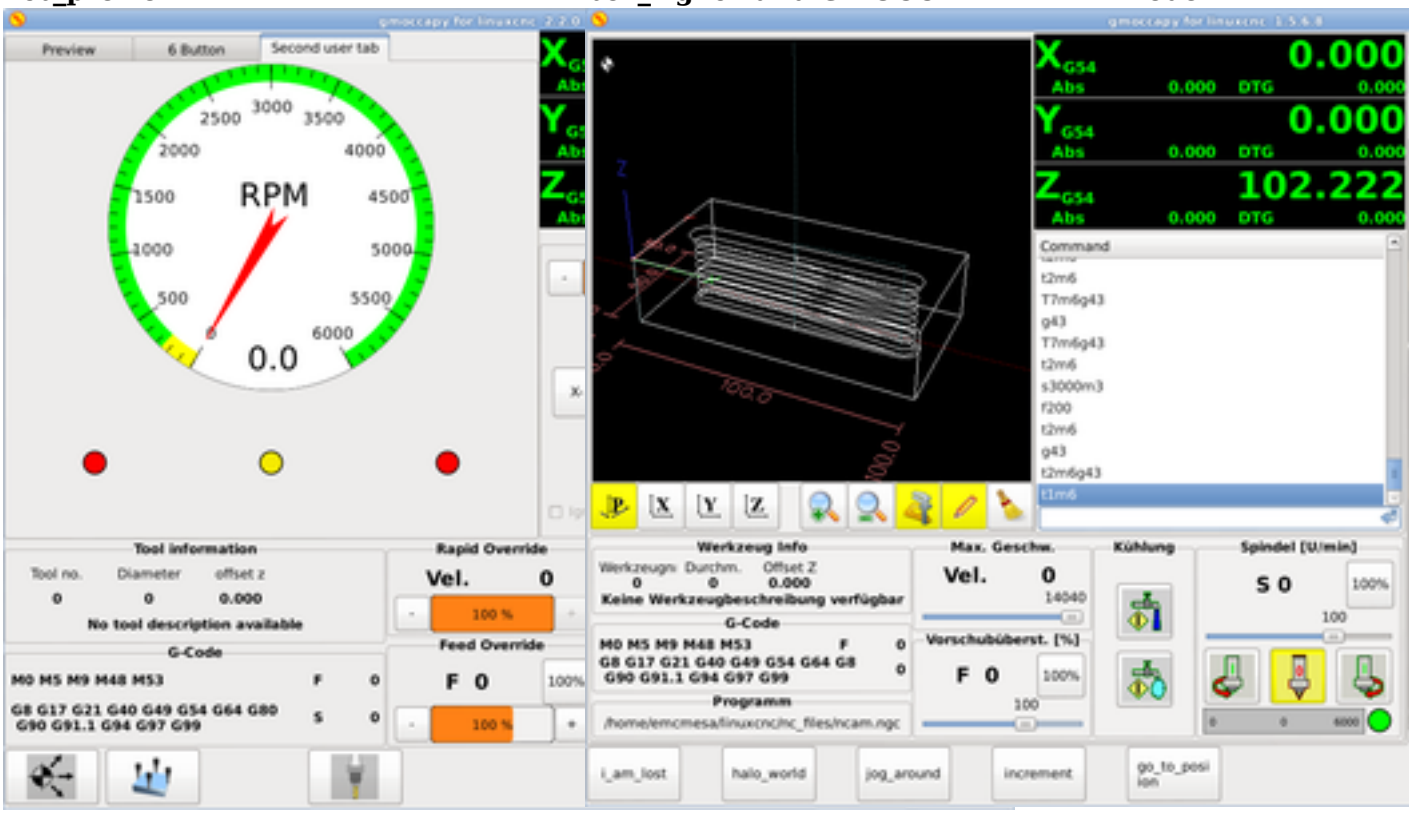
Figura 10.14: Embedded tab locations

---

nota
Si realiza alguna conexión HAL a su panel de glade personalizado, debe hacerlo en el archivo HAL especificado en la línea EMBED_TAB_COMMAND; de lo contrario, puede obtener un error de que el pin HAL no existe — esto se debe a las condiciones de carrera al cargar los archivos HAL. Las conexiones a pines GMOCCAPY HAL deben realizarse en el archivo HAL postgui especificado en su archivo INI, porque esos pines no existen antes de realizar la GUI.

Here are some examples:

ntb_preview **box_right - and GMOCCAPY in MDI mode**



10.2.4.5. User Created Messages

GMOCCAPY has the ability to create HAL driven user messages. To use them you need to introduce some lines in the [DISPLAY] section of the INI file.

These three lines are needed to define a user pop up message dialog:

```
MESSAGE_TEXT    = The text to be displayed, may be pango markup formatted
MESSAGE_TYPE    = ”status” , ”okdialog” , ”yesnodialog”
MESSAGE_PINNAME = is the name of the HAL pin group to be created
```

Los mensajes admiten el lenguaje de marcado pango. Puede encontrar información detallada sobre el lenguaje de marcado en Pango Markup.

The following three dialog types are available:

- **status** - Solo mostrará un mensaje como ventana emergente, usando el sistema de mensajes de GMOCCAPY.
- **okdialog** - Mantendrá el foco en el cuadro de diálogo del mensaje y activará un pin HAL `-waiting`.
- *yesnodialog* - Mantendrá el foco en el cuadro de diálogo del mensaje y activará un pin HAL `-waiting` y proveerá un pin HAL `-response`.

For more detailed information of the pins see User Created Message HAL Pins.

Example of User Message Configuration

```
MESSAGE_TEXT = Esto es una <span background=”#ff0000” foreground=”#ffffff”>información del ↩
    mensaje</span>prueba
MESSAGE_TYPE = status
MESSAGE_PINNAME = statustest
```

```
MESSAGE_TEXT = This is a yes no dialog test
MESSAGE_TYPE = yesnodialog
MESSAGE_PINNAME = yesnodialog

MESSAGE_TEXT = El texto puede ser <small>pequeño</small>, <big>grande</big>, <b>negrito</b ↩
    >,<i>itálico</i>, e incluso puede ser <span color="red">coloreado</span>.
MESSAGE_TYPE = okdialog
MESSAGE_PINNAME = okdialog
```

nota
Currently the formatting doesn't work.

10.2.4.6. Control de vista previa

Magic comments can be used to control the G-code preview. On very large programs the preview can take a long time to load. You can control what is shown and what is hidden on the graphics screen by adding the appropriate comments from this list into your G-code:

```
(PREVIEW,hide)
<G-code to be hidden>
(PREVIEW,show)
```

10.2.4.7. User Command File

Si existe el archivo `~/.gmoccapyrc` se ejecuta su contenido como código fuente Python justo después de mostrar la interfaz gráfica de usuario. Los detalles de lo que se puede escribir en `~/.gmoccapyrc` están sujetos a cambios durante el ciclo de desarrollo.

A configuration-specific Python file may be specified with an INI file setting

```
[DISPLAY]
USER_COMMAND_FILE=filename.py
```

If this file is specified, this file is sourced just after the GMOCCAPY GUI is displayed **instead** of `~/.gmoccapyrc`.

El ejemplo siguiente cambia el tamaño de los botones verticales: .Ejemplo de archivo .gmoccapyrc

```
self.widgets.vbtb_main.set_size_request(85,-1)
BB_SIZE = (70, 70) # default = (90, 56)
self.widgets.tbtn_estop.set_size_request(*BB_SIZE)
self.widgets.tbtn_on.set_size_request(*BB_SIZE)
self.widgets.rbt_manual.set_size_request(*BB_SIZE)
self.widgets.rbt_mdi.set_size_request(*BB_SIZE)
self.widgets.rbt_auto.set_size_request(*BB_SIZE)
self.widgets.tbtn_setup.set_size_request(*BB_SIZE)
self.widgets.tbtn_user_tabs.set_size_request(*BB_SIZE)
self.widgets.btn_exit.set_size_request(*BB_SIZE)
```

The widget names can the looked up in the /usr/share/gmoccapy.glade file

10.2.4.8. User CSS File

Similar to the User command file it's possible to influence the appearance by cascading style sheets (CSS). If a file `~/.gmoccapy_css` exists, its contents are loaded into the stylesheet provider and are so being applied to the GUI.

A configuration-specific CSS file may be specified with an INI file setting

```
[DISPLAY]
USER_CSS_FILE=filename.css
```

If this file is specified, this file is used **instead** of `~/.gmoccapy_css`.

Information what can be controlled by CSS can be found here: Overview of CSS in GTK

Here an example how the color of checked buttons can be set to yellow: .Example Yellow color for checked buttons

```
button:checked {
    background: rgba(250,230,0,0.8);
}
```

10.2.4.9. Logging

GMOCCAPY supports specifying the level of information (log level) that will be printed to the console and to the log file.

The order is *VERBOSE*, *DEBUG*, *INFO*, *WARNING*, *ERROR*, *CRITICAL*. Default is *WARNING*, that means *WARNING*, *ERROR* and *CRITICAL* are printed.

You can specify the log level in the INI file like this:

```
[DISPLAY]
DISPLAY = gmoccapy <log_level_param>
```

usando estos parámetros:

```
Log level   <log_level_param>
DEBUG       -d
INFO        -i
VERBOSE     -v
ERROR       -q
```

Example: Configure logging to print only errors

```
[DISPLAY]
DISPLAY = gmoccapy -q
```

You can specify where to save the log file:

```
[DISPLAY]
LOG_FILE = gmoccapy.log
```

If `LOG_FILE` is not set, logging happens to `$HOME/<base_log_name>.log`.

10.2.5. HAL Pins

GMOCCAPY exports several HAL pins to be able to react to hardware devices. The goal is to get a GUI that may be operated in a tool shop, completely/mostly without mouse or keyboard.

---

nota
You will have to do all connections to GMOCCAPY pins in your postgui.hal file. When GMOCCAPY is started, it creates the HAL pins for the GUI then it executes the post-GUI HAL file named in the INI file:

```
[HAL]
POSTGUI_HALFILE=<filename>
```

Típicamente <nombre-de-archivo> ѕerá la base del nombre de configuraciones + `_postgui.hal, p. ej. lathe_postgui.hal, pero puede ser cualquier nombre de archivo válido.
Estos comandos se ejecutan después de construir la pantalla, garantizando que los pines HAL de los widgets estén disponibles.
El archivo INI puede tener múltiples líneas de POSTGUI_HALFILE=<nombre-de-archivo>. Cada una se ejecutará una tras otra en orden de aparición.

10.2.5.1. Right and Bottom Button Lists

The screen has two main button lists, one on the right side an one on the bottom. The right handed buttons will not change during operation, but the bottom button list will change very often. The buttons are count from up to down and from left to right beginning with 0.

---

nota
Los nombres de pin han cambiado en GMOCCAPY 2 para ordenarlos mejor.

The pins for the right (vertical) buttons are:

- **gmoccapy.v-button.button-0** *(bit IN)*
- **gmoccapy.v-button.button-1** *(bit IN)*
- **gmoccapy.v-button.button-2** *(bit IN)*
- **gmoccapy.v-button.button-3** *(bit IN)*
- **gmoccapy.v-button.button-4** *(bit IN)*
- **gmoccapy.v-button.button-5** *(bit IN)*
- **gmoccapy.v-button.button-6** *(bit IN)*

For the bottom (horizontal) buttons they are:

- **gmoccapy.h-button.button-0** *(bit IN)*
- **gmoccapy.h-button.button-1** *(bit IN)*
- **gmoccapy.h-button.button-2** *(bit IN)*
- **gmoccapy.h-button.button-3** *(bit IN)*
- **gmoccapy.h-button.button-4** *(bit IN)*

- **gmoccapy.h-button.button-5** *(bit IN)*
- **gmoccapy.h-button.button-6** *(bit IN)*
- **gmoccapy.h-button.button-7** *(bit IN)*
- **gmoccapy.h-button.button-8** *(bit IN)*
- **gmoccapy.h-button.button-9** *(bit IN)*

As the buttons in the bottom list will change according to the mode and other influences, the hardware buttons will activate the displayed functions. So you don't have to take care about switching functions around in HAL, because that is done completely by GMOCCAPY!

For a three axes XYZ machine the HAL pins will react as shown in the following three tables:

Cuadro 10.3: Functional assignment of horizontal buttons (1)

Pin	Modo manual	MDI Mode	Auto Mode
gmoccapy.h-button.button-0	open homing button	macro 1 (if defined)	open file
gmoccapy.h-button.button-1	open touch off stuff	macro 2 (if defined)	reload program
gmoccapy.h-button.button-2		macro 3 (if defined)	run
gmoccapy.h-button.button-3	open tool dialogs	macro 4 (if defined)	stop
gmoccapy.h-button.button-4		macro 5 (if defined)	pause
gmoccapy.h-button.button-5		macro 6 (if defined)	step by step
gmoccapy.h-button.button-6		macro 7 (if defined)	run from line if enabled in settings, otherwise nothing
gmoccapy.h-button.button-7		macro 8 (if defined)	bloques opcionales
gmoccapy.h-button.button-8	full-size preview	macro 9 or button to show additional macros	full-size preview
gmoccapy.h-button.button-9	exit if machine is off, otherwise nothing	open keyboard or abort if macro is running	editar código

Cuadro 10.4: Functional assignment of horizontal buttons (2)

Pin	Settings Mode	Homing Mode	Modo touch off
gmoccapy.h-button.button-0	delete MDI history		touch X
gmoccapy.h-button.button-1		home all	touch Y
gmoccapy.h-button.button-2			touch Z
gmoccapy.h-button.button-3		home x	
gmoccapy.h-button.button-4	open classic ladder	home y	
gmoccapy.h-button.button-5	open HAL scope	home z	
gmoccapy.h-button.button-6	open HAL status		
gmoccapy.h-button.button-7	open HAL meter		
gmoccapy.h-button.button-8	open HAL calibration	unhome all	
gmoccapy.h-button.button-9	open HAL show	back	back

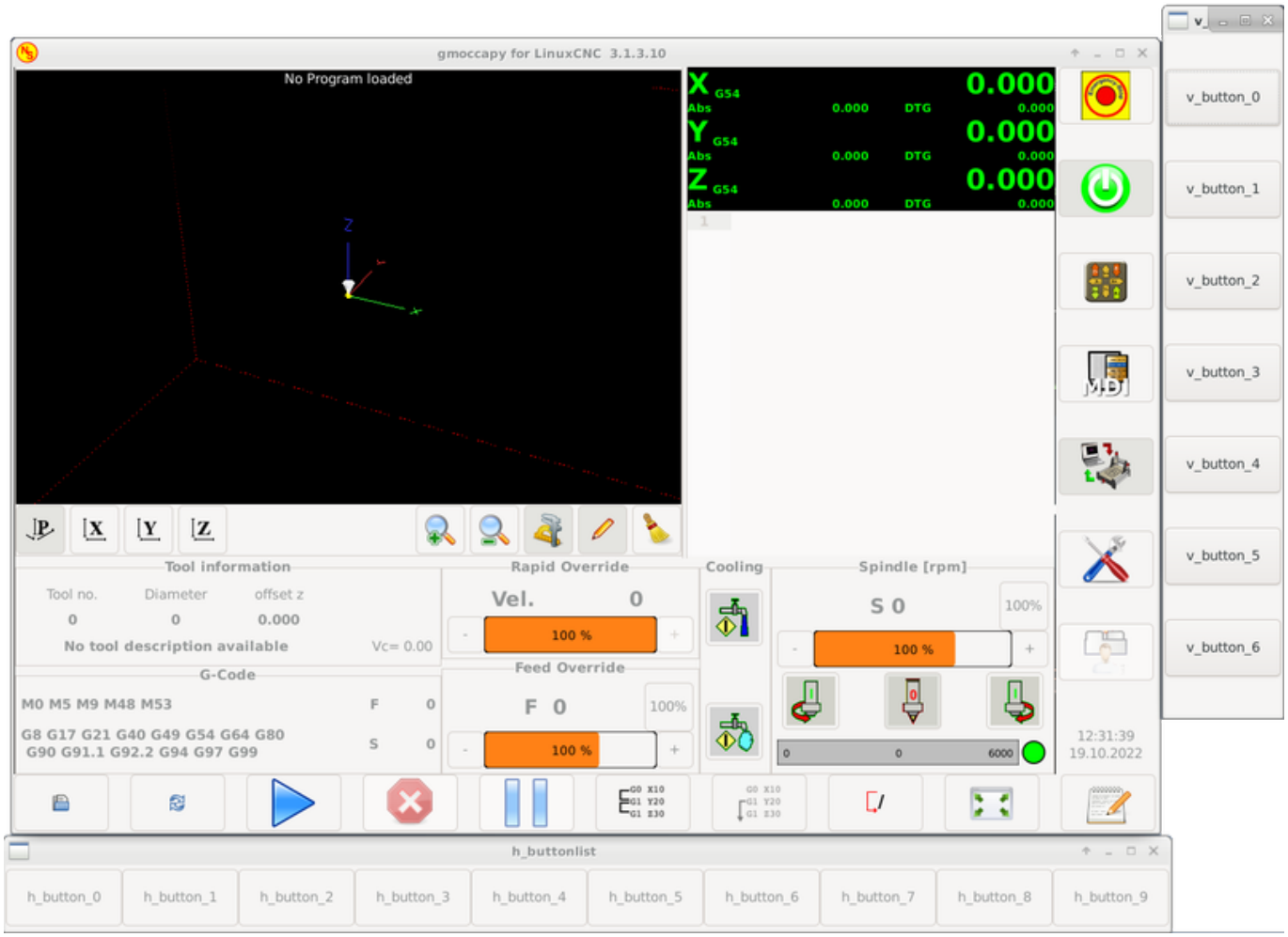
Cuadro 10.5: Functional assignment of horizontal buttons (3)

Pin	Tool Mode	Edit Mode	Select File
gmoccapy.h-button.button-0			go to home directory
gmoccapy.h-button.button-1		reload file	one directory level up
gmoccapy.h-button.button-2		save	
gmoccapy.h-button.button-3		save as	move selection left
gmoccapy.h-button.button-4	change tool by number T? M6		move selection right
gmoccapy.h-button.button-5	set tool by number without change M61 Q?		jump to directory as set in settings
gmoccapy.h-button.button-6	change tool to the selected one	new file	
gmoccapy.h-button.button-7			select / ENTER
gmoccapy.h-button.button-8	toque de herramienta en Z	show keyboard	
gmoccapy.h-button.button-9	back	back	back

¡Así que tenemos 67 reacciones con sólo 10 pines de HAL!

These pins are made available to be able to use the screen without a touch panel, or protect it from excessive use by placing hardware buttons around the panel. They are available in a sample configuration like shown in the image below.

Sample configuration ”gmoccapy_sim_hardware_button” showing the side buttons



10.2.5.2. Velocities and Overrides

All sliders from GMOCCAPY can be connected to hardware encoders or hardware potentiometers.

nota
For GMOCCAPY 3 some HAL pin names have changed when new controls have been implemented. Max velocity does not exist any more, it was replaced by rapid override due to the demand of many users.

To connect encoders, the following pins are exported:

- **gmoccapy.jog.jog-velocity.counts** *(s32 IN)* - Jog velocity
- **gmoccapy.jog.jog-velocity.count-enable** *(bit IN)* - Must be True, to enable counts
- **gmoccapy.feed.feed-override.counts** *(s32 IN)* - feed override
- **gmoccapy.feed.feed-override.count-enable** *(bit IN)* - Must be True, to enable counts
- **gmoccapy.feed.reset-feed-override** *(bit IN)* - reset the feed override to 0 %
- **gmoccapy.spindle.spindle-override.counts** *(s32 IN)* - spindle override
- **gmoccapy.spindle.spindle-override.count-enable** *(bit IN)* - Must be True, to enable counts

- **gmoccapy.spindle.reset-spindle-override** *(bit IN)* - reset the spindle override to 0 %
- **gmoccapy.rapid.rapid-override.counts** *(s32 IN)* - Maximal velocity of the machine
- **gmoccapy.rapid.rapid-override.count-enable** *(bit IN)* - Must be True, to enable counts

To connect potentiometers, use the following pins:

- **gmoccapy.jog.jog-velocity.direct-value** *(float IN)* - To adjust the jog velocity slider
- **gmoccapy.jog.jog-velocity.analog-enable** *(bit IN)* - Must be True, to allow analog inputs
- **gmoccapy.feed.feed-override.direct-value** *(float IN)* - To adjust the feed override slider
- **gmoccapy.feed.feed-override.analog-enable** *(bit IN)* - Must be True, to allow analog inputs
- **gmoccapy.spindle.spindle-override.direct-value** *(float IN)* - To adjust the spindle override slider
- **gmoccapy.spindle.spindle-override.analog-enable** *(bit IN)* - Must be True, to allow analog inputs
- **gmoccapy.rapid.rapid-override.direct-value** *(float)* - To adjust the max velocity slider
- **gmoccapy.rapid.rapid-override.analog-enable** *(bit IN)* - Must be True, to allow analog inputs

In addition, GMOCCAPY 3 offers additional HAL pins to control the new slider widgets with momentary switches. The values how fast the increase or decrease will be, must be set in the glade file. In a future release it will be integrated in the settings page.

SPEED

- **gmoccapy.spc_jog_vel.increase** *(bit IN)* - As long as True the value of the slider will increase
- **gmoccapy.spc_jog_vel.decrease** *(bit IN)* - As long as True the value of the slider will decrease
- **gmoccapy.spc_jog_vel.scale** *(float IN)* - A value to scale the output value (handy to change units/min to units/sec)
- **gmoccapy.spc_jog_vel.value** *(float OUT)* - Value of the widget
- **gmoccapy.spc_jog_vel.scaled-value** *(float OUT)* - Scaled value of the widget

FEED

- **gmoccapy.spc_feed.increase** *(bit IN)* - As long as True the value of the slider will increase
- **gmoccapy.spc_feed.decrease** *(bit IN)* - As long as True the value of the slider will decrease
- **gmoccapy.spc_feed.scale** *(float IN)* - A value to scale the output value (handy to change units/min to units/sec)
- **gmoccapy.spc_feed.value** *(float OUT)* - Value of the widget
- **gmoccapy.spc_feed.scaled-value** *(float OUT)* - Scaled value of the widget

SPINDLE

- **gmoccapy.spc_spindle.increase** *(bit IN)* - As long as True the value of the slider will increase
- **gmoccapy.spc_spindle.decrease** *(bit IN)* - As long as True the value of the slider will decrease
- **gmoccapy.spc_spindle.scale** *(float IN)* - A value to scale the output value (handy to change units/min to units/sec)

- **gmoccapy.spc_spindle.value** *(float OUT)* - Value of the widget
- **gmoccapy.spc_spindle.scaled-value** *(float OUT)* - Scaled value of the widget

RAPIDS

- **gmoccapy.spc_rapid.increase** *(bit IN)* - As long as True the value of the slider will increase
- **gmoccapy.spc_rapid.decrease** *(bit IN)* - As long as True the value of the slider will decrease
- **gmoccapy.spc_rapid.scale** *(float IN)* - A value to scale the output value (handy to change units/min to units/sec)
- **gmoccapy.spc_rapid.value** *(float OUT)* - Value of the widget
- **gmoccapy.spc_rapid.scaled-value** *(float OUT)* - Scaled value of the widget

Los pines float aceptan valores de 0.0 a 1.0, siendo el valor de porcentaje que desea establecer el valor del control deslizante.



aviso
If you use both connection types, do not connect the same slider to both pin as the influences between the two has not been tested! Different sliders may be connected to the one or other HAL connection type.



importante
Tenga en cuenta que la velocidad de jog depende del estado del botón tortuga. Tendrá escalas deslizantes diferentes dependiendo del modo (tortuga o conejo). Por favor, eche un vistazo también a velocidades jog y pin HAL jog-tortuga para más detalles.

Ejemplo 10.1 Setting a slider value

Spindle Override Min Value = 20 %
Spindle Override Max Value = 120 %
gmoccapy.analog-enable = 1
gmoccapy.spindle-override-value = 0.25

value to set = Min Value + (Max Value - Min Value) * gmoccapy.spindle-override-value
value to set = 20 + (120 - 20) * 0.25
value to set = 45 %

10.2.5.3. Jog HAL Pins

All axes given in the INI file have a jog-plus and a jog-minus pin, so hardware momentary switches can be used to jog the axes.

nota
El nombre de estos pines ha cambiado en GMOCCAPY 2.

For the standard XYZ config following HAL pins will be available:

- **gmoccapy.jog.axis.jog-x-plus** *(bit IN)*
- **gmoccapy.jog.axis.jog-x-minus** *(bit IN)*
- **gmoccapy.jog.axis.jog-y-plus** *(bit IN)*
- **gmoccapy.jog.axis.jog-y-minus** *(bit IN)*
- **gmoccapy.jog.axis.jog-z-plus** *(bit IN)*
- **gmoccapy.jog.axis.jog-z-minus** *(bit IN)*

Si usa configuración de 4 ejes, habrá dos pines adicionales:

- **gmoccapy.jog.jog-<your fourth axis letter >-plus** *(bit IN)*
- **gmoccapy.jog.jog-<your fourth axis letter >-minus** *(bit IN)*

For a C-axis you will see:

- **gmoccapy.jog.axis.jog-c-plus** *(bit IN)*
- **gmoccapy.jog.axis.jog-c-minus** *(bit IN)*

10.2.5.4. Jog Velocities and Turtle-Jog HAL Pin

The jog velocity can be selected with the corresponding slider. The scale of the slider will be modified if the turtle button (the one showing a rabbit or a turtle) has been toggled. If the button is not visible, it might have been disabled on the settings page. If the button shows the rabbit-icon, the scale is from min to max machine velocity. If it shows the turtle, the scale will reach only 1/20 of max velocity by default. The used divider can be set on the settings page.

So using a touch screen it is much easier to select smaller velocities.

GMOCCAPY ofrece este pin HAL para alternar entre jogs tortuga y conejo:

- **gmoccapy.jog.turtle-jog** *(bit IN)*

10.2.5.5. Jog Increment HAL Pins

Los incrementos de trote dados en el archivo INI como

```
[DISPLAY]
INCREMENTS = 5mm 1mm .5mm .1mm .05mm .01mm
```

are selectable through HAL pins, so a selection hardware switch can be used to select the increment to use. There will be a maximum of 10 HAL pins for the increments given in the INI file. If you give more increments in your INI file, they will be not reachable from the GUI as they will not be displayed.

If you have 6 increments in your INI file like in the example above, you will get **7** pins:

- **gmoccapy.jog.jog-inc-0** *(bit IN)* - This one is fixed and will represent continuous jogging.
- **gmoccapy.jog.jog-inc-1** *(bit IN)* - Primer incremento dado en el archivo INI.
- **gmoccapy.jog.jog-inc-2** *(bit IN)*
- **gmoccapy.jog.jog-inc-3** *(bit IN)*
- **gmoccapy.jog.jog-inc-4** *(bit IN)*

- **gmoccapy.jog.jog-inc-5** *(bit IN)*
- **gmoccapy.jog.jog-inc-6** *(bit IN)*

GMOCCAPY ofrece también un pin HAL para dar salida al incremento de jog seleccionado:

- **gmoccapy.jog.jog-increment** *(float OUT)*

10.2.5.6. Hardware Unlock Pin

To be able to use a key switch to unlock the settings page, the following pin is exported:

- **gmoccapy.unlock-settings** *(bit IN)* - The settings page is unlocked if the pin is high. To use this pin, you need to activate it on the settings page.

10.2.5.7. Error/Warning Pins

- **gmoccapy.error** *(bit OUT)* - Indicates an error, so a light can lit or even the machine may be stopped. It will be reset with the pin `gmoccapy.delete-message`.
- **gmoccapy.delete-message** *(bit IN)* - Will delete the first error and reset the `gmoccapy.error` pin to false after the last error has been cleared.
- **gmoccapy.warning-confirm** *(bit IN)* - Confirms warning dialog like click on OK

---

nota
Messages or user infos will not affect the `gmoccapy.error` pin, but the `gmoccapy.delete-message` pin will delete the last message if no error is shown!

10.2.5.8. User Created Message HAL Pins

GMOCCAPY may be configured to react to external errors, using 3 different user messages:

status

- **gmoccapy.messages.status** *(bit IN)* - Dispara el diálogo.

okdialog

- **gmoccapy.messages.okdialog** *(bit IN)* - Dispara el diálogo.
- **gmoccapy.messages.okdialog-waiting** *(bit OUT)* - Will be *1* as long as the dialog is open. Closing the message will reset the this pin.

yesnodialog

- **gmoccapy.messages.yesnodialog** *(bit IN)* - Dispara el diálogo.
- **gmoccapy.messages.yesnodialog-waiting** *(bit OUT)* - Will be *1* as long as the dialog is open. Closing the message will reset the this pin.
- **gmoccapy.messages.yesnodialog-response** *(bit OUT)* - This pin will change to *1* if the user clicks OK and in all other cases it will be *0*. This pin will remain *1* until the dialog is called again.

To add a user created message you need to add the message to the INI file in the DISPLAY section. See Configuration of User Created Messages.

User Message Example (INI file)

```
MESSAGE_TEXT = Fallo de lubricación
MESSAGE_TYPE = okdialog
MESSAGE_PINNAME = lube-fault

MESSAGE_TEXT = PIN DE CORTE X ROTO
MESSAGE_TYPE = status
MESSAGE_PINNAME = pin
```

To connect these new pins you need to do this in the postgui HAL file. Here are some example connections which connect the message signals to some place else in the HAL file.

Example Connection of User Messages (HAL file)

```
net gmoccapy-lube-fault gmoccapy.messages.lube-fault
net gmoccapy-lube-fault-waiting gmoccapy.messages.lube-fault-waiting
net gmoccapy-pin gmoccapy.messages.pin
```

For more information about HAL files and the net command see the HAL Basics.

10.2.5.9. Spindle Feedback Pins

Hay dos pines para la retroalimentación del husillo:

- **gmoccapy.spindle_feedback_bar** *(float IN)* - Pin para mostrar la velocidad del husillo on la barra del husillo.
- **gmoccapy.spindle_at_speed_led** *(bit IN)* - Pin to lit the is-at-speed-led.

10.2.5.10. Pins to Indicate Program Progress Information

There are three pins giving information about the program progress:

- **gmoccapy.program.length** *(s32 OUT)* - Muestra el número total de líneas del programa.
- **gmoccapy.program.current-line** *(s32 OUT)* - Indica la línea actual en ejecución del programa.
- **gmoccapy.program.progress** *(float OUT)* - Da el progreso del programa en porcentaje.

The values may not be very accurate if you are working with subroutines or large remap procedures. Also loops will cause different values.

10.2.5.11. Tool Related Pins

Tool Change Pins These pins are provided to use GMOCCAPY's internal tool change dialog, similar to the one known from AXIS, but with several modifications. So you will not only get the message to change to *tool number 3*, but also the description of that tool like *7.5 mm 3 flute cutter*. The information is taken from the tool table, so it is up to you what to display.

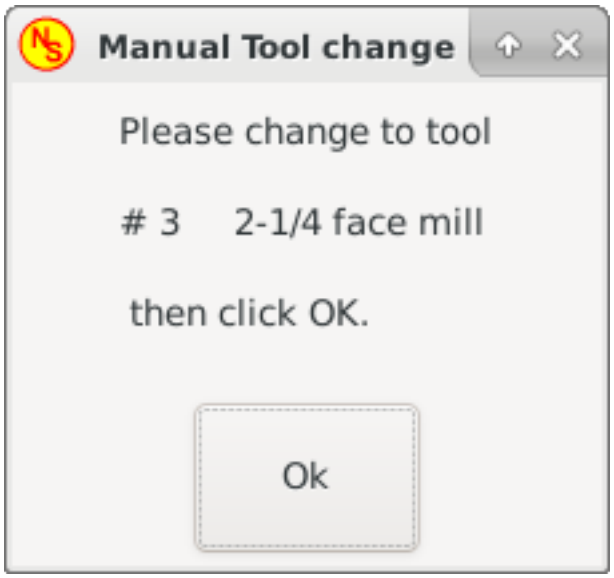


Figura 10.15: GMOCCAPY tool change dialog

- **gmoccapy.toolchange-number** *(s32 IN)* - The number of the tool to be changed
- **gmoccapy.toolchange-change** *(bit IN)* - Indicates that a tool has to be changed
- **gmoccapy.toolchange-changed** *(bit OUT)* - Indicates tool has been changed
- **gmoccapy.toolchange-confirm** *(bit IN)* - Confirms tool change

Usually they are connected like this for a manual tool change:

```
net tool-change gmoccapy.toolchange-change <= iocontrol.0.tool-change
net tool-changed gmoccapy.toolchange-changed <= iocontrol.0.tool-changed
net tool-prep-number gmoccapy.toolchange-number <= iocontrol.0.tool-prep-number
net tool-prep-loop iocontrol.0.tool-prepare <= iocontrol.0.tool-prepared
```

nota
Tenga cuidado, que estas conexiones deben hacerse en el archivo HAL postgui.

Tool Offset Pins These pins allow you to show the active tool offset values for X and Z in the tool information frame. You should know that they are only active after G43 has been sent.



Figura 10.16: Tool information area

- **gmoccapy.tooloffset-x** *(float IN)*

- **gmoccapy.tooloffset-z** *(float IN)*

nota
The tooloffset-x line is not needed on a mill, and will not be displayed on a mill with trivial kinematics.

Para mostrar los desplazamientos actuales, los pines tienen que estar conectados así en el archivo postgui HAL:

```
net tooloffset-x gmoccapy.tooloffset-x <= motion.tooloffset.x
net tooloffset-z gmoccapy.tooloffset-z <= motion.tooloffset.z
```



importante
Tenga en cuenta que GMOCAPPY se encarga de actualizar los offsets, enviando un G43 después de cualquier cambio de herramienta, **¡pero no en modo automático!**
De tal manera que escribir un programa ¡te hace responsable de incluir un G43 después de cada cambio de herramienta!

10.2.6. Auto Tool Measurement

GMOCCAPY offers an integrated auto tool measurement. To use this feature, you will need to do some additional settings and you may want to use the offered HAL pin to get values in your own NGC remap procedure.



importante
Antes de comenzar la primera prueba, ¡no olvide ingresar altura y velocidades de la sonda en la página de configuración! Ver Página de configuración de herramienta de medición.

También podría ser una buena idea echar un vistazo al video de la herramienta de medición: ver videos relacionados con la medición de herramientas.

Tool Measurement in GMOCCAPY is done a little bit different to many other GUIs. You should follow these steps:

1. Toque de su pieza en X e Y.
2. Mida la altura de su bloque desde la base donde está ubicado su interruptor de herramienta hasta la cara superior del bloque (incluyendo la mordaza etc.).
3. Presione el botón de altura del bloque e ingrese el valor medido.
4. Ir al modo automático e iniciar su programa.

Here is a small sketch:

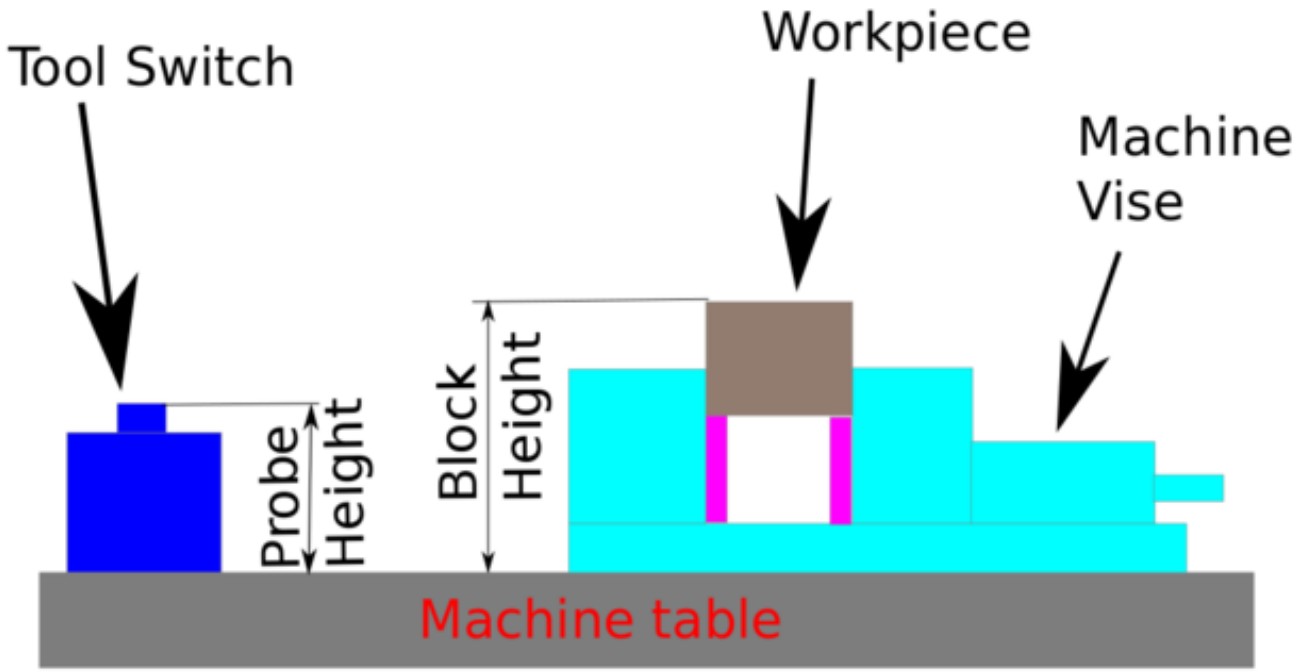


Figura 10.17: Tool measurement data

With the first given tool change the tool will be measured and the offset will be set automatically to fit the block height. The advantage of the GMOCCAPY way is, that you do not need a reference tool.

nota
Your program must contain a tool change at the beginning! The tool will be measured, even it has been used before, so there is no danger, if the block height has changed. There are several videos showing the way to do that on YouTube.

10.2.6.1. Provided Pins

GMOCCAPY offers five pins for tool measurement purposes. These pins are mostly used to be read from a G-code subroutine, so the code can react to different values.

- **gmoccapy.toolmeasurement** *(bit OUT)* - Enable or not tool measurement
- **gmoccapy.blockheight** *(float OUT)* - The measured value of the top face of the workpiece
- **gmoccapy.probeheight** *(float OUT)* - The probe switch height
- **gmoccapy.searchvel** *(float OUT)* - The velocity to search for the tool probe switch
- **gmoccapy.probevel** *(float OUT)* - The velocity to probe tool length

10.2.6.2. INI File Modifications

Modifique su archivo INI para incluir las secciones siguientes.

The RS274NGC Section

```
[RS274NGC]
# sub que se llama cuando ocurre un error durante el cambio de herramienta, no se necesita ↩
    en todas las configuraciones de máquina
ON_ABORT_COMMAND=O <on_abort> call

# The remap code
REMAP=M6  modalgroup=6 prolog=change_prolog ngc=change epilog=change_epilog
```

nota
Make sure INI_VARS and HAL_PIN_VARS are not set to 0. They are set to 1 by default.

The Tool Sensor Section The position of the tool sensor and the start position of the probing movement, all values are absolute coordinates, except MAXPROBE, which must be given in relative movement.

```
[TOOLSENSOR]
X = 10
Y = 10
Z = -20
MAXPROBE = -20
```

The Change Position Section This is not named TOOL_CHANGE_POSITION on purpose - **canon uses that name and will interfere otherwise.** The position to move the machine before giving the change tool command. All values are in absolute coordinates.

```
[CHANGE_POSITION]
X = 10
Y = 10
Z = -2
```

La Sección de Python The Python plug ins serves interpreter and task.

```
[PYTHON]
# The path to start a search for user modules
PATH_PREPEND = python
# The start point for all.
TOPLEVEL = python/toplevel.py
```

10.2.6.3. Needed Files

Primero cree un directorio ”python” en su carpeta de configuración. Desde `<su_directorio_linuxcnc-dev>` copie los archivos siguientes en la carpeta recién creada `config_dir/python`:

- `toplevel.py`
- `remap.py`
- `stdglue.py`

From `<your_linuxcnc-dev_directory>/configs/sim/gmoccapy/macros` copy

- `on_abort.ngc`
- `change.ngc`

al directorio especificado como SUBROUTINE_PATH, ver la Sección RS274NGC.

Open change.ngc with a editor and uncomment the following lines (49 and 50):

```
F #<_hal[gmoccapy.probevel]>
G38.2 Z-4
```

You may want to modify this file to fit more your needs.

10.2.6.4. Needed HAL Connections

Connect the tool probe in your HAL file like so:

```
net probe  motion.probe-input <= <your_input_pin>
```

The line might look like this:

```
net probe  motion.probe-input <= parport.0.pin-15-in
```

In your postgui.hal file add the following lines:

```
# The next lines are only needed if the pins had been connected before
unlinkp iocontrol.0.tool-change
unlinkp iocontrol.0.tool-changed
unlinkp iocontrol.0.tool-prep-number
unlinkp iocontrol.0.tool-prepared

# enlace al cambio de herramienta GMOCCAPY, para obtener la ventaja de la descripción en el ↩
     cuadro de diálogo de cambio de herramienta
net tool-change gmoccapy.toolchange-change <= iocontrol.0.tool-change
net tool-changed gmoccapy.toolchange-changed => iocontrol.0.tool-changed
net tool-prep-number gmoccapy.toolchange-number <= iocontrol.0.tool-prep-number
net tool-prep-loop iocontrol.0.tool-prepare <= iocontrol.0.tool-prepared
```

10.2.7. The Settings Page

To enter the page you will have to click on  and give an unlock code, which is **123** by default. If you want to change it at this time you will have to edit the hidden preference file, see the display section for details.

The page is separated in three main tabs:

10.2.7.1. Appearance

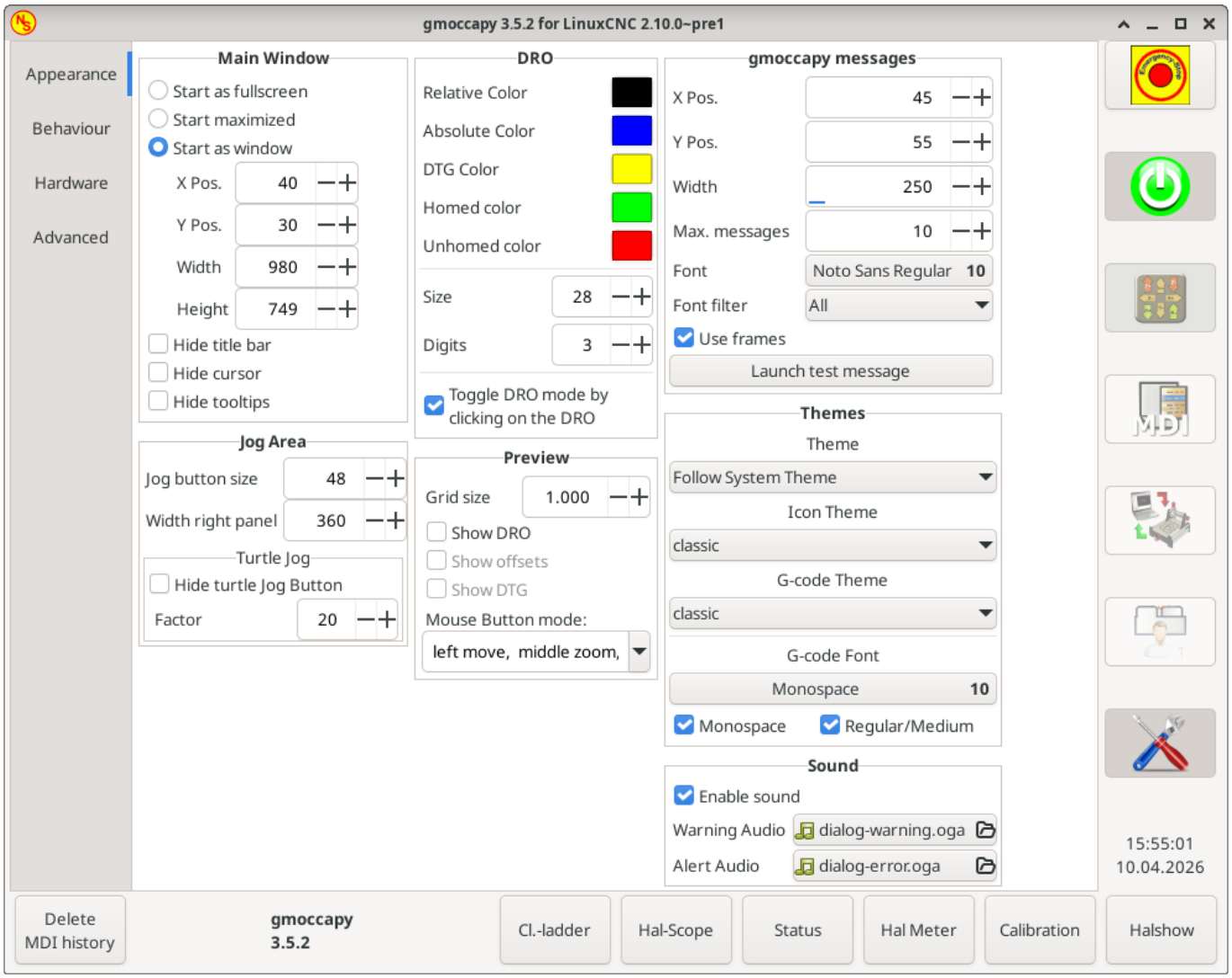


Figura 10.18: GMOCCAPY settings page Appearance

On this tab you will find the following options:

Main Window Here you can select how you wish the GUI to start. The main reason for this was the wish to get an easy way for the user to set the starting options without the need to touch code. You have three options:

- *Start as full screen*
- *Start maximized*
- *Iniciar como ventana* - Si selecciona Iniciar como ventana, se activarán los cuadros para establecer la posición y el tamaño. Una vez establecidos, la GUI se iniciará cada vez en el lugar y con el tamaño seleccionado. Sin embargo, el usuario puede cambiar el tamaño y la posición con el ratón, pero eso influirá en la configuración.
- *hide title bar* - Allows the title bar to be hidden. (default: title bar visible).
- *hide cursor* - Does allow to hide the cursor, what is very useful if you use a touch screen.

- *hide tooltips* - Hides the tool tips.

Jog Area

- *Jog button size* - Change the size of the job buttons in the manual mode page
- *Width right panel* - Change the width of the right part of window (= width of the DRO)

Turtle Jog

This settings will have influence on the jog velocities.

- *Hide turtle jog button* - Will hide the button right of the jog velocity slider. If you hide this button, please take care that the ”rabbit mode” is activated, otherwise you will not be able to jog faster than the turtle jog velocity, which is calculated using the turtle jog factor.
- *Factor de trote de tortuga* - Establece la escala para el modo trote de tortuga (botón presionado mostrando la tortuga). Si le pone un factor de 20, la velocidad máxima de trote de tortuga será 1/20 de la velocidad máxima de la máquina.

---

nota
This button can be controlled using the Turtle-Jog HAL Pin.

DRO Tiene la opción de seleccionar los colores de fondo de los diferentes estados del DRO. Por lo tanto, los usuarios que sufren de protanopia (debilidad roja/verde) pueden seleccionar los colores adecuados.

By default, the background colors are:

- *Relative Color* = black
- *Absolute Color* = blue
- *DTG Color* = yellow

The foreground color of the DRO can be selected with:

- *Homed Color* = green
- *Unhomed Color* = red

---

nota
Puede cambiar a través de los modos DRO (absoluto, relativo, distancia a recorrer) ¡haciendo clic en el DRO! Si haces clic en la letra del lado izquierdo del visualizador, una ventana emergente te permitirá establecer el valor de los ejes, lo que facilita el ajuste del valor, ya que no tendrás que pasar por el botón inferior de apagado.

- *tamaño* - Permite configurar el tamaño de la fuente DRO; el valor predeterminado es 28; si usa una pantalla más grande, es posible que desee aumentar el tamaño hasta 56. Si utiliza 4 ejes, el tamaño de fuente DRO será 3/4 del valor, debido a razones de espacio.
- *dígitos* - Establece el número de dígitos del DRO de 1 a 5.

---

nota
Imperial will show one digit more that metric. So if you are in imperial machine units and set the digit value to 1, you will get no digit at all in metric.

- *Toggle DRO mode by clicking on the DRO* - If not active, a mouse click on the DRO will not take any action.
  By default this checkbox is active, so every click on any DRO will toggle the DRO readout from actual to relative to DTG (distance to go).
  Neverthereless a click on the axis letter will open the popup dialog to set the axis value.

Preview

- *Grid Size* - Sets the grid size of the preview window. Unfortunately the size **has to be set in inches**, even if your machine units are metric. We do hope to fix that in a future release.

> **nota**
> The grid will not be shown in perspective view.

- *Mostrar DRO* - Mostrará el DRO también en el panel de vista previa. Se mostrará siempre en vista previa a tamaño completo.
- *Show DTG* - Will show the DTG (direct distance to end point) in the preview pane if Show DRO is active. Otherwise only in full size preview.
- *Show Offsets* - Will show the offsets in the preview pane when Show DRO is active. Otherwise only in full size preview.
- *Mouse Button Mode* - This combobox allows you to select the button behavior of the mouse to rotate, move or zoom within the preview:
  - left rotate, middle move, right zoom
  - left zoom, middle move, right rotate
  - left move, middle rotate, right zoom
  - left zoom, middle rotate, right move
  - left move, middle zoom, right rotate
  - left rotate, middle zoom, right move

Predeterminado es mover a la izquierda, zoom medio, rotación a la derecha.
La rueda del ratón seguirá ampliando la vista previa en todos los modos.

> **sugerencia**
> If you select an element in the preview, the selected element will be taken as rotation center point and in auto mode the corresponding code line will be highlighted.

Gmoccapy Messages

This will display small pop up windows displaying a message or error text, similar to the ones known from AXIS. You can delete a specific message by clicking on its close button. If you want to delete the last one, just hit the `WINDOWS` key on your keyboard, or delete all messages at once with `Control + Space`.

You are able to set some options:

- *X Pos* - The position of the top left corner of the message in X counted in pixel from the top left corner of the screen.
- *Y Pos* - The position of the top left corner of the message in Y counted in pixel from the top left corner of the screen.

- *Ancho* - El ancho del cuadro de mensaje.
- *Max. messages* - The maximum number of messages you want to see at once. If you set this to 10, the 11th message will delete the first one, so you will only see the last 10.
- *Fuente* - La fuente y el tamaño que desea utilizar para mostrar los mensajes.
- *Font filter* - A filter for the font chooser above.
- *Use frames* - If you activate the checkbox, each message will be displayed in a frame, so it is much easier to distinguish the messages. But you will need a little bit more space.
- *Launch test message*-button - It will show a message, so you can see the changes of your settings without the need to generate an error.

Themes

- *Theme* - This lets the user select what desktop theme to apply. By default ”Follow System Theme” is set.
- *Icon Theme* - Change the icon theme. Currently there are three themes available:
  - classic
  - material
  - material light

  For more information and how to create custom icon themes, see section Icon Theme.
- *G-code Theme* - Select a colour theme for the G-code viewer/editor.
- *G-code Font* - Select a font for the G-code viewer/editor. Default is ”monospace 10”.

Sound Select what error and messages sounds should be played and if sound should be played at all.

10.2.7.2. Behaviour

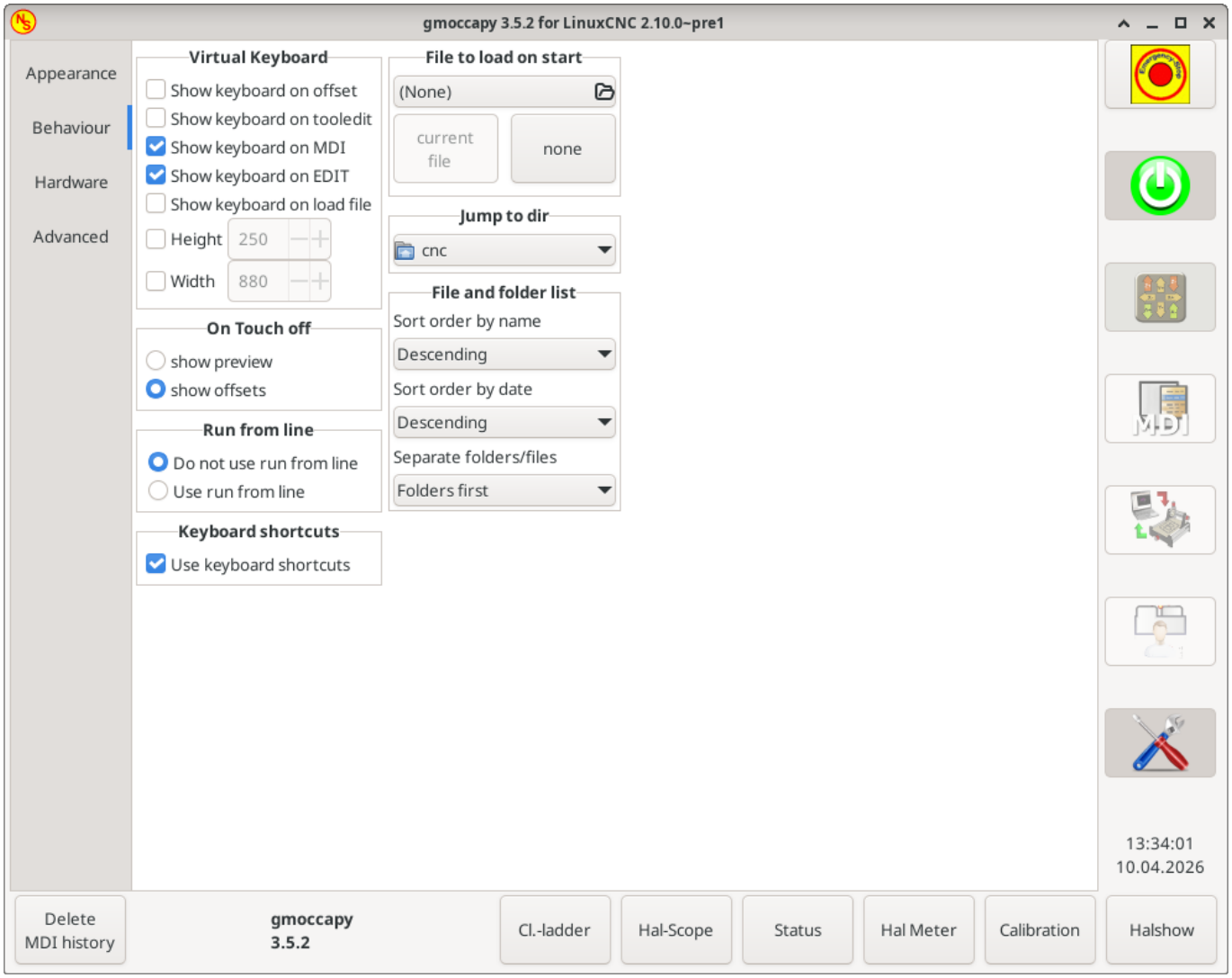


Figura 10.19: GMOCCAPY settings page Behaviour

Virtual Keyboard The checkboxes allow the user to select if he wants the on board keyboard to be shown immediately when entering the MDI Mode, the offset page, the tooledit widget or when open a program in the EDIT mode. The keyboard button on the bottom button list will not be affected by these settings, so you are able to show or hide the keyboard by pressing the button.

La configuración predeterminada es:

- *Show keyboard on offset* = False
- *Show keyboard on tooledit* = False
- *Show keyboard on MDI* = True
- *Show keyboard on EDIT* = True
- *Show keyboard on load file* = False

nota
If this section is not sensitive, you have not installed a virtual keyboard, supported ones are *onboard* and *matchbox-keyboard*.

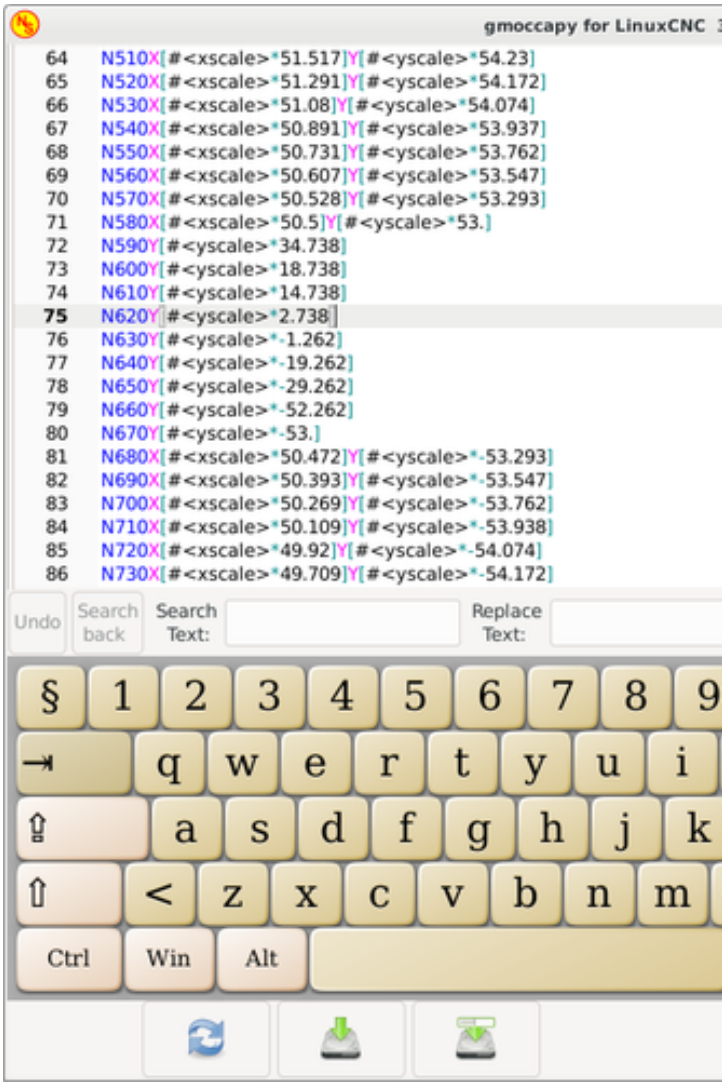
nota
If the keyboard layout is not correct, i.e. clicking Y gives Z, than the layout has not been set properly, related to your locale settings. For onboard it can be solved with a small batch file with the following content:

```
#!/bin/bash
setxkbmap -model pc105 -layout de -variant basic
```

Las letras ”de” son para Alemán, tendrá que configurar de acuerdo con su configuración regional (para español de España, **es**). Simplemente ejecute este archivo antes de iniciar LinuxCNC. También puede agregar un iniciador a su carpeta local.

```
./config/autostart
```

So that the layout is set automatically on starting.
For matchbox-keyboard you will have to make your own layout, for a German layout ask in the forum.



GMOCCAPY with Onboard keyboard in edit mode

On Touch Off This gives the option whether to show the preview tab or the offset page tab when you enter the touch off mode by clicking the corresponding bottom button.

- *show preview*
- *show offsets*

Run from Line You can allow or disallow the run from line. This will set the corresponding button insensitive (grayed out), so the user will not be able to use this option. The default is disable run from line.



aviso
No se recomienda usar *Ejecutar desde línea*, ya que LinuxCNC no comprobará ninguna línea anterior en el código antes de la línea de inicio. Así, errores o choques son mas que probables.

Keyboard shortcuts Algunos usuarios desean hacer jog usando los botones del teclado y hay otros que nunca lo permitirán. Así que todos pueden elegir si usarlos o no.
Los atajos de teclado están deshabilitados predeterminadamente. Pueden activarse con la casilla de verificación

- *Use keyboard shortcuts*



aviso
It is not recommended to use keyboard jogging, as it represents a serious risk for operator and machine.

Por favor, tenga cuidado si usa un torno, ya que los atajos serán diferentes. Ver la Sección específica de torno.

En general

- *F1* - Trigger Estop (will work even if keyboard shortcuts are disabled)
- *F2* - Toggle machine on/off
- *F3* - Manual mode
- *F5* - MDI mode
- *ESC* - Abort

En modo manual

- *Arrow_Left* or *NumPad_Left* - Jog X minus
- *Arrow_Right* or *NumPad_Right* - Jog X plus
- *Arrow_up* or *NumPad_Up* - Jog Y plus
- *Arrow_Down* or *NumPad_Down* - Jog Y minus
- *Page_Up* or *NumPad_Page_Up* - Jog Z plus
- *Page_Down* or *NumPad_Page_Down* - Jog Z minus

In Auto Mode

- *R* or *r* - Run program
- *P* or *p* - Pause program
- *S* or *s* - Resume program
- *Control* + *R* or *Control* + *r* - Reload the loaded file

Message handling (see Message behavior and appearance)

- *WINDOWS* - Delete last message
- *Control* + *Space* - Delete all messages

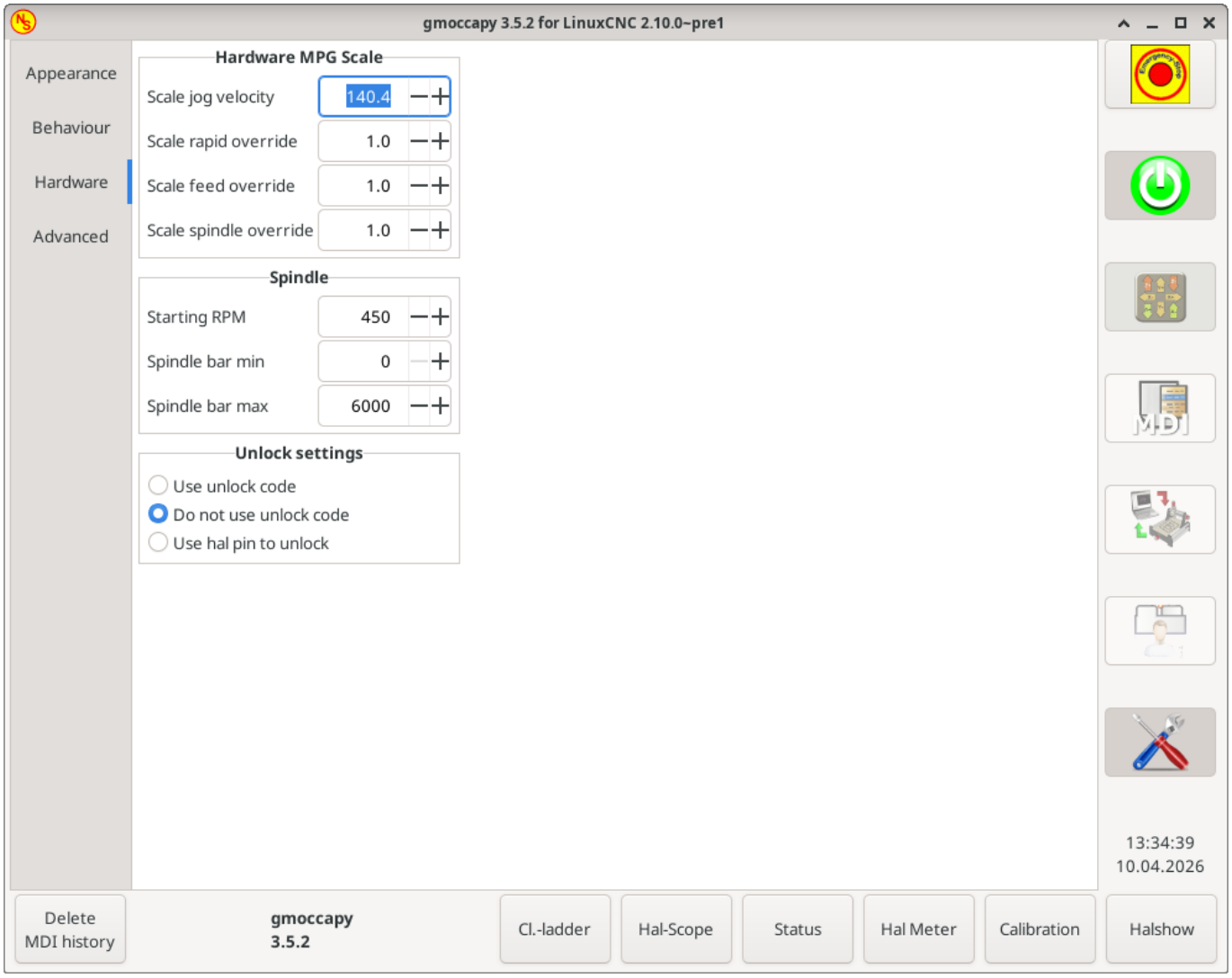
File to load on start up Select the file you want to be loaded on start up. If a file is loaded, it can be set by pressing the current button. To avoid that any program is loaded at start up, just press the None button.

La pantalla de selección de archivos utilizará los filtros que haya configurado en el archivo INI. Si no hay filtros, solo verá los **archivos NGC**. La ruta se establecerá de acuerdo con la configuración de INI en `[DISPLAY] PROGRAM_PREFIX`.

Jump to dir You can set here the directory to jump to if you press the corresponding button in the file selection dialog.

File and folder list For the file chooser page it is possible to change the sorting options with these parameters.

10.2.7.3. Hardware



Hardware MPG Scale For the different HAL pins to connect MPG wheels to, you may select individual scales to be applied. The main reason for this was my own test to solve this through HAL connections, resulting in a very complex HAL file. Imagine a user having an MPG Wheel with 100 ppr and he wants to slow down the max. vel. from 14000 to 2000 mm/min, that needs 12000 pulses, resulting in 120 turns of the wheel! Or an other user having a MPG Wheel with 500 ipr and he wants to set the spindle override which has limits from 50 to 120 % so he goes from min to max within 70 pulses, meaning not even 1/4 turn.

By default all scales are set using the calculation:

```
(MAX - MIN)/100
```

Husillo

- *Starting RPM* - Sets the rpm to be used if the spindle is started and no S value has been set.

> **nota**
> Este valor será predefinido de acuerdo con su configuración en `[DISPLAY]DEFAULT_SPINDLE_SPEED` de su archivo INI. Si cambia la configuración en la página de configuración, ese valor será el predeterminado desde ese momento; su archivo INI no será modificado.

- *Spindle bar min* and *Spindle bar max* - Sets the limits of the spindle bar shown in the INFO frame on the main screen.

  Los valores por defecto son:
  MIN = 0
  MAX = 6000

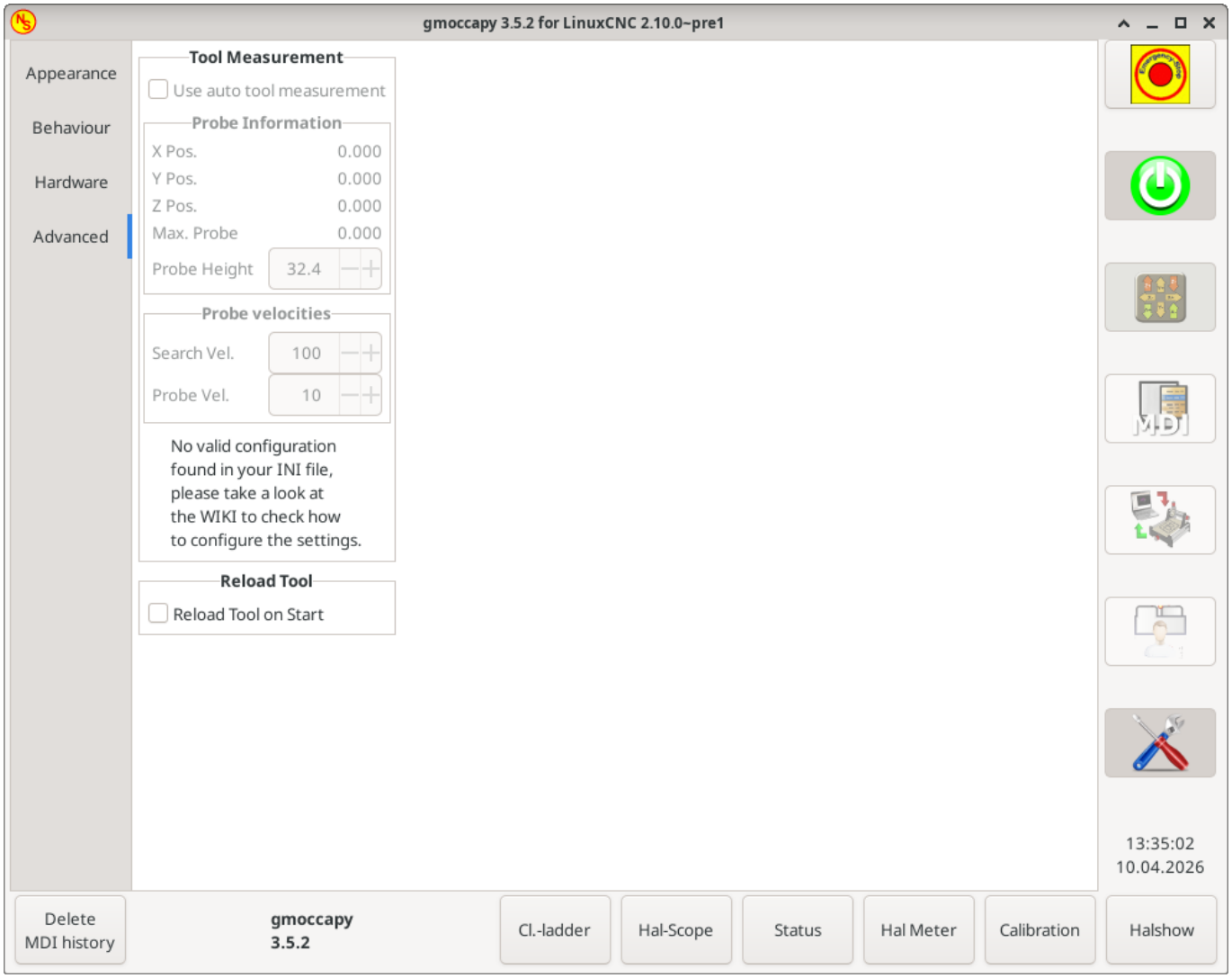
  nota
  It is no error giving wrong values. If you give a maximum of 2000 and your spindle makes 4000 RPM, only the bar level will be wrong on higher speeds than 2000 RPM.

Unlock settings There are three options to unlock the settings page:

- *Usar código de desbloqueo* - El usuario debe dar un código para entrar.
- *No usar código de desbloqueo* - No habrá verificación de seguridad.
- *Usar un pin para desbloquear* - El pin de hardware debe estar alto para desbloquear la configuración, ver pin de desbloqueo hardware.

El valor predeterminado es *usar el código de desbloqueo* (predeterminado = * 123 *).

10.2.7.4. Advanced Settings



Tool Measurement

Please check Auto Tool Measurement

---

nota
If this part is not sensitive, you do not have a valid INI file configuration to use tool measurement.

- *Use auto tool measurement* - If checked, after each tool change, a tool measurement will be done, the result will be stored in the tool table and a G43 will be executed after the change.

Probe Information

Las información siguiente se toma de su archivo INI y se debe proporcionar en coordenadas absolutas:

- *X Pos.* - La posición X del interruptor de herramienta.
- *Y Pos.* - La posición Y del interruptor de herramienta.
- *Z Pos.* - La posición Z del interruptor de la herramienta, iremos como movimiento rápido a esta coordenada.
- *Max. Probe* - The distance to search for contact, an error will be launched, if no contact is given in this range. The distance has to be given in relative coordinates, beginning the move from Z Pos., so you have to give a negative value to go down!
- *Probe Height* - The height of your probe switch, you can measure it. Just touch off the base where the probe switch is located and set that to zero. Then make a tool change and watch the tool_offset_z value, that is the height you must enter here.

Probe velocities

- *Search Vel.* - The velocity to search for the tool switch. After contact the tool will go up again and then goes towards the probe again with probe vel, so you will get better results.
- *Vel. Sonda* - Es la velocidad para el segundo movimiento hacia el interruptor. Debe ser más lento para obtener mejores resultados. En modo sim, esto es comentado en macros/change.ngc, de lo contrario el usuario tendría que hacer clic dos veces en el botón de la sonda.

Reload Tool::

- *Reload Tool on Start* - Loads the last tool on start after homing.

If checked, the tool in spindle will be saved on each change in the preference file, making it possible to reload the last mounted tool on start up. The tool will be loaded after all axes are homed, because before it is not allowed to execute MDI commands. If you use NO_FORCE_HOMING you can not use this feature, because the needed all_homed_signal will never be emitted.

10.2.8. Icon Theme

Icon themes are used to customize the look and feel of GMOCCAPY's icons.

GMOCCAPY ships with three different icon themes:

- *classic* - The classic GMOCCAPY icons.
- *material* - A modern icon theme inspired by Google's Material Icons that automatically adopts its coloring from the selected desktop theme.
- *material-light* - Derived from material but optimized for light desktop themes.

The icon theme used in GMOCCAPY is a regular GTK icon theme that follows the freedestktop icon theme specification. Thus every valid GTK icon theme can be used as GMOCCAPY icon theme as long as it contains the required icons.

GMOCCAPY scans the following directories for icon themes:

- linuxcnc/share/gmoccapy/icons
- ~/.icons

10.2.8.1. Custom Icon Theme

Creating a custom icon theme is pretty easy. All you need is a text editor and of course the desired icons as pixel or vector graphics. Details about how exactly an icon theme is built can be found at the Freedesktop Icon Theme Specification.

Start by creating an empty directory with the name of the icon theme. Place the directory in one of GMOCCAPY's icon theme directories. Then we need a file called index.theme in the root folder of our icon theme which contains the required metadata for the theme. That's a simple text file with at least the following sections:

- [Icon Theme]

```
[Icon Theme]
Name=YOUR_THEME_NAME
Comment=A DESCRIPTION OF YOUR THEME
Inherits=hicolor
Directories=16x16/actions,24x24/actions,32x32/actions,48x48/actions,scalable/actions
```

  - Name: The name of your icon theme.
  - Comment: A description of your icon theme.
  - Inherits: A icon theme can derive from another icon theme, the default is hicolor.
  - Directories: A comma separated list of all the directories of your icon theme.
    Each directory usually contains all the icons of the theme in a specific size, for example 16x16/actions should contain all icons with the category "actions" in the size 16x16 pixels as pixel-graphics (e.g. png files). A special case is the directory called "scalable/actions", this contains scalable icons not tied to a specific size (e.g. svg files).
    By supplying different sized versions of the icons, we can guarantee a nice looking icon if different sizes and we also have the ability to change the icon according to its size, for example a 64x64 px sized icon may contain more details than its 16x16 px version.

- For each directory we also have to write a section in the index.theme file:

```
[16x16/actions]
Size=16
Type=Fixed
Context=Actions

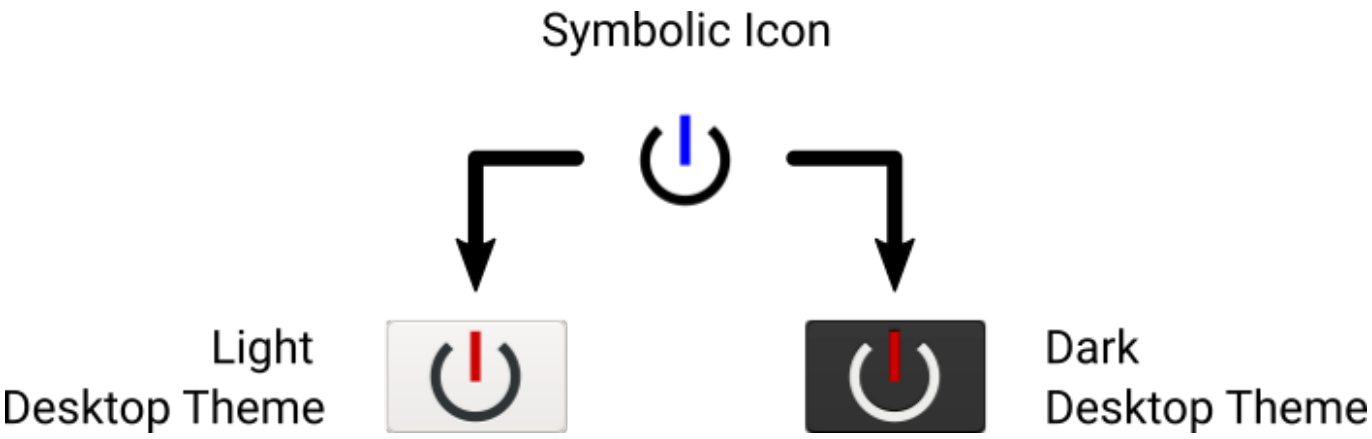
[scalable/actions]
Size=48
Type=Scalable
Context=Actions
```

  - Size: Nominal icon size in this directory
  - Type: Fixed, Threshold or Scalable
  - Context: Intended "category" of icons

Basically that's everything needed to create a custom icon theme.

10.2.8.2. Symbolic Icons

Symbolic icons are a special type of icon, usually a monochrome image. The special feature of symbolic icons is that the icons are automatically colored at runtime to match the desktop theme. That way, icon themes can be created that work well with dark and also light desktop themes (in fact, that's not always the best option, that's why a dedicated "material-light" theme exists).



To make use of the symbolic feature, a icon file has to have the suffix .symbolic.$ext (where $ext is the regular file extension like png) for example "power_on.symbolic.png".

With that name, GTK treats this image as symbolic icon and applies some recoloring when loading the icon. There are only four colors allowed to use:

Color	Hex Code	Descripción
black	`#000000`	Primary color, gets changed to match the desktop themes primary color.
red	`#ff0000`	Success: this color indicates "success" (usually something green'ish).
green	`#00ff00`	Warning: this color indicates "warning" (usually something yellow/orange'ish).
blue	`#0000ff`	Error: this color indicates "error" (usually something red'ish).

---

sugerencia
Examples of symbolic icons can be found at `linuxcnc/share/gmoccapy/icons/material`.

10.2.9. Lathe Specific Section

If in the INI file `LATHE = 1` is given, the GUI will change its appearance to the special needs for a lathe. Mainly the Y axis will be hidden and the jog buttons will be arranged in a different order.

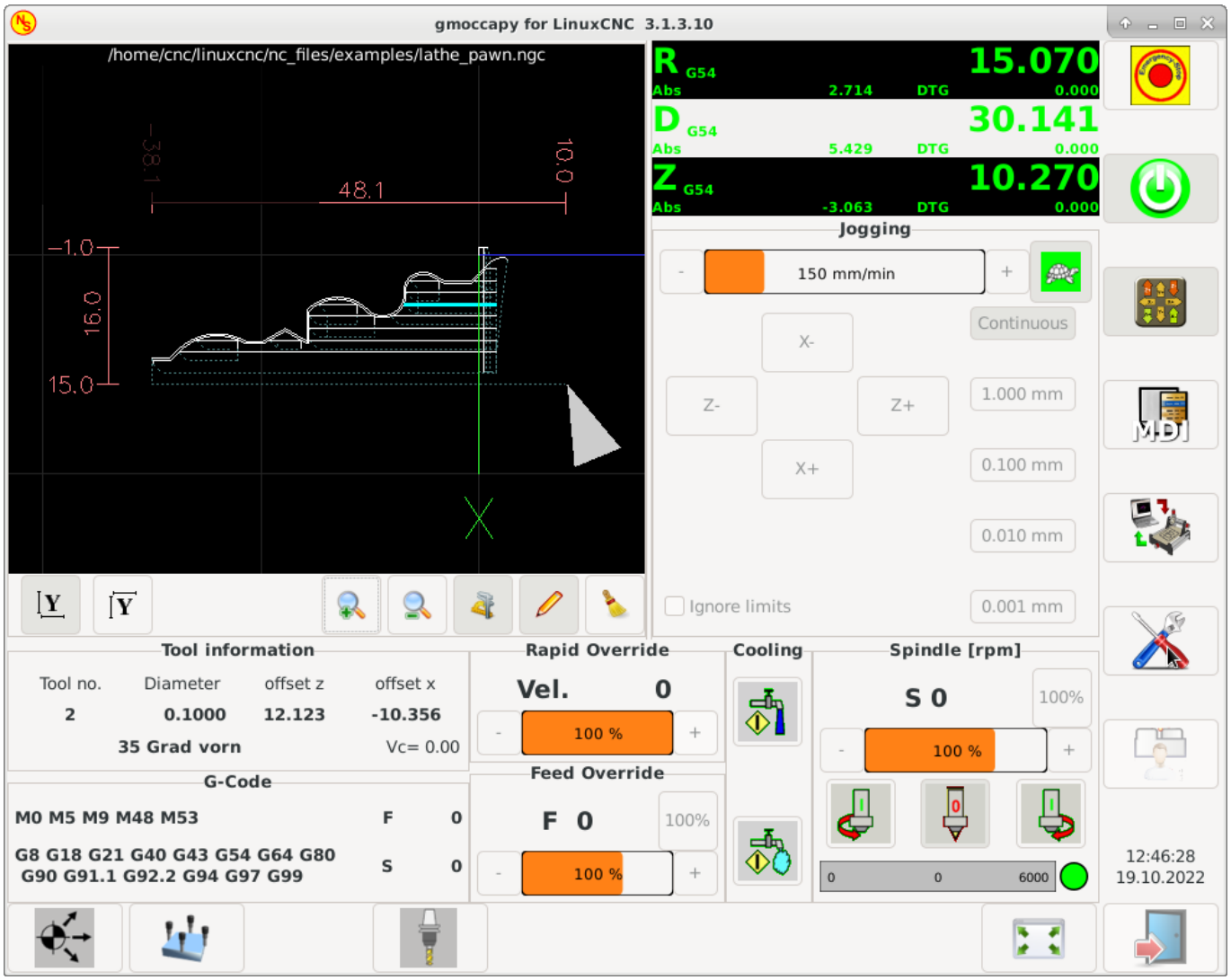


Figura 10.20: Normal Lathe

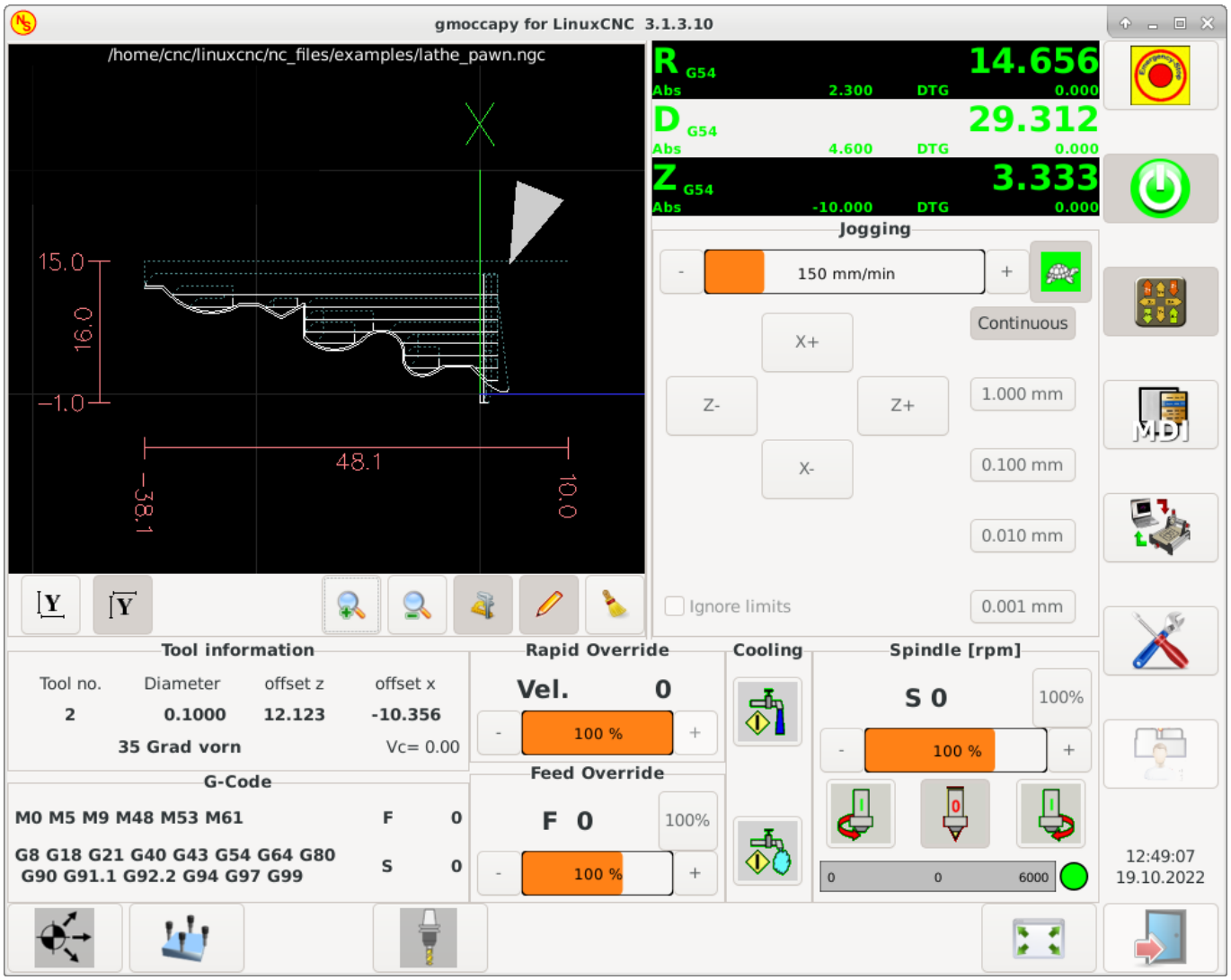


Figura 10.21: Back Tool Lathe

As you see the R DRO has a black background and the D DRO is gray. This will change according to the active G-code G7 or G8. The active mode is visible by the black background, meaning in the shown images G8 is active.

The next difference to the standard screen is the location of the jog buttons. X and Z have changed places and Y is gone. You will note that the X+ and X- buttons changes there places according to normal or back tool lathe.

Also the keyboard behavior will change:

Normal Lathe:

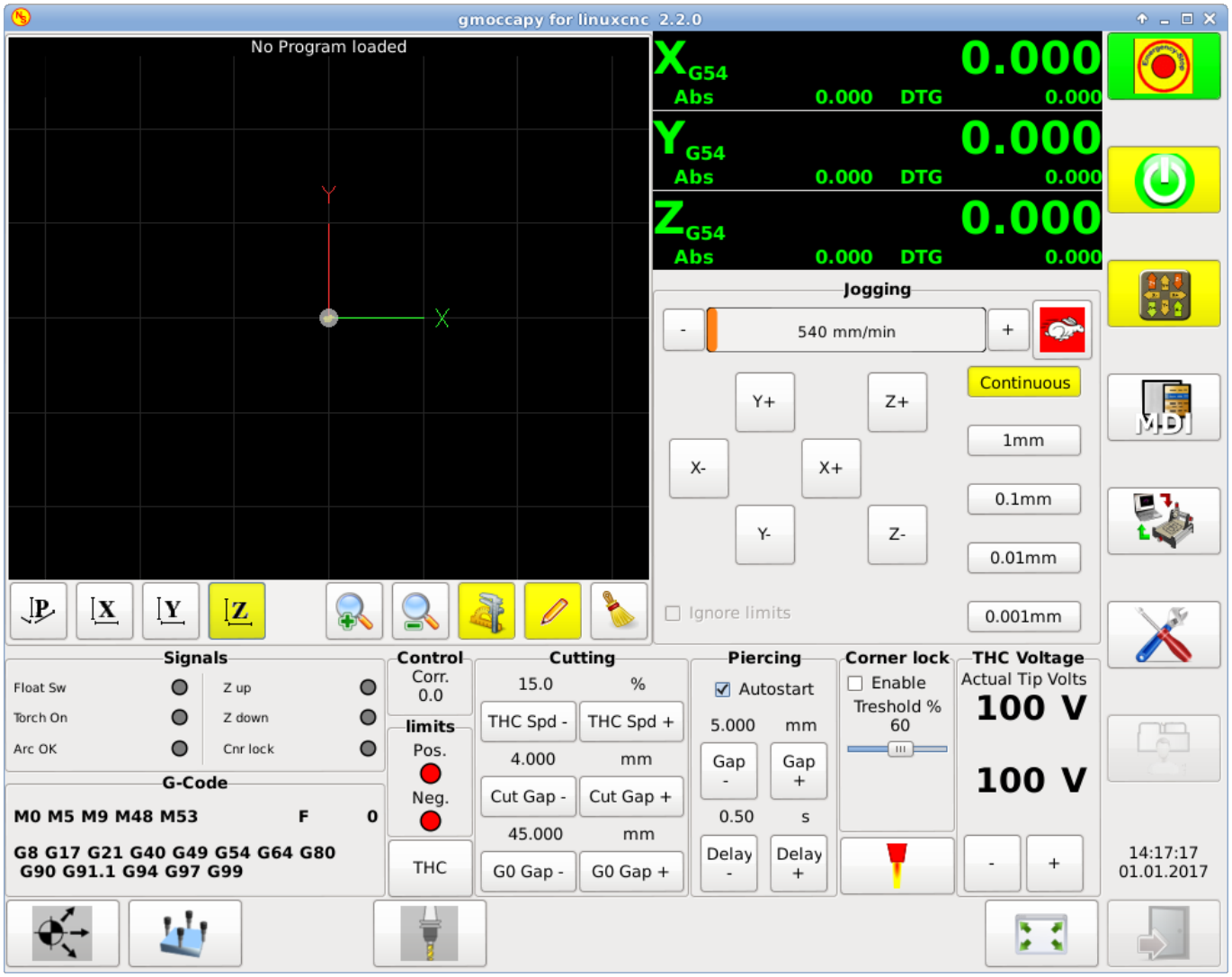
- *Arrow_Left* or *NumPad_Left* - Jog Z minus
- *Arrow_Right* or *NumPad_Right* - Jog Z plus
- *Arrow_up* or *NumPad_Up* - Jog X minus
- *Arrow_Down* or *NumPad_Down* - Jog X plus

Back Tool Lathe:

- *Arrow_Left* or *NumPad_Left* - Jog Z minus
- *Arrow_Right* or *NumPad_Right* - Jog Z plus
- *Arrow_up* or *NumPad_Up* - Jog X plus
- *Arrow_Down* or *NumPad_Down* - Jog X minus

The tool information frame will show not only the Z offset, but also the X offset and the tool table is showing all lathe relevant information.

10.2.10. Plasma Specific Section



Hay un muy buen WIKI, que en realidad está creciendo, mantenido por Marius. Vea página wiki de Plasma.

10.2.11. Videos on YouTube

Below is a series of videos that show GMOCCAPY in action. Unfortunately, these videos don’t show the latest version of GMOCCAPY, but the way to use it will still be the same as in the current version. I will update the videos as soon as possible.

10.2.11.1. Basic Usage

https://youtu.be/O5B-s3uiI6g

10.2.11.2. Simulated Jog Wheels

https://youtu.be/ag34SGxt97o

10.2.11.3. Settings Page

https://youtu.be/AuwhSHRJoiI

10.2.11.4. Simulated Hardware Button

German: https://youtu.be/DTqhY-MfzDE
English: https://youtu.be/ItVWJBK9WFA

10.2.11.5. User Tabs

https://youtu.be/rG1zmeqXyZI

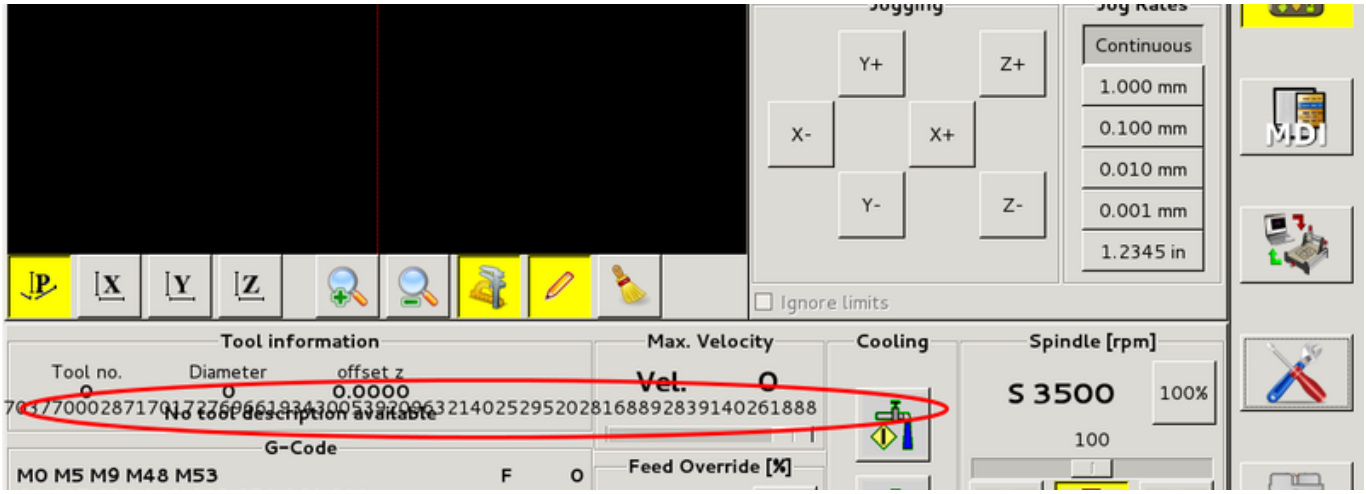
10.2.11.6. Tool Measurement Videos

Auto Tool Measurement Simulation: https://youtu.be/rrkMw6rUFdk
Auto Tool Measurement Screen: https://youtu.be/Z2ULDj9dzvk
Auto Tool Measurement Machine: https://youtu.be/1arucCaDdX4

10.2.12. Known Problems

10.2.12.1. Strange numbers in the info area

If you get strange numbers in the info area of GMOCCAPY like:



Ha creado su archivo de configuración con una versión anterior de StepConfWizard. Este ha hecho una entrada incorrecta en el archivo INI bajo [TRAJ]MAX_LINEAR_VELOCITY = xxx. Cambia esa entrada a MAX_VELOCITY = xxx.

10.2.12.2. Not ending macro

If you use a macro without movement, like this one:

```
 o<zeroxy> sub

G92.1
G92.2
G40

G10 L20 P0 X0 Y0

o<zeroxy> endsub
m2
```

GMOCCAPY will not see the end of the macro, because the interpreter needs to change its state to IDLE, but the macro does not even set the interpreter to a new state. To avoid that just add a G4 P0.1 line to get the needed signal. The correct macro would be:

```
 o<zeroxy> sub

G92.1
G92.2
G40

G10 L20 P0 X0 Y0

G4 P0.1

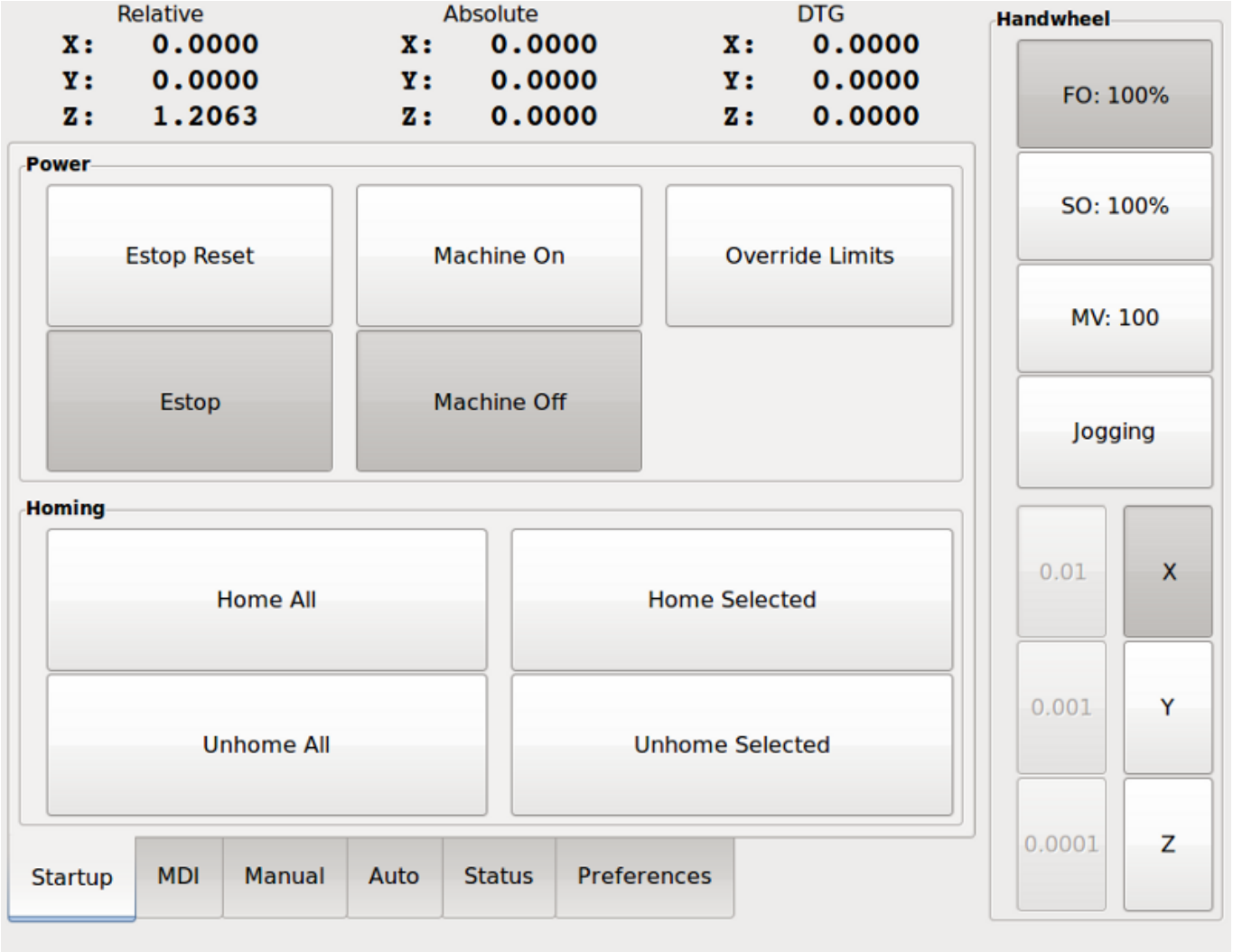
o<zeroxy> endsub
m2
```

10.3. The Touchy Graphical User Interface

Touchy is a user interface for LinuxCNC meant for use on machine control panels, and therefore does not require keyboard or mouse.

It is meant to be used with a touch screen, and works in combination with a wheel/MPG and a few buttons and switches.

La pestaña *Volante* tiene botones radio para seleccionar entre las funciones de *Porcentaje de alimentación*, *Porcentaje de husillo*, *Velocidad máxima* y *Jogging* para la entrada del volante/MPG. También están previstos botones radio para la selección de eje e incrementos para jog.



10.3.1. Panel Configuration

10.3.1.1. HAL connections

Touchy looks in the INI file, under the heading *[HAL]* for entries of *POSTGUI_HALFILE=<filename>*. Typically *<filename>* would be *touchy_postgui.hal*, but can be any legal filename. These commands are executed after the screen is built, guaranteeing the widget HAL pins are available. You can have multiple line of *POSTGUI_HALFILE=<filename>* in the INI. Each will be run one after the other in the order they appear in the INI file.

---

nota
Touchy used to require that you create a file named *touchy.hal* in your configuration directory (the directory your INI file is in). For legacy reasons this will continue to work, but INI based postgui files are preferred.

For more information on HAL files and the net command see the HAL Basics.

Touchy has several output pins that are meant to be connected to the motion controller to control wheel jogging:

- *touchy.jog.wheel.increment*, which is to be connected to the *axis*.N.*jog-scale* pin of each axis *N*.

- *touchy.jog.wheel*.N, which is to be connected to *axis*.N.*jog-enable* for each axis *N*.

nota
N represents the axis number 0-8.

- In addition to being connected to *touchy.wheel-counts*, the wheel counts should also be connected to *axis*.N.*jog-counts* for each axis *N*. If you use HAL component *ilowpass* to smooth wheel jogging, be sure to smooth only *axis*.N.*jog-counts* and not *touchy.wheel-counts*.

Controles requeridos

- Botón de cancelación (contacto momentáneo) conectado al pin HAL *touchy.abort*.
- Botón de inicio de ciclo (contacto momentáneo) conectado a *touchy.cycle-start*.
- Rueda/MPG, conectada a *touchy.wheel-counts* y pines de movimiento como se describe anteriormente.
- Bloque único (interruptor de palanca) conectado a *touchy.single-block*.

Controles opcionales

- For continuous jog, one center-off bidirectional momentary toggle (or two momentary buttons) for each axis, hooked to *touchy.jog.continuous.x.negative*, *touchy.jog.continuous.x.positive*, etc.
- If a quill up button is wanted (to jog Z to the top of travel at top speed), a momentary button connected to *touchy.quill-up*.

Lámparas de panel opcionales

- *touchy.jog.active* shows when the panel jogging controls are live.
- *touchy.status-indicator* is on when the machine is executing G-code, and flashes when the machine is executing but is in pause/feedhold.

10.3.1.2. Recommended for any setup

- Estop button hardwired in the estop chain

10.3.2. Setup

10.3.2.1. Enabling Touchy

Para usar Touchy, en la sección *[DISPLAY]* de su archivo INI cambie la línea del selector a *DISPLAY = touchy*.

10.3.2.2. Preferences

When you start Touchy the first time, check the Preferences tab. If using a touchscreen, choose the option to hide the pointer for best results.

The Status Window is a fixed height, set by the size of a fixed font. This can be affected by the Gnome DPI, configured in System / Preferences / Appearance / Fonts / Details. If the bottom of the screen is cut off, reduce the DPI setting.

All other font sizes can be changed on the Preferences tab.

10.3.2.3. Macros

Touchy puede invocar macros palabra-O usando la interfaz MDI. Para configurar esto, en la sección *[MACROS]* del archivo INI, agregue una o más líneas *MACRO*. Cada una debe tener el formato siguiente:

```
MACRO=increment xinc yinc
```

In this example, increment is the name of the macro, and it accepts two parameters, named xinc and yinc.

Now, place the macro in a file named *increment.ngc*, in the *PROGRAM_PREFIX* directory or any directory in the *SUBROUTINE_PATH*.

It should look like:

```
O<increment> sub
G91 G0 X#1 Y#2
G90
O<increment> endsub
```

Notice the name of the sub matches the file name and macro name exactly, including case.

When you invoke the macro by pressing the Macro button on the MDI tab in Touchy, you can enter values for xinc and yinc. These are passed to the macro as *#1* and *#2* respectively. Parameters you leave empty are passed as value 0.

If there are several different macros, press the Macro button repeatedly to cycle through them.

In this simple example, if you enter -1 for xinc and press cycle start, a rapid *G0* move will be invoked, moving one unit to the left.

This macro capability is useful for edge/hole probing and other setup tasks, as well as perhaps hole milling or other simple operations that can be done from the panel without requiring specially-written G-code programs.

10.4. Gscreen

10.4.1. Introducción

Gscreen is an infrastructure to display a custom screen to control LinuxCNC. Gscreen borrows heavily from GladeVCP. GladeVCP uses the GTK widget editor GLADE to build virtual control panels (VCP) by point and click. Gscreen combines this with Python programming to create a GUI screen for running a CNC machine.

Gscreen is customizable if you want different buttons and status LEDs. Gscreen supports GladeVCP which is used to add controls and indicators. To customize Gscreen you use the Glade editor. Gscreen is not restricted to adding a custom panel on the right or a custom tab it is fully editable.

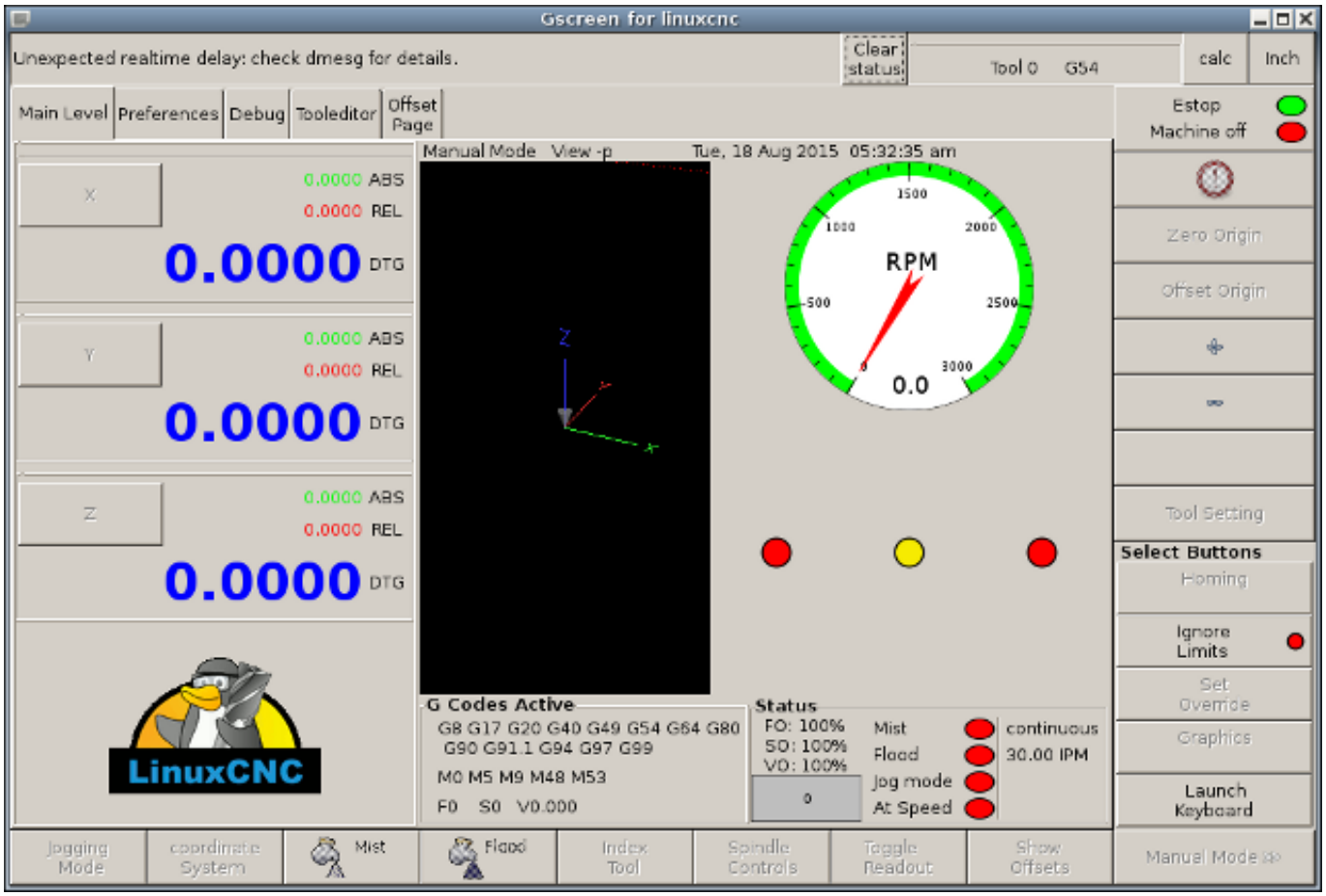


Figura 10.22: Gscreen Default Screen

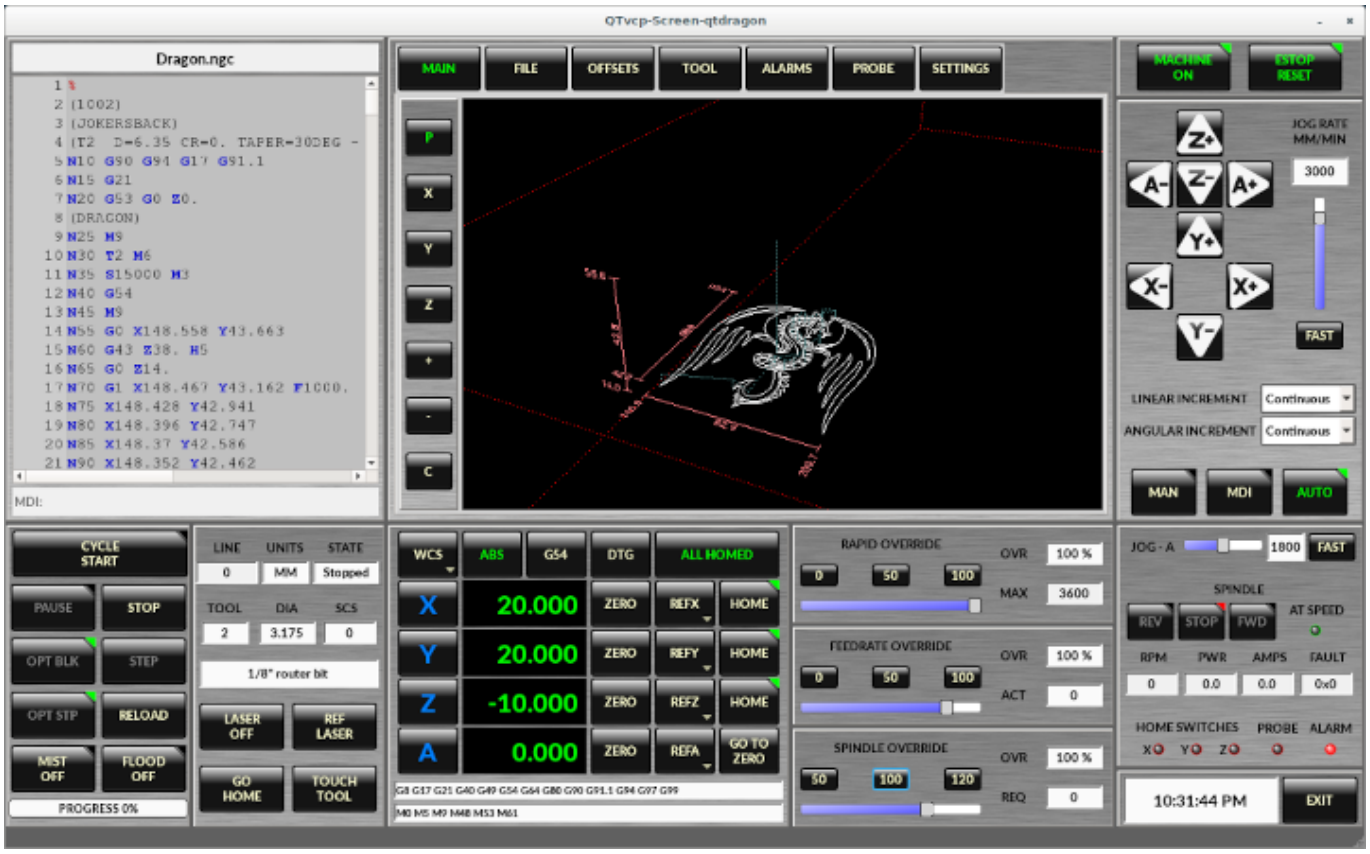


Figura 10.23: Gscreen Silverdragon Screen

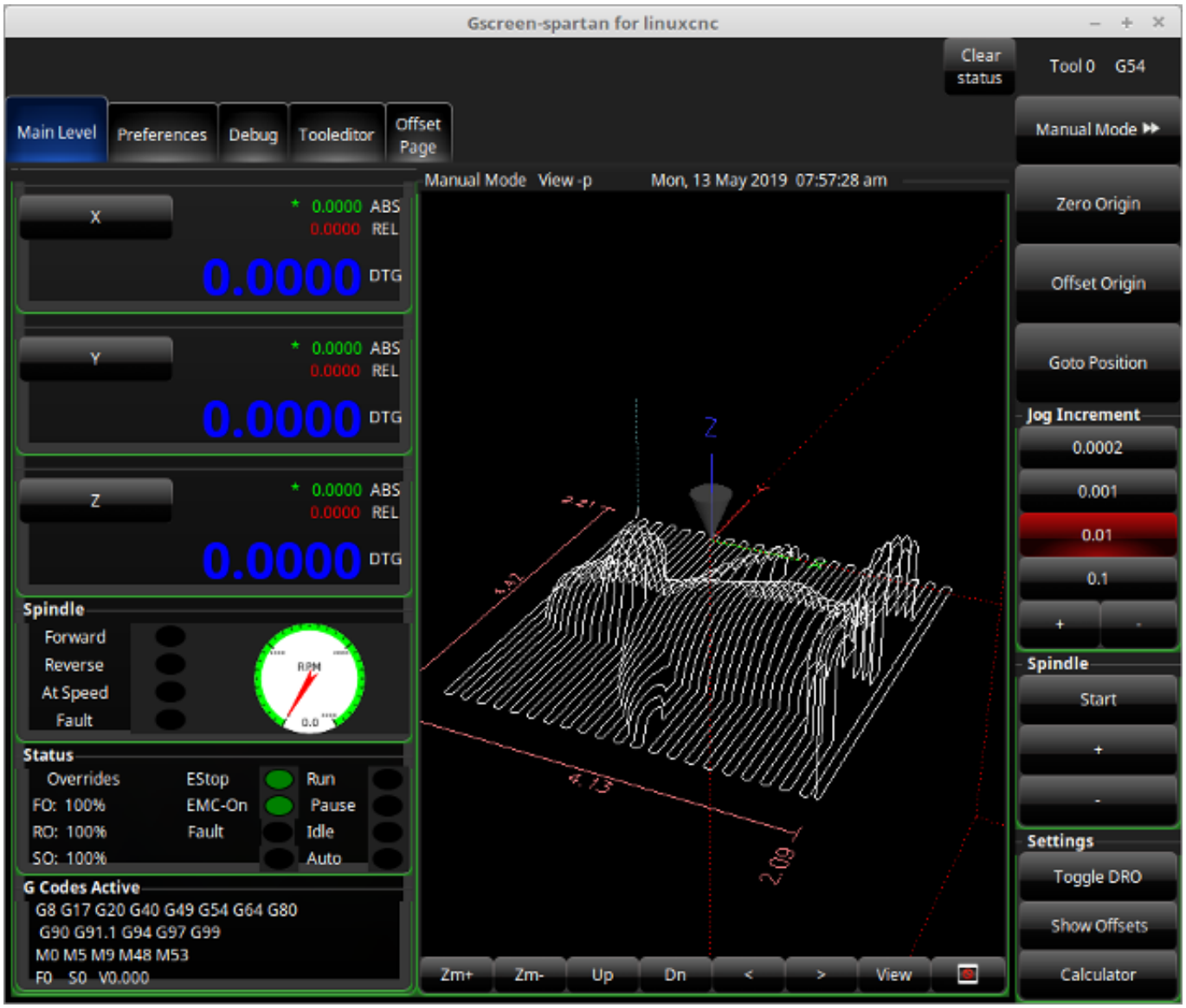


Figura 10.24: Gscreen Spartan Screen

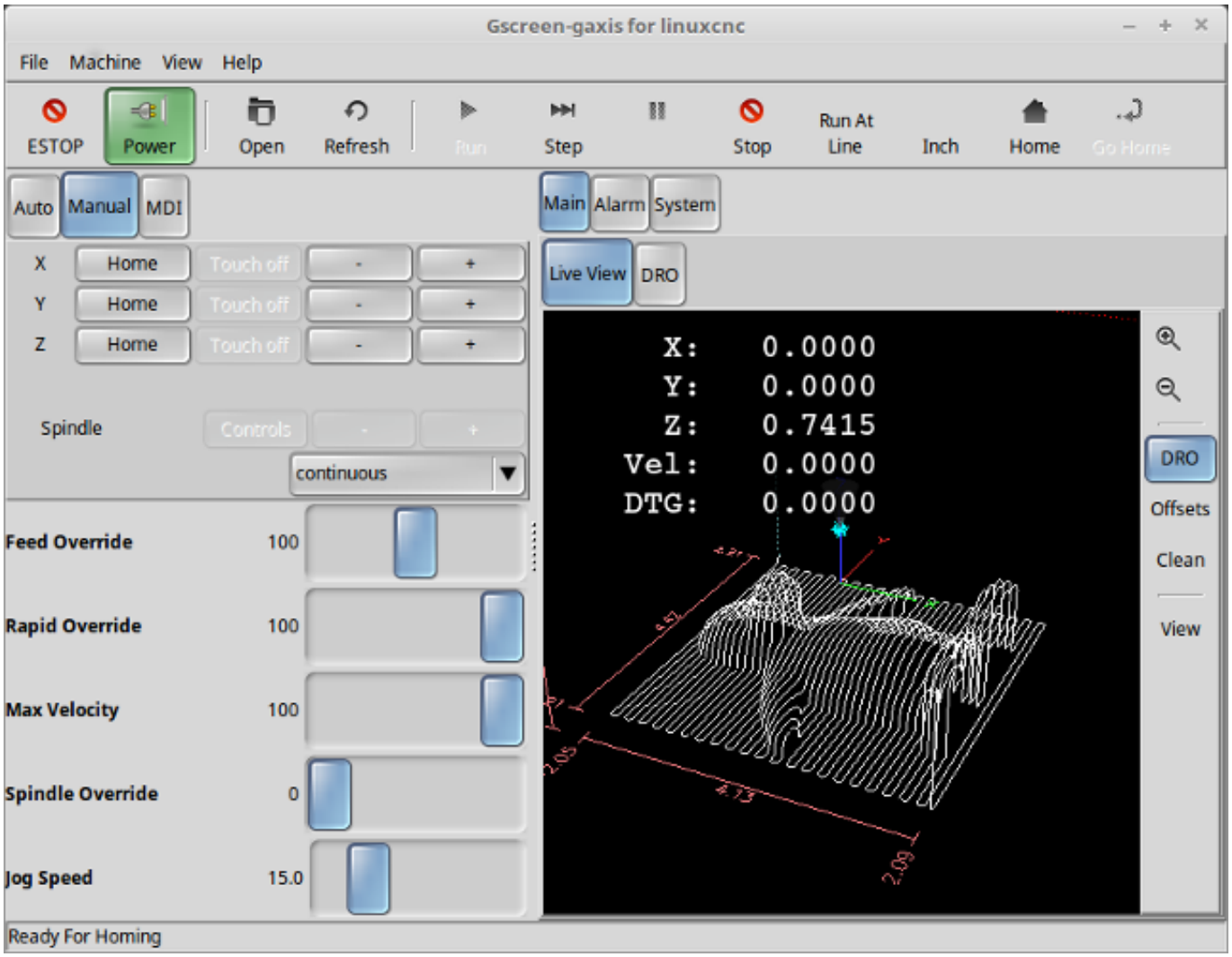


Figura 10.25: Gscreen Gaxis Screen



Figura 10.26: Gscreen Industrial Screen

Gscreen is based on *Glade* (the editor), *PyGTK* (the widget toolkit), and *GladeVCP* (LinuxCNC's connection to Glade and PyGTK). GladeVCP has some special widgets and actions added just for LinuxCNC A widget is just the generic name used for the buttons, sliders, labels etc of the PyGTK toolkit.

10.4.1.1. Glade File

A Glade file is a text file organized in the XML standard that describes the layout and the widgets of the screen. PyGTK uses this file to actually display and react to those widgets. The Glade editor makes it relatively easy to build and edit this file You must use the Glade 3.38.2 editor that uses the GTK3 widgets.

10.4.1.2. PyGTK

PyGTK is the Python binding to GTK. GTK is the *toolkit* of visual widgets, it is programmed in C. PyGTK uses Python to *bind* with GTK.

10.4.2. GladeVCP

GladeVCP binds LinuxCNC, HAL, PyGTK and Glade all together. LinuxCNC requires some special widgets so GladeVCP supplies them. Many are just HAL extensions to existing PyGTK widgets. GladeVCP creates the HAL pins for the special widgets described in the Glade file. GladeVCP also allows one to add Python commands to interact with the widgets, to make them do things not available in their default form. If you can build a GladeVCP panel you can customize Gscreen!

10.4.2.1. Descripción general

There are two files that can be used, individually or in combination to add customizations. Local Glade files and handler files. Normally Gscreen uses the stock Glade file and possibly a handler file (if using a sample *skin*). You can specify Gscreen to use *local* Glade and handler files. Gscreen looks in the folder that holds all the configuration files for the configuration you selected.

Local Glade Files If present, local Glade files in the configuration folder will be loaded instead of the stock Glade files. Local Glade files allow you to use your customized designs rather then the default screens. There is a switch in the INI file to set set the base name: -c name so Gscreen looks for MYNAME.glade and MYNAME_handler.py.

You can tell Gscreen to just load the Glade file and not connect its internal signals to it. This allows gscreen to load any GTK builder saved Glade file. This means you can display a completely custom screen, but also requires you to use a handler file. Gscreen uses the Glade file to define the widgets, so it can show and interact with them. Many of them have specific names, others have Glade given generic names. If the widget will be displayed but never changed then a generic name is fine. If one needs to control or interact with the widget then a hopefully purposeful name is given (all names must be unique). Widgets can also have signals defined for them in the GLADE editor. It defines what signal is given and what method to call.

Modificación de Skins de serie If you change the name of a widget, Gscreen might not be able to find it. If this widget is referenced to from Python code, at best this makes the widget not work anymore at worst it will cause an error when loading Gscreen's default screens don't use many signals defined in the editor, it defines them in the Python code. If you move (cut and paste) a widget with signals, the signals will not be copied. You must add them again manually.

Handler Files A handler file is a file containing Python code, which Gscreen adds to its default routines. A handler file allows one to modify defaults, or add logic to a Gscreen skin without having to modify Gscreen proper. You can combine new functions with Gscreen's function to modify behavior as you like. You can completely bypass all of Gscreen's functions and make if work completely differently. If present a handler file named gscreen_handler.py (or MYNAME_handler.py if using the INI switch) will be loaded and registered only one file is allowed Gscreen looks for the handler file, if found it will look for specific function names and call them instead of the default ones. If adding widgets you can set up signal calls from the Glade editor to call routines you have written in the handler file. In this way you can have custom behavior. Handler routines can call Gscreen's default routines, either before or after running its own code. In this way you can tack on extra behavior such as adding a sound. Please see the GladeVCP Chapter for the basics to GladeVCP handler files. Gscreen uses a very similar technique.

Themes Gscreen usa el kit de herramientas PyGTK para mostrar la pantalla. PyGTK es el enlace del lenguaje Python con GTK. GTK soporta *temas*. Los temas son una forma de modificar la apariencia de los widgets en la pantalla. Por lo tanto, el color o el tamaño de los botones y los controles deslizantes se pueden cambiar utilizando temas. Hay muchos temas GTK disponibles en la web. Los temas también se pueden personalizar para modificar la visualizacion de widgets particulares. Esto vincula el archivo de tema al archivo de Glade más firmemente. Algunas de las máscaras de pantalla de muestra permiten al usuario seleccionar cualquiera de los temas en el sistema. La muestra *gscreen* es un ejemplo. Algunos cargarán el tema que tiene el mismo nombre en el archivo de configuración. La muestra *gscreen-gaxis* es un ejemplo. Esto se hace poniendo la carpeta de temas en la carpeta de configuración que tiene los archivos INI y HAL y nombrándola: SCREENNAME_theme (siendo SCREENNAME el nombre

base de los archivos, por ejemplo, gaxis_theme). Dentro de esta carpeta hay otra llamada gtk-2.0, dentro de la cual están los archivos del tema. Si agrega este archivo, Gscreen se establecerá de forma predeterminada en este tema al inicio. gscreen-gaxis tiene un tema personalizado de muestra que busca ciertos widgets con nombre y cambia el comportamiento visual de esos widgets específicos. Los botones Estop y máquina ON utilizan colores diferentes que el resto de los botones para que se destaquen . Esto se hace en el archivo handler dándoles nombres específicos y agregando comandos específicos en el archivo gtkrc del tema. Para obtener información sobre la temática GTK (el tema de muestra utiliza el motor de tema pixmap). Vea: Temas GTK, Motor de Tema Pixmap.

10.4.2.2. Build a GladeVCP Panel

Gscreen is just a big complicated GladeVCP panel, with Python code to control it. To customize it we need the Glade file loaded in the Glade editor.

Installed LinuxCNC Si tiene LinuxCNC 2.6+ instalado en Ubuntu 10.04, simplemente inicie el editor Glade desde el menú de aplicaciones o desde el terminal. Las nuevas versiones de Linux requieren que instale Glade 3.8.0 - 3.8.6 (podría necesitar compilarlo usted mismo).

RIP compiled commands Using a compiled from source version of LinuxCNC open a terminal and cd to the top of the LinuxCNC folder. Set up the environment by entering *. ./scripts/rip-environment* now enter *glade*, you see a bunch of warnings in the terminal that you can ignore and the editor should open. The stock Gscreen Glade file is in: src/emc/usr_intf/gscreen/ sample skins are in /share/gscreen/skins/. This should be copied to a configuration folder. Or you can make a clean-sheet Glade file by saving it in a configuration folder.

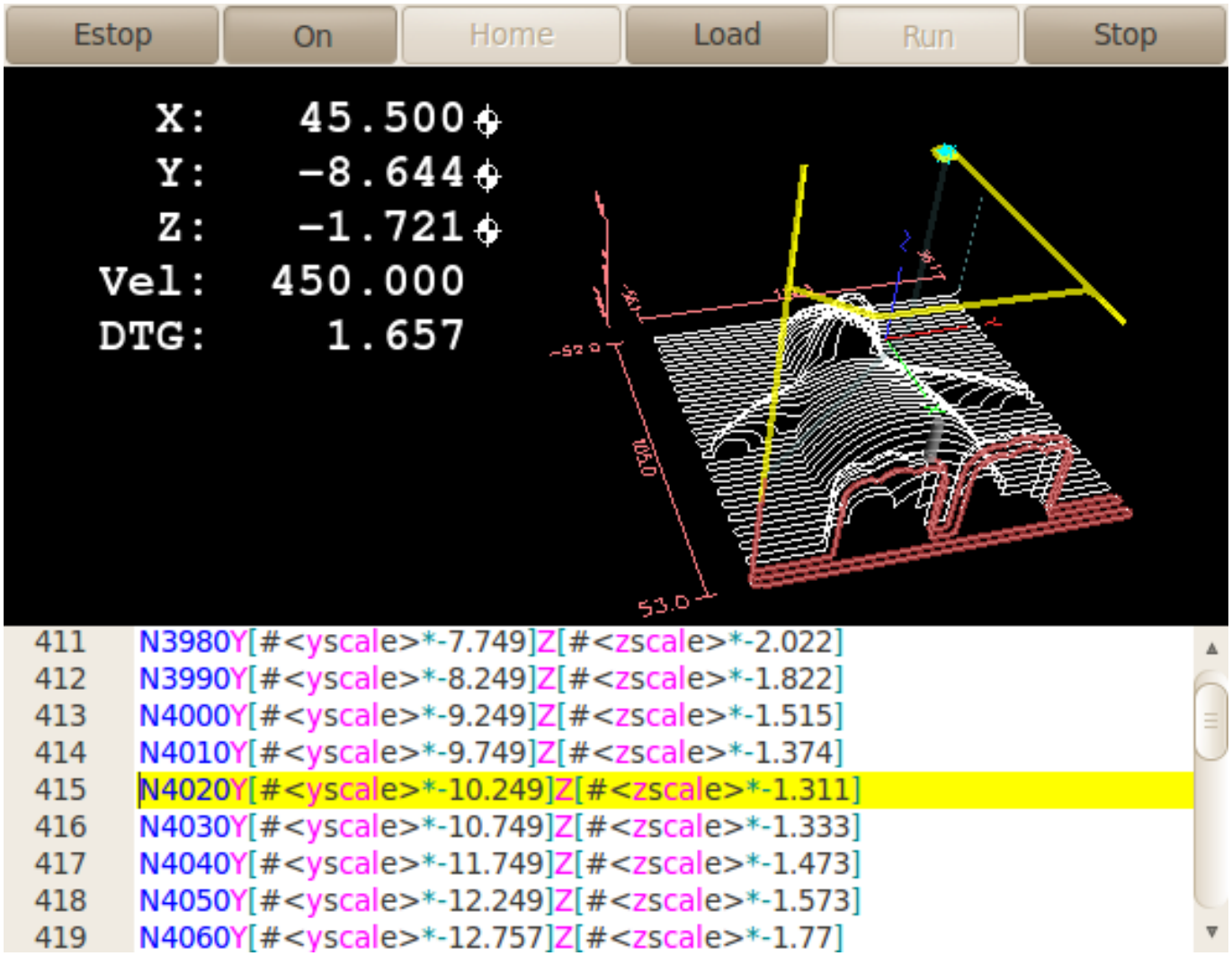
Ok you have loaded the stock Glade file and now can edit it. The first thing you notice is it does not look in the editor like what it is displayed like Gscreen uses some tricks, such as hiding all boxes of buttons except one and changing that one depending on the mode. The same goes for notebooks, some screens use notebooks with the tabs not shown. To change pages in the editor you need to temporarily show those tabs.

When making changes it is far easier to add widgets then subtract widgets and still have the screen work properly making objects *not visible* is one way to change the display without getting errors. This won't always work some widgets will be set visible again. Changing the names of Gscreen's regular widgets is probably not gonna work well without changing the Python code, but moving a widget while keeping the name is usually workable.

Gscreen leverages GladeVCP widgets as much as possible, to avoid adding Python code. Learning about GladeVCP widgets is a prerequisite. If the existing widgets give you the function you want or need then no Python code needs be added, just save the Glade file in your configuration folder. If you need something more custom then you must do some Python programming. The name of the parent window needs to be window1. Gscreen assumes this name.

Remember, if you use a custom screen option YOU are responsible for fixing it (if required) when updating LinuxCNC.

10.4.3. Building a simple clean-sheet custom screen

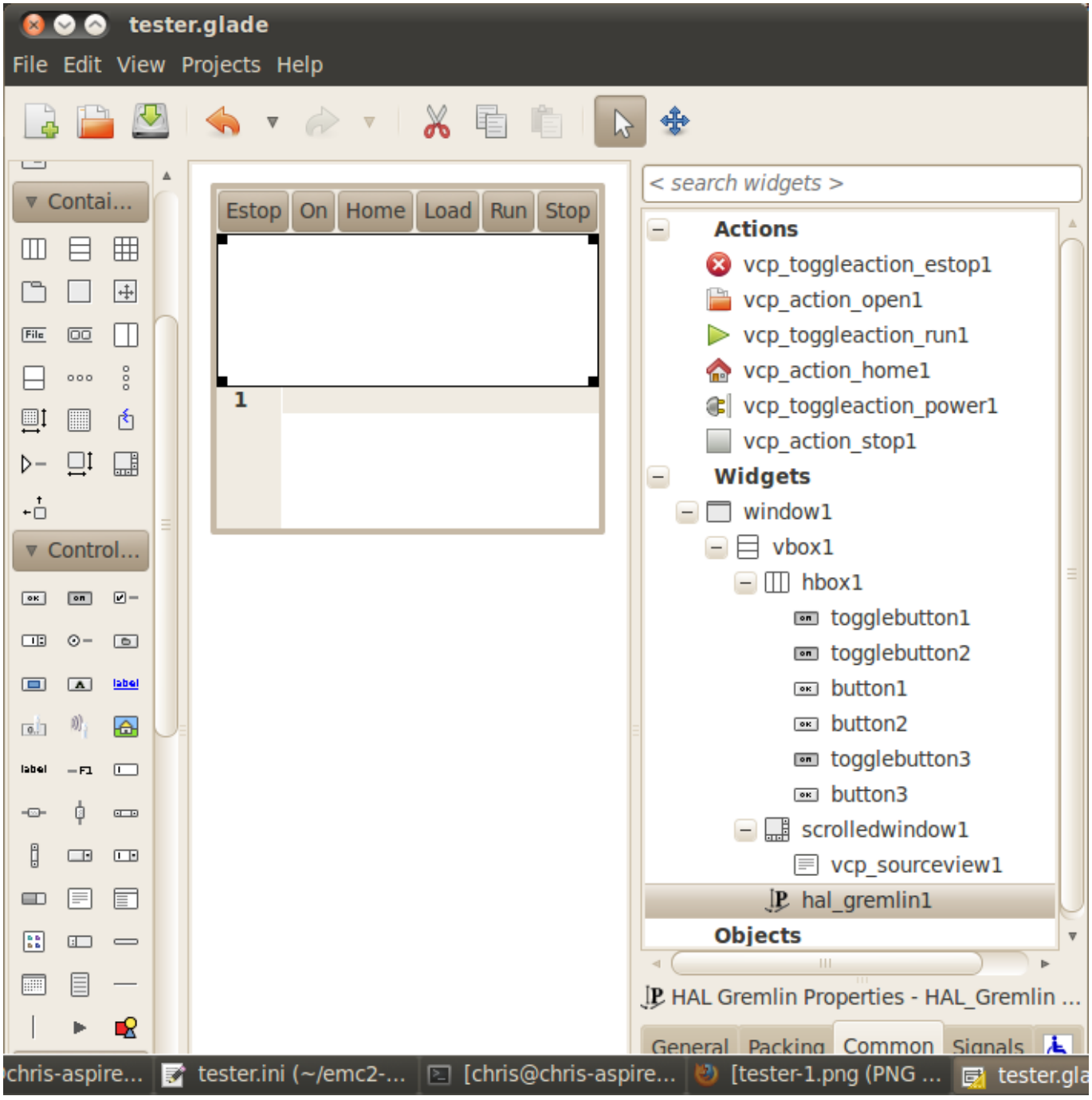


Lets build a simple usable screen. Build this in the Glade editor (if using a RIP package run it from a terminal after using . scripts/rip-environment).

Things to note:

- The top level window must be called the default name, *window1* - Gscreen relies on this.
- Add actions by right clicking, and selecting *add as toplevel widget* they don't add anything visual to the window but are added to the right most action list. Add all the ones you see on the top right.
- After adding the actions we must link the buttons to the actions for them to work (see below).
- The gremlin widget doesn't have a default size so setting a requested size is helpful (see below).
- El widget SourceView intentará usar toda la ventana, por lo que agregarlo a una ventana con scroll la cubrirá (esto ya se ha hecho en el ejemplo.)
- The buttons will expand as the window is made larger which is ugly so we will set the box they are in, to not expand (see below).
- The button types to use depend on the VCP_action used -eg vcp_toggle_action usually require toggle buttons (Follow the example for now).

- The buttons in this example are regular buttons not HAL buttons. We don’t need the HAL pins.



En esta pantalla usamos VCP_actions para comunicar a LinuxCNC las acciones que queremos. Esto nos permite funciones estándar sin agregar código python en el archivo handler. Vamos a vincular el botón Estop a la acción estop; seleccione el botón estop y bajo la pestaña general busque *Acción relacionada* y haga clic en el botón junto a él. Ahora seleccione la acción conmutada estop. Ahora el boton estop se activará y desactivará cuando se haga clic. Bajo la pestaña general puede cambiar el texto de la etiqueta del botón para describir su función. Haga esto en todos los botones.

Select the gremlin widget click the common tab and set the requested height to 100 and click the checkbox beside it.

Click the horizontal box that holds the buttons. Click the packing tab and click *expand* to *No*.

Save it as tester.glade and save it in sim/gscreen/gscreen_custom/ folder. Now launch LinuxCNC and click to sim/gscreen/gscreen_custom/tester and start it. If all goes well our screen will pop up and the buttons will do their job. This works because the tester.ini tells gscreen to look for and load tester.glade and tester_handler.py. The tester_handler.py file is included in that folder and is coded just show the screen and not much else. Since the special widgets directly communicate with LinuxCNC you can still do useful things. If your screen needs are covered by the available special widgets then this is as far as you need to go to build a screen. If you want something more there are still many tricks available from just adding *function calls* to get canned behaviour. To coding your own Python code to customize exactly what you want. But that means learning about handler files.

10.4.4. Handler file example

There are special functions Gscreen checks the handler file for. If you add these in you handler file Gscreen will call them instead of gscreen's internal same-named functions.

- initialize_preferences(self): You can install new preference routines.
- initialize_keybindings(self) You can install new keybinding routines. In most cases you won't want to do this, you will want to override the individual keybinding calls. You can also add more keybindings that will call an arbitrary function.
- initialize_pins(self): makes / initializes HAL pins
- connect_signals(self,handlers): If you are using a completely different screen the default Gscreen you must add this or gscreen will try to connect signals to widgets that are not there. Gscreen's default function is called with self.gscreen.connect_signals(handlers). If you wish to just add extra signals to your screen but still want the default ones call this first then add more signals. If you signals are simple (no user data passed) then you can also use the Glade signal selection in the Glade editor.
- initialize_widgets(self): You can use this to set up any widgets. Gscreen usually calls *self.gscreen.initialize_* which actually calls many separate functions. If you wish to incorporate some of those widgets then just call those functions directly. Or add self.gscreen.init_show_windows() so widgets are just shown. Then if desired, initialize/adjust your new widgets.
- initialize_manual_toolchange(self): allows a complete revamp of the manual toolchange system.
- set_restart_line(self.line):
- timer_interrupt(self): allows one to complete redefine the interrupt routine. This is used for calling periodic() and checking for errors from linuxcnc.status.
- check_mode(self): used to check what mode the screen is in. Returns a list[] 0 -manual 1- mdi 2- auto 3- jog.
- on_tool_change(self,widget): You can use this to override the manual tool change dialog -this is called when *gscreen.tool-change* changes state.
- dialog_return(self,dialog_widget,displaytype,pinname): Use this to override any user message or manual tool change dialog. Called when the dialog is closed.
- periodic(self): This is called every (default 100) milliseconds. Use it to update your widgets/HAL pins. You can call Gscreen regular periodic afterwards too, self.gscreen.update_position() or just add pass to not update anything. Gscreen's update_position() actually calls many separate functions. If you wish to incorporate some of those widgets then just call those functions directly.

You can also add you own functions to be called in this file. Usually you would add a signal to a widget to call your function.

10.4.4.1. Adding Keybindings Functions

Our tester example would be more useful if it responded to keyboard commands. There is a function called keybindings() that tries to set this up. While you can override it completely, we didn't - but it assumes some things:

- It assumes the estop toggle button is call *button_estop* and that F1 key controls it.
- It assumes the power button is called *button_machine_on* and the F2 key controls it.

These are easily fixed by renaming the buttons in the Glade editor to match. But instead we are going to override the standard calls and add our own.

Add these command to the handler file:

```
# override Gscreen Functions
# keybinding calls
def on_keycall_ESTOP(self,state,SHIFT,CNTRL,ALT):
  if state: # only if pressed, not released
    self.widgets.togglebutton1.emit('activate')
    self.gscreen.audio.set_sound(self.data.alert_sound)
    self.gscreen.audio.run()
    return True # stop progression of signal to other widgets
def on_keycall_POWER(self,state,SHIFT,CNTRL,ALT):
  if state:
    self.widgets.togglebutton2.emit('activate')
    return True
def on_keycall_ABORT(self,state,SHIFT,CNTRL,ALT):
  if state:
    self.widgets.button3.emit('activate')
    return True
```

So now we have overridden Gscreen's function calls of the same name and deal with them in our handler file We now reference the widgets by the name we used in the Glade editor. We also added a built in gscreen function to make a sound when Estop changes. Note that we we call Gscreen's built in functions we must use self.gscreen.[FUNCTION NAME]() If we used self.[FUNCTION NAME]() it would call the function in our handler file.

Lets add another key binding that loads halmeter when F4 is pressed.

In the handler file under *def initialize_widgets(self):* change to:

```
  def initialize_widgets(self):
    self.gscreen.init_show_windows()
    self.gscreen.keylookup.add_conversion('F4','TEST','on_keycall_HALMETER')
```

Then add these functions under the *HandlerClass* class:

```
  def on_keycall_HALMETER(self,state,SHIFT,CNTRL,ALT):
    if state:
      self.gscreen.on_halmeter()
      return True
```

This adds a keybinding conversion that directs gscreen to call on_keycall_HALMETER when F4 is pressed. Then we add the function to the handle file to call a Gscreen builtin function to start halmeter.

10.4.4.2. Linuxcnc State Status

The module *Gstat* polls LinuxCNC's state every 100ms and sends callback messages to user functions when state changes. You can register messages to act on specific state changes. As an example we will register to get *file-loaded* messages when LinuxCNC loads a new file. First we must import the module and instantiate it: In the import section of the handler file add:

```
from hal_glib import GStat
GSTAT = GStat()
```

En el archivo manejador en *def __init__(self):* agregue:

```
GSTAT.connect('file-loaded', self.update_filepath)
```

Luego, en la *HandlerClass*, agregue la función:

```
self.update_filepath(self, obj, path):
    self.widgets.my_path_label.set_text(path)
```

Cuando LinuxCNC carga un archivo nuevo, Gstat enviará un mensaje de vuelta a la función *update_filepath*. En este ejemplo actualizamos una etiqueta con ese nombre de ruta (asumiendo que hay una etiqueta nombrada *my_path_label*) en el archivo GLADE.

10.4.4.3. Jogging Keys

No hay widgets especiales para hacer jogging con botones de pantalla, así que debemos hacerlo con el código de Python. Bajo la función connect_signals agregue este código:

```
        for i in('x','y','z'):
            self.widgets[i+'neg'].connect("pressed", self['jog_'+i],0,True)
            self.widgets[i+'neg'].connect("released", self['jog_'+i],0,False)
            self.widgets[i+'pos'].connect("pressed", self['jog_'+i],1,True)
            self.widgets[i+'pos'].connect("released", self['jog_'+i],1,False)
        self.widgets.jog_speed.connect("value_changed",self.jog_speed_changed)
```

Add these functions under the HandlerClass class:

```
    def jog_x(self,widget,direction,state):
        self.gscreen.do_key_jog(_X,direction,state)
    def jog_y(self,widget,direction,state):
        self.gscreen.do_key_jog(_Y,direction,state)
    def jog_z(self,widget,direction,state):
        self.gscreen.do_key_jog(_Z,direction,state)
    def jog_speed_changed(self,widget,value):
        self.gscreen.set_jog_rate(absolute = value)
```

Finalmente, agregue dos botones al archivo GLADE para cada eje, uno para positivo, otro para trote en dirección negativa. Nombre estos botones xneg, xpos, yneg, ypos zneg, zpos respectivamente. Agregue un widget SpeedControl al archivo GLADE y llámelo jog_speed.

10.4.5. Gscreen Start Up

Gscreen es realmente solo una infraestructura para cargar un archivo GladeVCP personalizado e interactuar con ella.

1. Gscreen reads the options it was started with.
2. Gscreen sets the debug mode and set the optional skin name.
3. Gscreen checks to see if there are *local* XML, handler and/or locale files in the configuration folder. It will use them instead of the default ones (in share/gscreen/skins/) (There can be two separate screens displayed).
4. The main screen is loaded and translations set up. If present the second screen will be loaded and translations set up.

5. Optional Audio is initialized if available.
6. It reads some of the INI file to initialize the units, and the number/type of axes.
7. Initializes Python's binding to HAL to build a non-realtime component with the Gscreen name.
8. GladeVCP's makepins is called to parse the XML file to build HAL pins for the HAL widgets and register the LinuxCNC connected widgets.
9. Checks for a *local* handler file in the configuration folder or else uses the stock one from the skin folder.
10. Si hay un archivo manejador, gscreen lo analiza y registra las llamadas a la función en el espacio de nombres de Gscreen.
11. Glade matches/registers all signal calls to functions in gscreen and the handler file.
12. Gscreen checks the INI file for an option preference file name otherwise it uses *.gscreen_preferences* =.
13. Gscreen checks to see if there is a preference function call (*initialize_preferences(self)*) in the handler file otherwise it uses the stock Gscreen one.
14. Gscreen checks for ClassicLadder realtime component.
15. Gscreen checks for the system wide GTK theme.
16. Gscreen collects the jogging increments from the INI file.
17. Gscreen collects the angular jogging increments from the INI file.
18. Gscreen collects the default and max jog rate from the INI.
19. Gscreen collects the max velocity of any axes from the INI's TRAJ section.
20. Gscreen checks to see if there is angular axes then collects the default and max velocity from the INI file.
21. Gscreen collect all the override setting from the INI.
22. Gscreen checks if its a lathe configuration from the INI file.
23. Gscreen finds the name of the tool_table,tool editor and param file from the INI.
24. Gscreen checks the handler file for keybindings function (*initialize_keybindings(self)*) or else use Gscreen stock one.
25. Gscreen checks the handler file for pins function (*initialize_pins(self)*) or else use Gscreen stock one.
26. Gscreen checks the handler file for manual_toolchange function (*initialize_manual_toolchange(self)*) or else use Gscreen stock one.
27. Gscreen checks the handler file for connect_signals function (*initialize_connect_signals(self)*) or else use Gscreen stock one.
28. Gscreen checka the handler file for widgets function (*initialize_widgets(self)*) or else use Gscreen stock one.
29. Gscreen set up messages specified in the INI file.
30. Gscreen tells HAL the Gscreen HAL component is finished making pins and is ready. If there is a terminal widget in the screen it will print all the Gscreen pins to it.
31. Gscreen sets the display cycle time based on the INI file.
32. Gscreen checks the handler file for *timer_interupt(self)* function call otherwise use Gscreen's default function call.

10.4.6. Configuración INI

Under the [DISPLAY] heading:

```
DISPLAY = gscreen -c tester
  options:
   -d debugging on
   -v verbose debugging on
```

The -c switch allows one to select a *skin*. Gscreen assumes the Glade file and the handler file use this same name. The optional second screen will be the same name with a 2 (e.g., tester2.glade). There is no second handler file allowed. It will only be loaded if it is present. Gscreen will search the LinuxCNC configuration file that was launched first for the files, then in the system skin folder.

10.4.7. User Dialog Messages

This function is used to display pop up dialog messages on the screen. These are defined in the INI file and controlled by HAL pins:

MESSAGE_BOLDTEXT
: is generally a title.

MESSAGE_TEXT
: is below that and usually longer.

MESSAGE_DETAILS
: is hidden unless clicked on.

MESSAGE_PINNAME
: es el nombre base de los pines HAL.

MESSAGE_TYPE
: specifies whether its a yes/no, ok, or status message

- Status messages
  - will be shown in the status bar and the notify dialog,
  - require no user intervention.
- ok messages
  - require the user to click ok to close the dialog.
  - have one HAL pin to launch the dialog and one to signify it is waiting for response.
- yes/no messages
  - require the user to select yes or no buttons to close the dialog.
  - tiene tres pines HAL:
    1. one to show the dialog,
    2. uno para espera, y
    3. one for the answer.

Aquí hay un código INI de muestra. Estaría bajo el encabezado [DISPLAY].

```
# This just shows in the status bar and desktop notify popup.
MESSAGE_BOLDTEXT = NONE
MESSAGE_TEXT = This is a statusbar test
MESSAGE_DETAILS = STATUS DETAILS
MESSAGE_TYPE = status
MESSAGE_PINNAME = statustest
```

```
# This will pop up a dialog that asks a yes no question
MESSAGE_BOLDTEXT = NONE
MESSAGE_TEXT = This is a yes no dialog test
MESSAGE_DETAILS = Y/N DETAILS
MESSAGE_TYPE = yesnodialog
MESSAGE_PINNAME = yndialogtest

# This pops up a dialog that requires an ok response and it shows in the status bar and
# the desktop notify popup.
MESSAGE_BOLDTEXT = This is the short text
MESSAGE_TEXT = This is the longer text of the both type test. It can be longer then the  ↩
    status bar text
MESSAGE_DETAILS = BOTH DETAILS
MESSAGE_TYPE = okdialog status
MESSAGE_PINNAME = bothtest
```

10.4.7.1. Copy the Stock Handler/Glade File For Modification

Si desea utilizar una pantalla de serie pero modificar su archivo de controlador, debe copie el archivo existente a su carpeta de archivo de configuración. Gscreen verá esto y usará el archivo copiado. Pero ¿dónde está el archivo original? Si utiliza un LinuxCNC RIP, las skins de muestra están en /share/gscreen/skins/ *SCREENNAME* Las versiones instaladas de LinuxCNC las tienen en lugares ligeramente diferentes dependiendo de la distribución utilizada. Una manera fácil de encontrar la ubicación es abrir una terminal e inicie la pantalla sim que desea usar. En el terminal se imprimirán las ubicaciones de los archivos. Puede ser útil agregar la opción -d a la línea de carga de gscreen en el INI.

Here is a sample:

```
chris@chris-ThinkPad-T500 ~/emc-dev/src $ linuxcnc
LINUXCNC - 2.7.14
Machine configuration directory is '/home/chris/emc-dev/configs/sim/gscreen/gscreen_custom'
Machine configuration file is 'industrial_lathe.ini'
Starting LinuxCNC...
Found file(lib): /home/chris/emc-dev/lib/hallib/core_sim.hal
Note: Using POSIX non-realtime
Found file(lib): /home/chris/emc-dev/lib/hallib/sim_spindle_encoder.hal
Found file(lib): /home/chris/emc-dev/lib/hallib/axis_manualtoolchange.hal
Found file(lib): /home/chris/emc-dev/lib/hallib/simulated_home.hal
**** GSCREEN WARNING: no audio alerts available - Is python-gst0.10 library installed?
**** GSCREEN INFO ini: /home/chris/emc-dev/configs/sim/gscreen/gscreen_custom/ ↩
    industrial_lathe.ini
**** GSCREEN INFO: Skin name = industrial

**** GSCREEN INFO:  Using SKIN glade file from /home/chris/emc-dev/share/gscreen/skins/ ↩
    industrial/industrial.glade ****

**** GSCREEN INFO:  No Screen 2 glade file present
**** GSCREEN INFO: handler file path: ['/home/chris/emc-dev/share/gscreen/skins/industrial/ ↩
    industrial_handler.py']
```

The line:

```
**** GSCREEN INFO: handler file path: ['/home/chris/emc-dev/share/gscreen/skins/industrial/ ↩
    industrial_handler.py']
```

shows where the stock file lives. Copy this file to your config folder. This works the same for the Glade file.

10.5. QtDragon GUI

10.5.1. Introducción

QtDragon and QtDragon_hd are built with the QtVCP framework. It is the creative vision of forum personality Persei8. Much of it is based on the excellent work of others in the LinuxCNC community. LinuxCNC's version is adapted from Persei8's Github versions. It is primarily meant for 3-5 axes machines such as mills or routers. It works well with a touchscreen and/or mouse. QtDragon supports multiple ways to touch off tools and probing work pieces. You can use LinuxCNC's external offsets capability to automatically raise the spindle during a pause. If you the VersaProbe option and remap code you can add auto tool length probing at tool changes.

nota
QtDragon and QtVCP are relatively new programs added into LinuxCNC. Bugs and oddities are possible. Please test carefully when using a dangerous machine. Please forward reports to the forum or maillist.

10.5.1.1. QtDragon

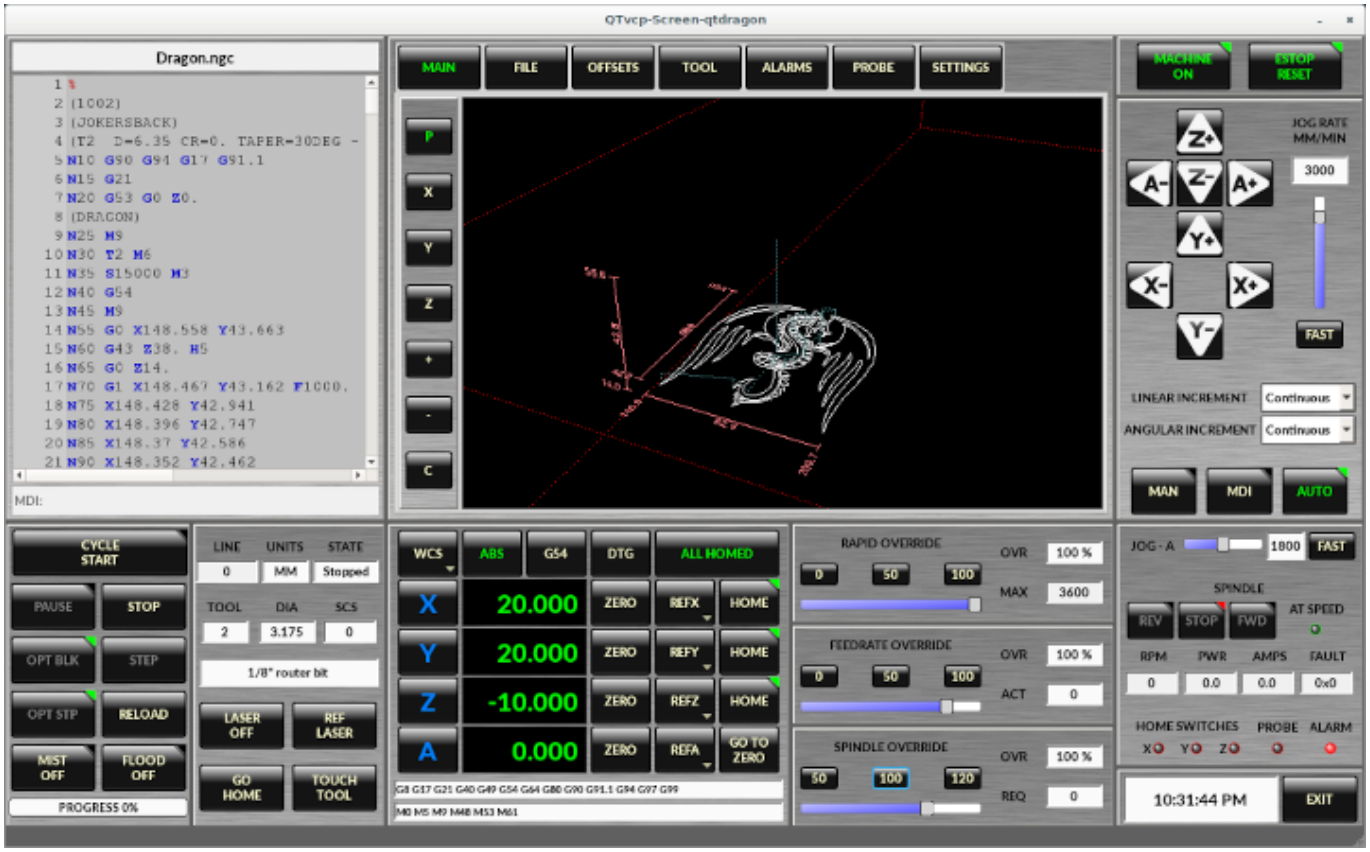
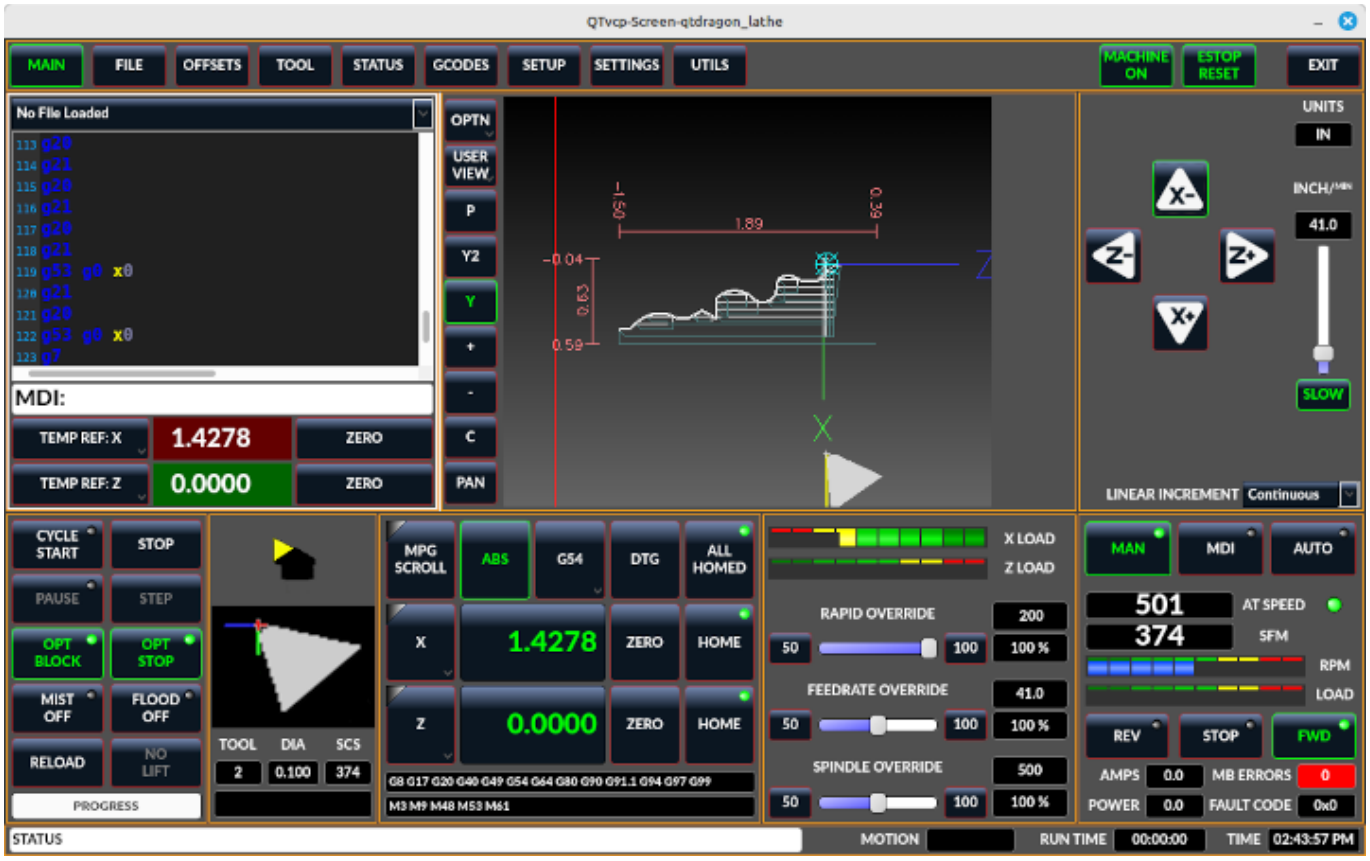


Figura 10.27: QtDragon - 3 to 5 axis sample (1440x860) in silver theme

QtDragon is resizable from a resolution of 1280x768 to 1680x1200. It will work in window mode on any monitor with higher resolution but not on monitors with lower resolution.

10.5.1.2. QtDragon_lathe



QtDragon_lathe is a modified version of QtDragon more suitable for lathes.
It is resizable from a resolution of 1280x768 to 1680x1200.
It will work in window mode on any monitor with higher resolution but not on monitors with lower resolution.

10.5.1.3. QtDragon_hd

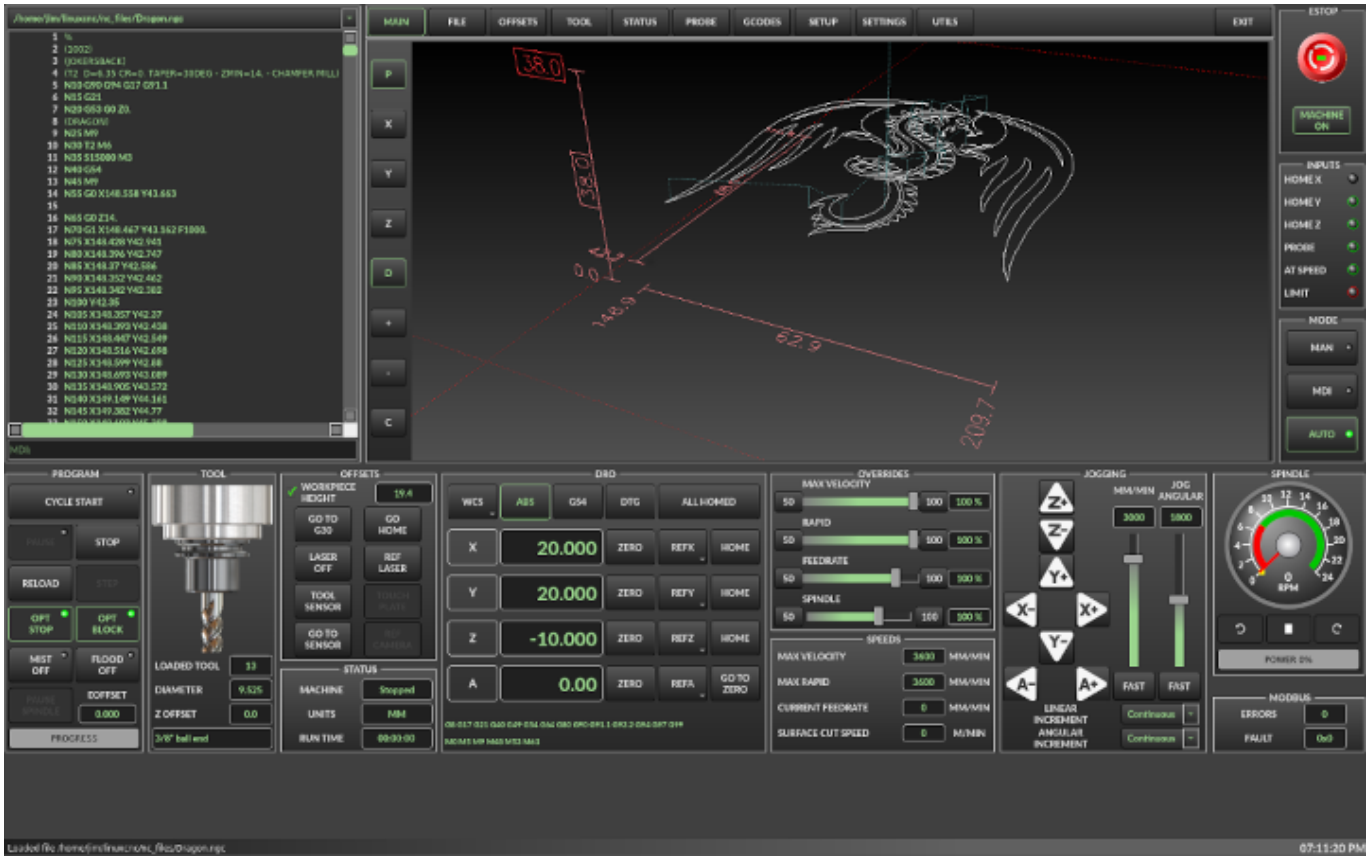


Figura 10.28: QtDragon_hd - 3 to 5 axis sample for larger monitors (1920x1056) in dark theme

QtDragon_hd is a similar design as QtDragon but modified to utilize the extra space of modern larger monitors. There are some small differences in layout and utility.

QtDragon_hd has a resolution of 1920x1056 and is not resizeable. It will work in window mode on any monitor with higher resolution but not on monitors with lower resolution.

10.5.1.4. QtDragon_hd_vertical

QtDragon_hd_vertical is a vertical orientated version. It is not resizable.

10.5.2. Getting Started - The INI File

If your configuration is not currently set up to use QtDragon, you can change it by editing the INI file sections. For an exhaustive list of options, see the display section of the INI file documentation.

---

nota
You can only have one of each section (e.g., [HAL]) in the INI file. If you see in these docs multiple section options, place them all under the one appropriate section name.

10.5.2.1. Display

In the section `[DISPLAY]` change the `DISPLAY =` assignment to read:

- `qtdragon` for a small version
- `qtdradon_hd` para la versión grande.

You can add `-v`, `-d`, `-i`, or `-q` for (respectably) verbose, debug, info or quiet output to the terminal.

```
[DISPLAY]
DISPLAY = qtvcp qtdragon
```

10.5.2.2. Preferences

To keep track of preferences, QtDragon looks for a preference text file. Add the following entry under the `[DISPLAY]` heading.
It can use ~ for home directory or `WORKINGFOLDER` or `CONFIGFOLDER` to represent QtVCP's idea of those directories:
This example will save the file in the config folder of the launch screen. (Other options are possible see the QtVCP's screenoption widget docs.)

```
[DISPLAY]
PREFERENCE_FILE_PATH = WORKINGFOLDER/qtdragon.pref
```

10.5.2.3. Logging

You can specify where to save history/logs.
These file names can be user selected.
In the section `[DISPLAY]` add:

```
[DISPLAY]
MDI_HISTORY_FILE = mdi_history.dat
MACHINE_LOG_PATH = machine_log.dat
LOG_FILE = qtdragon.log
```

10.5.2.4. Anular controles

These set qtdragon's override controls (1.0 = 100 percent):

```
[DISPLAY]
MAX_SPINDLE_0_OVERRIDE = 1.5
MIN_SPINDLE_0_OVERRIDE = .5
MAX_FEED_OVERRIDE       = 1.2
```

10.5.2.5. Spindle controls

Spindle control settings (in rpm and watts):

```
[DISPLAY]
DEFAULT_SPINDLE_0_SPEED = 500
SPINDLE_INCREMENT = 200
MIN_SPINDLE_0_SPEED = 100
MAX_SPINDLE_0_SPEED = 2500
MAX_SPINDLE_POWER = 1500
```

10.5.2.6. Jogging increments

Set selectable jogging increments.
These increments can be user changed.

```
[DISPLAY]
INCREMENTS = Continuous, .001 mm, .01 mm, .1 mm, 1 mm, 1.0 inch, 0.1 inch, 0.01 inch
ANGULAR_INCREMENTS = 1, 5, 10, 30, 45, 90, 180, 360
```

10.5.2.7. Grid Increments

Set the available optional grid sizes for graphics display.
This will override the default sizes.
mm and *in* are used to specify units.
The grid selection box is shown when clicking the *OPTN* button on the graphics display.

```
[DISPLAY]
GRIDS = 0, .1 mm, 1 mm, 2 mm, 5 mm, 10 mm, .25 in, .5 in
```

10.5.2.8. Jog speed

Establece controles de velocidad de trote (en unidades por segundo)

```
[DISPLAY]
MIN_LINEAR_VELOCITY     = 0
MAX_LINEAR_VELOCITY     = 60.00
DEFAULT_LINEAR_VELOCITY = 50.0
DEFAULT_ANGULAR_VELOCITY = 10
MIN_ANGULAR_VELOCITY = 1
MAX_ANGULAR_VELOCITY = 360
```

10.5.2.9. User message dialog system

Optional popup custom message dialogs, controlled by HAL pins.
MESSAGE_TYPE can be *okdialog* or *yesnodialog*. See `qtvcp/library/messages` for more information.
This example shows how to make a dialog that requires the user to select *ok* to acknowledge and hide.
These dialogs could be used for such things as low lube oil warnings, etc.

```
[DISPLAY]
MESSAGE_BOLDTEXT = Este es un texto corto
MESSAGE_TEXT = Este es un texto mas largo de ambos tipos de prueba. Puede ser mas largo que ↩
     el texto de la barra de estado
MESSAGE_DETAILS = BOTH DETAILS
MESSAGE_TYPE = okdialog
MESSAGE_PINNAME = oktest
```

Multimessages use an s32 HAL pin to pop multiple defined messages.

```
[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
```

10.5.2.10. Embed Custom VCP Panels

You can optionally embed QtVCP Virtual Control Panels into the QtDragon screens.
These panels can be either user built or builtin QtVCP Panels.
See QtVCP/VCP panels for other available builtin panels.

The `EMBED_TAB_NAME` entry will used as the title for the new tab.(must be unique)
Tab `EMBED_TAB_LOCATION` options include: `tabWidget_utilities`, `tabWidget_setup` and `stackedWidget_ma`
and `WINDOW`.
Tab `EMBED_TAB_COMMAND` specifies what embed-able program to run, including any of its command line options.

If using the `tabWidget_utilities` or `tabWidget_setup` locations, an extra tab will appear with the panel.
If using `stackedWidget_mainTab`, a button labelled *User* will appear.
Pressing this button will cycle through displaying all available panels (specified for this location) on the main tab area.
You also use `WINDOW` to create a pop up window that cab be shown/hidden with an arrow button that will appear near the machine on button

Sample adding a builtin panel to the utilities tab, i.e., a graphical animated machine using the vismach library.

```
[DISPLAY]
EMBED_TAB_NAME = Vismach demo
EMBED_TAB_COMMAND = qtvcp vismach_mill_xyz
EMBED_TAB_LOCATION = tabWidget_utilities
```

This example panel is designed to display additional RS485 VFD data and also to configure a 4 sheave, 2 belt spindle drive via a series of buttons.




```
[DISPLAY]
EMBED_TAB_NAME = Spindle Belts
EMBED_TAB_COMMAND = qtvcp spindle_belts
EMBED_TAB_LOCATION = tabWidget_utilities
```

10.5.2.11. Subroutine Paths

If using NGCGUI, remap or custom M codes routines, LinuxCNC needs to know where to look for the files.
This sample is typical of what is needed for NgcGui, Basic Probe. and Versa Probe remap code.
These paths will need to be adjusted to point to the actual files on your system. RS274NZGC Section Details

```
[RS274NGC]
SUBROUTINE_PATH = :~/linuxcnc/nc_files/examples/ngcgui_lib:~/linuxcnc/nc_files/examples/ ↩
    ngcgui_lib/utilitysubs: \
~/linuxcnc/nc_files/examples/probe/basic_probe/macros:~/linuxcnc/nc_files/examples/remap- ↩
    subroutines: \
~/linuxcnc/nc_files/examples/ngcgui_lib/remap_lib
```

QtVCP's NGCGUI program also need to know where to open for subroutine selection and pre-selection. NGCGUI_SUBFILE_PATH must point to an actual path on your system and also a path described in SUBROUTINE_PATHS.

```
[DISPLAY]
# NGCGUI subroutine path.
# Thr path must also  be in [RS274NGC] SUBROUTINE_PATH
NGCGUI_SUBFILE_PATH = ~/linuxcnc/nc_files/examples/ngcgui_lib
# pre selected programs tabs
# specify filenames only, files must be in the NGCGUI_SUBFILE_PATH
NGCGUI_SUBFILE = slot.ngc
NGCGUI_SUBFILE = qpocket.ngc
```

10.5.2.12. Control de vista previa

Magic comments can be used to control the G-code preview.

On very large programs the preview can take a long time to load. You can control what is shown and what is hidden the the graphics screen by adding the appropriate comments from this list into your G-code:

```
(PREVIEW,stop)
(PREVIEW,hide)
(PREVIEW,show)
```

10.5.2.13. Program Extensions/Filters

You can control what programs are displayed in the filemanager window with program extensions. Create a line with the *.* endings you wish to use separated by commas, then a space and the description. You can add multiple lines for different selections in the combo box.

```
[FILTER]
PROGRAM_EXTENSION = .ngc,.nc,.tap G-Code file (*.ngc,*.nc,*.tap)
```

QtDragon has the ability to send loaded files through a *filter program*. This filter can do any desired task: Something as simple as making sure the file ends with *M2*, or something as complicated as generating G-code from an image. See Filter Programs for more information.

La sección *[FILTER]* del archivo INI controla cómo funcionan los filtros. Primero, para cada tipo de archivo, escriba una línea *PROGRAM_EXTENSION*. Luego, especifique el programa a ejecutar para cada tipo de archivo. Este programa recibe el nombre del archivo de entrada como su primer argumento, y debe escribir el código rs274ngc en la salida estándar. Esta salida es lo que se mostrará en el área de texto, se previsualizará en el área de visualización y será ejecutado por LinuxCNC con *Run*.

Las siguientes líneas agregan soporte para el convertidor `image-to-gcode` incluido en LinuxCNC y para correr programas de filtro basados en Python:

```
[FILTER]
PROGRAM_EXTENSION = .png,.gif,.jpg # Imagen de profundidad de escala de grises
PROGRAM_EXTENSION = .py # Script python
png = image-to-gcode
gif = image-to-gcode
jpg = image-to-gcode
py = python
```

10.5.2.14. Probe/Touchplate/Laser Settings

QtDragon has INI entries for two optional probing tab screens available. Comment/uncomment which ever you prefer.

- *Versa probe* is a QtVCP ported version of a popular GladeVCP probing panel.
- *Basic Probe* is a QtVCP ported version based on the third party basic probe screen.

Both perform similar probing routines, though Versa probe optionally handles auto tool measurement.

```
[PROBE]
#USE_PROBE = versaprobe
USE_PROBE = basicprobe
```

10.5.2.15. Abort detection

When using qtdragon's probing routines, it is important to detect a user abort request.
By default, LinuxCNC does not report an abort in a useful way for the probe routines.
You need to add a ngc file to print out an error that can be detected. Remap Abort Details

```
[RS274NGC]
# on abort, this ngc file is called. required for basic/versa probe routines. +
ON_ABORT_COMMAND=O <on_abort> call
```

This example code will send a message on abort. The probe routines can detect this sample.
According to the setting above, it would need to be saved as *on_abort.ngc* within LinuxCNC's [RS274NGC] SUBROUTINE_PATHS and [DISPLAY] PROGRAM_PREFIX search paths.

```
o<on_abort> sub

o100 if [#1 eq 5]
    (machine on)
o100 elseif [#1 eq 6]
    (machine off)
o100 elseif [#1 eq 7]
    (estopped)
o100 elseif [#1 eq 8]
    (msg,Process Aborted)
o100 else
    (DEBUG,Abort Parameter is %d[#1])
o100 endif

o<on_abort> endsub
m2
```

10.5.2.16. Startup codes

You should set default M/G code for start up. These will be overridden by running a NGC file.
These are only sample codes, integrator should choose appropriate codes.

```
[RS274NGC]
# start up G/M codes when first loaded
RS274NGC_STARTUP_CODE = G17 G20 G40 G43H0 G54 G64P0.0005 G80 G90 G94 G97 M5 M9
```

10.5.2.17. Macro Buttons

QtDragon has up to ten convenience buttons for calling *macro actions*.
These are under the heading *[MDI_COMMAND_LIST]* as *MDI_COMMAND_MACRO0* = to *MDI_COMMAND_*
=
These could also call OWord routines if desired.
In the sample configurations they are labelled for moving between current user system origin (zero point) and Machine system origin.
User origin is the first MDI command in the INI list, machine origin is the second.
This example shows how to move Z axis up first. Commands separated by the *;* are run one after another.
The button label text can be set with any text after a comma, the `\n` symbol adds a line break.

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

10.5.2.18. Post GUI HAL File

These optional HAL files will be called after QtDragon has loaded everything else.
You can add multiple line for multiple file. Each one will be called in the order they appear.
Calling HAL files after QtDragon is already loaded assures that QtDragon's HAL pins are available.

Sample with typical entries for the specificion of HAL files to be read after the QtDragon was started. Adjust these lines to match actual requirements.

```
[HAL]
POSTGUI_HALFILE = qtdragon_hd_postgui.hal
POSTGUI_HALFILE = qtdragon_hd_debugging.hal
```

10.5.2.19. Post GUI HAL Command

These optional HAL commands will be run after QtDragon has loaded everything else.
You can add multiple line. Each one will be called in the order they appear.
Any HAL command can be used.

Sample with typical files in INI file to load modules after the GUI is available. Adjust these to match your actual requirements.

```
[HAL]
POSTGUI_HALCMD = loadusr qtvcp test_probe
POSTGUI_HALCMD = loadusr qtvcp test_led
POSTGUI_HALCMD = loadusr halmeter
```

10.5.2.20. HAL Bridge

Hal Bridge is similar to HALUI - it has HAL pins that communicate with QtDragon.
These pins could be used with HALUI to built a more friendly control panel.

- It can report/change the current selected Axis button.
- The jog rate/increments will be reported.
- There is a cycle start and pause pin - these call the code in QtDragon rather then the motion controller.
- If there are macros defined in the INI, there will be pins available to initiate them.

Sample entry. Remove -d to quiet debugging messages.

```
[HAL]
HALBRIDGE= hal_bridge -d
```

Typical HAL pins avaialble:

```
Component Pins:
Owner   Type  Dir                Value  Name
    38  s32   OUT                    0  base-thread.time
    29  bit   OUT                FALSE  bridge.axis-x-is-selected
    29  bit   IN                 FALSE  bridge.axis-x-select
```

```
    29  bit   OUT                FALSE  bridge.axis-y-is-selected
    29  bit   IN                 FALSE  bridge.axis-y-select
    29  bit   OUT                FALSE  bridge.axis-z-is-selected
    29  bit   IN                 FALSE  bridge.axis-z-select
    29  bit   IN                 FALSE  bridge.cycle-pause-in
    29  bit   IN                 FALSE  bridge.cycle-start-in
    29  float OUT                    0  bridge.jog-increment
    29  float OUT                    0  bridge.jog-increment-angular
    29  float OUT                   15  bridge.jog-rate
    29  float OUT                  360  bridge.jog-rate-angular
    29  s32   OUT                   -1  bridge.joint-selected
    29  bit   IN                 FALSE  bridge.macro-cmd-MACRO0
    29  bit   IN                 FALSE  bridge.macro-cmd-MACRO1
```

10.5.2.21. Builtin Sample Configurations

The sample configurations `sim/qtdragon/` or `sim/qtdragon_hd` are already configured to use QtDragon as the screen. There are several examples that demonstrate various machine configurations.

10.5.3. Key Bindings

QtDragon is not intended to primarily use a keyboard for machine control.
It lacks many keyboard short cuts that for instance AXIS has - but you can use a mouse or touchscreen.
There are several key presses that will control the machine for convenience.

```
F1 - Estop on/off
F2 - Machine on/off
F12 - Style Editor
Home - Home All Joint of the Machine
Escape - Abort Movement
Pause - Pause Machine Movement
```

10.5.4. Buttons

Buttons that are checkable will change their text colour when checked. This is controlled by the stylesheet/theme

10.5.5. Virtual Keyboard

QtDragon includes a virtual keyboard for use with touchscreens. To enable the keyboard, check the Use Virtual Keyboard checkbox in the Settings page. Clicking on any input field, such as probe parameters or tool table entries, will show the keyboard. To hide the keyboard, do one of the following:

- press the *HIDE* button on the virtual keyboard.
- Haga clic en el botón de página PRINCIPAL
- go into AUTO mode

It should be noted that keyboard jogging is disabled when using the virtual keyboard.

10.5.6. HAL Pins

These pins are specific to the QtDragon screen.
There are of course are many more HAL pins that must be connected for LinuxCNC to function.

If you need a manual tool change prompt, add these lines in your postgui file.
QtDragon emulates the hal_manualtoolchange HAL pins - don't load the separate HAL component *hal_manualtoolchange*.

```
net tool-change      hal_manualtoolchange.change   <=  iocontrol.0.tool-change
net tool-changed     hal_manualtoolchange.changed  <=  iocontrol.0.tool-changed
net tool-prep-number hal_manualtoolchange.number   <=  iocontrol.0.tool-prep-number
```

Also if you don't have an automatic tool changer make sure these pins are connected in one of the HAL files:

```
net tool-prepare-loopback iocontrol.0.tool-prepare => iocontrol.0.tool-prepared
```

This input pin should be connected to indicate probe state.

```
qtdragon.led-probe
```

These pins are inputs related to spindle VFD indicating.
The volt and amp pins are used to calculate spindle power. You must also set the MAX_SPINDLE_POWER in the INI.

```
qtdragon.spindle-modbus-connection
qtdragon.spindle-modbus-errors
qtdragon.spindle-amps
qtdragon.spindle-fault
qtdragon.spindle-volts
```

This bit pin is an output to the spindle control to pause it.
You would connect it to `spindle.0.inhibit`.

```
qtdragon.spindle-inhibit
```

QtDragon spindle speed display and spindle-at-speed LED require that `spindle.0.speed-in` be connected to spindle speed feedback.
Encoder or VFD feedback could be used, as long as the feedback is in revolutions per second (RPS).
If no feedback is available you can have the display show the requested speed by connecting pins like so:

```
net spindle-speed-feedback spindle.0.speed-out-rps => spindle.0.speed-in
```

This bit output pin can be connected to turn on a laser:

```
qtdragon.btn-laser-on
```

This float output pin indicates the camera rotation in degrees:

```
qtdragon.cam-rotation
```

These bit/s32/float pins are related to external offsets if they are used:

```
qtdragon.eoffset-clear
qtdragon.eoffset-enable
qtdragon.eoffset-value
qtdragon.eoffset-spindle-count
qtdragon.eoffset-zlevel-count
qtdragon.eoffset-is-active
```

These float output pins reflect the current slider jograte (in machine units):

```
qtdragon.slider-jogspeed-linear
qtdragon.slider-jogspeed-angular
```

These float output pins reflect the current slider override rates:

```
qtdragon.slider-override-feed
qtdragon.slider-override-maxv
qtdragon.slider-override-rapid
qtdragon.slider-override-spindle
```

These output pins are available when setting the Versa Probe INI option. They can be used for auto-tool-length-probe at tool change - with added remap code.

```
qtversaprobe.enable
qtversaprobe.blockheight
qtversaprobe.probeheight
qtversaprobe.probevel
qtversaprobe.searchvel
qtversaprobe.backoffdist
```

This pin will be true when the loaded tool equals the number set in the Versa Probe tool number in the preference file.
It can be used (for example) to inhibit the spindle when the probe is loaded by connecting it to `spindle.0.inhibit`.

```
qtversaprobe.probe-loaded
```

This output pin is available when setting the Basic Probe INI option.
This pin will be true when the loaded tool equals the number set in the Basic Probe tool number edit box.
It can be used (for example) to inhibit the spindle when the probe is loaded by connecting it to `spindle.0.inhibit`.

```
qtbasicprobe.probe-loaded
```

This input pin is available to toggle pause/resume of a running program.

```
qtdragon.external-pause
```

You can externally operate dialog responses on most qtdragon dialogs.

```
qtdragon.dialog-ok
qtdragon.dialog-no
qtdragon.dialog-cancel
```

10.5.7. HAL files

The HAL files supplied are for simulation only. A real machine needs its own custom HAL files. The QtDragon screen works with 3 or 4 axes with one joint per axis or 3 or 4 axes in a gantry configuration (2 joints on 1 axis).

10.5.8. Manual Tool Changes

If your machine requires manual tool changes, QtDragon can pop a message box to direct you. QtDragon emulates the hal_manualtoolchange HAL pins - don't load the separate HAL component *hal_manualtoolc*
Hereto you must connect the proper HAL pin in the postgui HAL file, for example:

```
net tool-change      hal_manualtoolchange.change   <=  iocontrol.0.tool-change
net tool-changed     hal_manualtoolchange.changed  <=  iocontrol.0.tool-changed
net tool-prep-number hal_manualtoolchange.number   <=  iocontrol.0.tool-prep-number
```

10.5.9. Husillo

The screen is intended to interface to a VFD, but will still work without it.
There are a number of VFD drivers included in the LinuxCNC distribution.
It is up to the end user to supply the appropriate driver and HAL file connections according to his own machine setup.

10.5.10. Auto Raise Z Axis on Program Pause

QtDragon can be set up to automatically raise and lower the Z axis and stop the spindle, when the program is paused.
You toggle the *SPINDLE LIFT* or *NO LIFT* button to select the option to raise the spindle in Z when paused.
Then when you press the *PAUSE* button the spindle will lift the amount set on the *Settings* tab and the spindle will stop.
Pressing *RESUME* will start the spindle and lower the spindle.
If you have the HAL pin `spindle.0.at-speed` connected to a driving pin, the spindle will not lower until the pin is true
You typically connect this to a timer or logic that detects the speed of the spindle.
If that pin is not connected to a driving pin, a dialog will pop up to warn you to wait for the spindle speed.
The spindle will lower when you close that dialog.
The amount to raise is set in the *Settings* tab under the heading *SPINDLE RAISE*.
This line edit box can only be directly set when not in Auto mode.
The up/down buttons can be used to adjust the raise amount at any time, including when the spindle is already raised.
The button increments are 1 inch or 5 mm (depending on the units the machine is based on).

---

nota
If using the Spindle lift option, HALUI can not be used to pause/resume the program. There is a pin, *QtDragon.external-pause* available to pause/resume from an external source. You must also enable external offsets. In the setting tab check *use external offsets* If you wish to inhibit the spindle when a probe tool is loaded, you will need to use an logical `or`-component to combine the two spindle inhibit signals to connect to `spindle.0.inhibit`.

This optional behaviour requires additions to the INI and the QtDragon_postgui HAL file.

In the INI, under the AXIS_Z heading.

```
[AXIS_Z]
OFFSET_AV_RATIO  = 0.2
```

This reserves 20 % of max velocity and max acceleration for the external offsets.
This will limit max velocity of the machine by 20 %

In the qtdragon_postgui.hal file add:

```
# Set up Z axis external offsets
net eoffset_clear    qtdragon.eoffset-clear          =>  axis.z.eoffset-clear
net eoffset_count    qtdragon.eoffset-spindle-count  =>  axis.z.eoffset-counts
net eoffset          qtdragon.eoffset-value          <=  axis.z.eoffset
net eoffset-state    qtdragon.eoffset-is-active      <=  motion.eoffset-active

# Inhibit spindle when paused
net spindle-pause    qtdragon.spindle-inhibit        =>  spindle.0.inhibit

# uncomment for dragon_hd
#net limited          qtdragon.led-limits-tripped     <= motion.eoffset-limited

setp axis.z.eoffset-enable 1
setp axis.z.eoffset-scale 1.0
```

10.5.11. Z level compensation

QtDragon_hd can be set up to probe and compensate for Z level height changes by utilizing the external program *G-code Ripper*.

nota
This is only available in the QtDragon_hd version.

Z level compensation is a bed levelling/distortion correction function typically used in 3D printing or engraving. It uses a HAL non-realtime component which utilizes the external offsets feature of LinuxCNC. The component has a HAL pin that specifies an interpolation type, which must be one of cubic, linear or nearest (0, 1, 2 respectively). If none is specified or if an invalid number is specified, the default is assumed to be cubic.

When Z LEVEL COMP is enabled, the compensation component reads a probe data file, which must be called *probe_points.txt*. The file can be modified or updated at any time while compensation is disabled. When next enabled, the file will be reread and the compensation map is recalculated. This file is expected to be in the configuration directory.

The probe data file is generated by a probing program, which itself is generated by an external python program called `gcode_ripper`, which can be launched from the file manager tab using the *G-code Ripper* button.

10.5.11.1. Using G-code Ripper for Z level Compensation

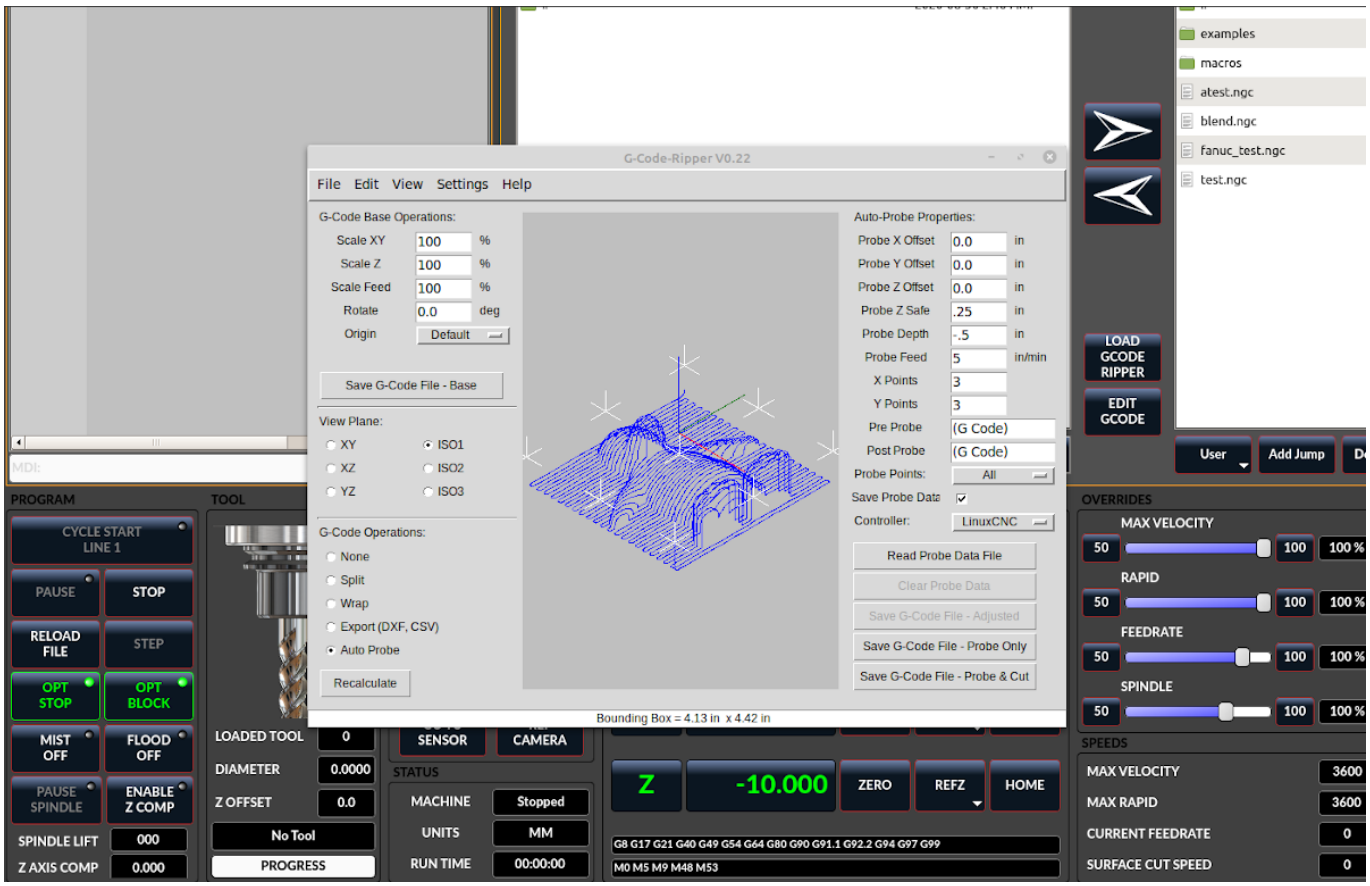


Figura 10.29: QtDragon_hd showing G-code Ripper

nota
G-code Ripper offers many functions that we will not go in to here. This is only available in the QtDragon_hd version.

- In qtdragon_hd, switch to the file tab and press the load G-code Ripper button.
- Set origin to match the origin of the G-code file to be probed.
- Under G-Code Operations, check Auto Probe.
- File -> Open G-Code File (The file you will run after compensation)
- If necessary, make adjustments and press Recalculate.
- Press Save G-Code File - Probe Only.
- Save the generated file to the nc_files folder.
- Exit gcode_ripper.
-
- Without changing the offsets, run this program. Make sure the probe tool is installed. When complete, there will be a file in the config directory called *probe_points.txt*.

- In qtdragon_hd, press the *Enable Z Comp* button to enable compensation. Look at the status line for indication of success or failure. Active compensation will be displayed beside the label: *Z Level Comp* While jogging that display should change based on the compensation component.

---

nota
If you use auto raise Z to lift the spindle on pause, you must combine the two with a HAL component and feed that to LinuxCNC's motion component.

Sample postgui HAL file for combined spindle raise and Z Level compensation

```
# load components
#############################################################################

loadrt logic names=logic-and personality=0x102
addf logic-and servo-thread

# load a summing component for adding spindle lift and Z compensation
loadrt scaled_s32_sums
addf scaled-s32-sums.0 servo-thread

loadusr -Wn z_level_compensation z_level_compensation
# method parameter must be one of nearest(2), linear(1), cubic (0)
setp z_level_compensation.fade-height 0.0
setp z_level_compensation.method 1

# conecta señales al componente de movimiento de Linux CNC
#############################################################################

net eoffset-clear    axis.z.eoffset-clear
net eoffset-counts   axis.z.eoffset-counts
setp axis.z.eoffset-scale .001
net eoffset-total          axis.z.eoffset
setp axis.z.eoffset-enable True

# desplazamientos externos para función de pausa del husillo
#############################################################################
net eoffset-clear             qtdragon.eoffset-clear
net eoffset-spindle-count   <= qtdragon.eoffset-spindle-count
net spindle-pause             qtdragon.spindle-inhibit       =>  spindle.0.inhibit
net eoffset-state             qtdragon.eoffset-is-active     <=  motion.eoffset-active

## Z level compensation
#####################################################
net eoffset-clr2            z_level_compensation.clear      => logic-and.in-01
net xpos-cmd                z_level_compensation.x-pos      <= axis.x.pos-cmd
net ypos-cmd                z_level_compensation.y-pos      <= axis.y.pos-cmd
net zpos-cmd                z_level_compensation.z-pos      <= axis.z.pos-cmd
net z_compensation_on       z_level_compensation.enable-in  <= qtdragon.comp-on
net eoffset-zlevel-count    z_level_compensation.counts     => qtdragon.eoffset-zlevel- ↩
    count

# add Z level and scaled spindle raise level values together
net eoffset-spindle-count   scaled-s32-sums.0.in0
net eoffset-zlevel-count    scaled-s32-sums.0.in1
setp scaled-s32-sums.0.scale0 1000
net eoffset-counts          scaled-s32-sums.0.out-s
```

10.5.12. Probing

The probe screen has been through basic testing but there could still be some minor bugs. When running probing routines, use extreme caution until you are familiar with how everything works. Probe routines run without blocking the main GUI. This gives the operator the opportunity to watch the DROs and stop the routine at any time.

nota
Probing is very unforgiving to mistakes; be sure to check settings before using.

QtDragon has 2 methods for setting Z0. The first is a touchplate, where a metal plate of known thickness is placed on top of the workpiece, then the tool is lowered until it touches the plate, triggering the probe signal. The current user system's (G5x) Z0 is set to probe height - the entered plate thickness.

The second method uses a tool setter in a fixed position and a known height above the table where the probe signal will be triggered. In order to set Z0 to the top of the workpiece, it has to know

1. how far above the table the probe trigger point is (tool setter height) and
2. how far above the table the top of the workpiece is.

This operation has to be done every time the tool is changed as the tool length is not saved.

For touching off with a touch probe, whether you use the touchplate operation with thickness set to 0 or use a probing routine, the height from table to top of workpiece parameter is not taken into account and can be ignored. It is only for the tool setter.

10.5.12.1. Versa Probe

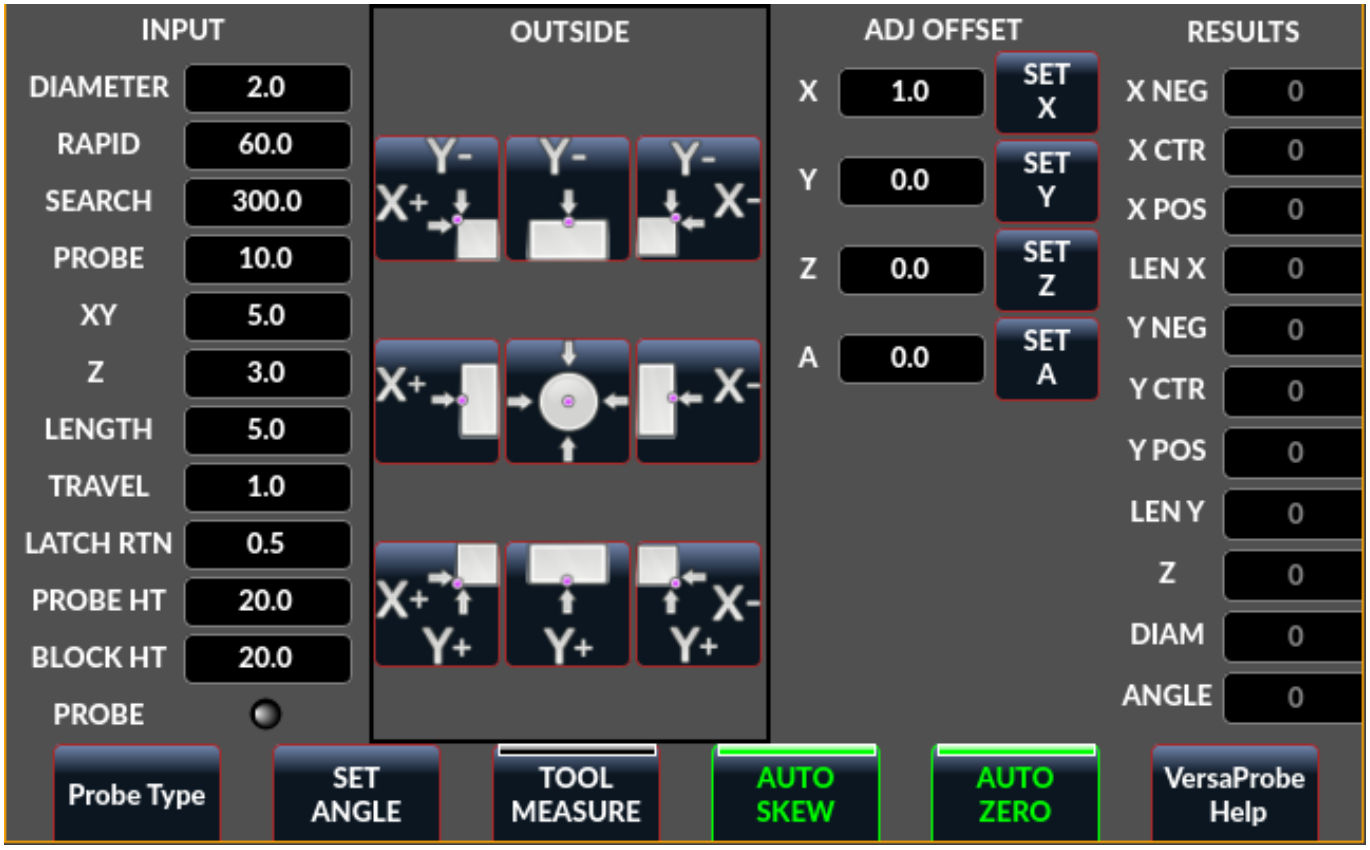


Figura 10.30: QtDragon - Versa Probe Option

Versa probe is used to semi-automatically probe work pieces to find edges, centers and angles.
It can also be sued to auto probe tool length at tool changes with added remap code.

You must carefully set the *Probing Parameters*:

DIAMETER
: This is the diameter of the probe tip. The accuracy of probe measurements is directly affected by the accuracy of the probe tip diameter.

TRAVEL
: The distance that the probe will travel during the initial search. If the search distance is too short, you will receive a message like ”G38 finished without making contact”. For safety reasons, it is recommended to set this parameter to 3-4 mm more than probe stylus diameter.

LATCH RTN
: The distance the probe is retracted after making initial contact with the workpiece. This should be a short distance because the second approach will be at a slow speed, but large enough for the probe to break contact and bring it to the search ready state. If the Latch Rtn distance too large, you will end up spending a lot of time waiting for the search to complete. Recommendation: 1-2 mm

SEARCH
: This is the feed rate at which the probe searches for the target workpiece in machine units per minute. The search speed should be slow enough to give an acceptable initial accuracy, but fast enough to not waste time waiting for movement. Recommendation: 200-500 mm/min.

PROBE
: Once initial contact has been made and the probe is retracted, it will wait for 0.5 seconds before performing the search again at a lower speed, the probe velocity. This lower speed ensures the machine can stop movement as quickly as possible on contact with the workpiece.

RAPID
: Axis movements not associated with searching are done at the speed defined by RAPID in machine units per minute.

SIDE/EDGE LENGTH
: This is the distance the probe will move at the rapid rate to the position where it will begin a search. If measuring a corner, it will move EDGE LENGTH units away from the corner, then move away from the workpiece by XY CLEARANCE, lower by Z CLEARANCE and begin the initial search. If measuring an inner circle, then EDGE LENGTH should be set to the approximate radius of the circle. Note: NOT the diameter.

PROBE HT
: The height of the tool sensor from the machine table surface. This value is used to calculate the Z zero height for the current work coordinate system when using the probe with a tool setter sensor.

BLOCK HT
: The height of the top of the workpiece from the machine table surface. This value is used to calculate the Z zero height for the current work coordinate system when using the probe with a tool setter sensor.

XY CLEARANCE
: The distance that the probe will move away from an edge or corner before performing a search. It should be large enough to ensure that the probe will not contact the workpiece or any other fixtures before moving down. It should be small enough to avoid excessive waiting for movement while searching.

Z CLEARANCE
: The distance that the probe will move down before performing a search. If measuring an inside hole, the probe could be manually jogged to the starting Z height and then set Z CLEARANCE to 0.

There are three toggle buttons:

Auto Zero
: This selects if after probing the relevant axis is set to zero in the current user system.

Auto Skew
: This selects if after probing, the system will be rotated or just display the calculated rotation.

Tool Measure
: This (if integrated) turns auto tool probing on and off.

Versaprobe ofrece 5 pines para fines de medición de herramientas y uno que puede usarse para inhibir el husillo cuando la sonda se ha cargado.
Los 5 pines se utilizan para leer desde una subrutina de código G de reasignación, para que el código pueda reaccionar a diferentes valores.
Actualmente el número de herramientas sonda solo es editable en el archivo de preferencias:

```
[VERSA_PROBE_OPTIONS]
ps_probe_tool = 1
```

qtversaprobe.enable (HAL_BIT)
Permitir o no medición de herramientas. Refleja el estado del botón en la pantalla.

qtversaprobe.blockheight (HAL_FLOAT)
El valor medido de la cara superior de la pieza de trabajo Refleja entrada en la pantalla.

qtversaprobe.probeheight (HAL_FLOAT)
La altura del interruptor de la sonda asignadora de herramienta. Refleja la entrada en la pantalla.

qtversaprobe.searchvel (HAL_FLOAT)
The velocity to search for the tool probe switch

qtversaprobe.probevel (HAL_FLOAT)
La velocidad para sondear la longitud de la herramienta. Refleja la entrada en la pantalla.

qtversaprobe.backoffdist (HAL_FLOAT)
La distancia que regresa la sonda después del disparo. Refleja la entrada en la pantalla.

qtversaprobe.probe-loaded (HAL_BIT)
Reflect if the current tool is equal to the preference file probe number.

10.5.12.2. Basic probe

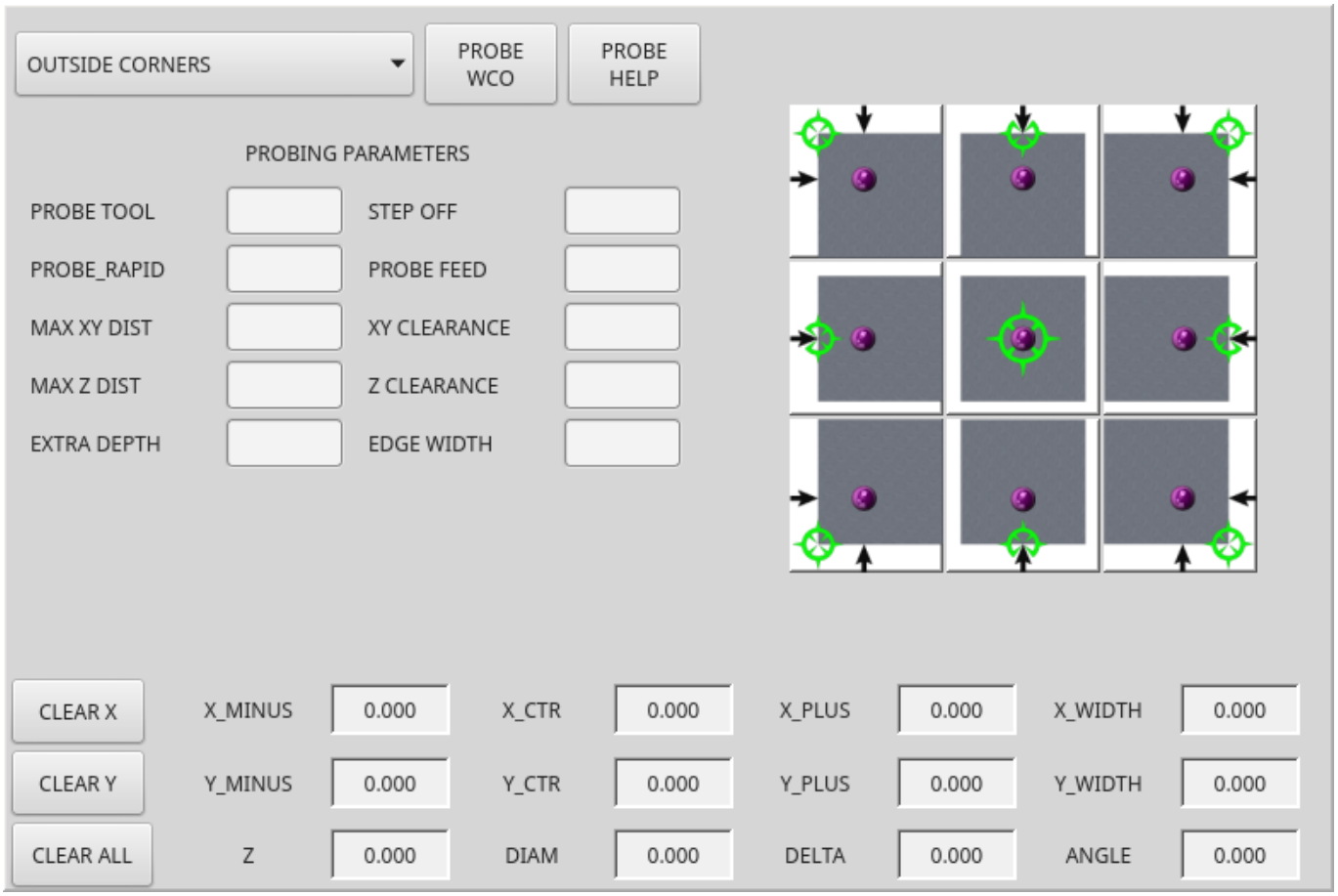


Figura 10.31: QtDragon - Basic Probe Option

Basic probe is used to semi-automatically probe work pieces to find edges, centers and angles. The combo box allows selecting the basic type of probing buttons shown:

- Outside Corners
- Inside Corners
- Edge Angles
- Boss and Pockets
- Ridge and Valleys
- Calibration

You must carefully set the *Probing Parameters*:

Probe Tool
: Will only allow probing if this tool number is in the spindle

Probe Diameter
: El tamaño de la punta de la sonda

Probe Rapid
: La velocidad de movimientos rápidos en unidades de máquina

Probe Search
: La velocidad de la primera búsqueda (*burda*) en unidades de máquina

Probe Feed
: La velocidad de la segunda búsqueda (*fina*) en unidades de máquina

Step Off
: Back off and re-probe distance

Max XY Distance
: The maximum distance the probe will search for in X and Y before failing with error

Max Z Distance
: The maximum distance the probe will search for in Z before failing with error

XY Clearance
: Clearance distance from probe to wall edge before rapid traversing down in Z and *rough* probing

Z Clearance
: Clearance distance from probed to top of material

Extra Depth
: Distance from top of material to desired probe depth

There are also hint parameters depending on selected probing type:

Edge Width
: Desired distance from the probe start position, along wall edge before starting to probe

Diameter Hint
: Used by Round Boss or Round Pocket probing (start move: 1/2 diameter plus XY clearance)

X Hint
: Used by Rectangular Boss/Pocket probing (start move: 1/2 X length plus XY clearance)

Y Hint
: Used by Rectangular Boss/Pocket probing (start move: 1/2 Y length plus XY clearance)

Después de configurar los parámetros y pistas:

- Manually move the probe to the approximate position represented by the green target on the button.
- Confirm the parameters are reasonable.
- Press the desired probing button.

The probing routine will start immediately.

nota
Pressing the stop button or the keyboard escape key, will abort the probing.

This can be used to inhibit the spindle when the probe is loaded.
You would connect it to spindle.0.inhibit

```
qtbasicprobe.probe-loaded
```

Lets discuss inside corner probing using the top right button in Basic Probe (back_right_inside). While all probe entries must be correct, the most important settings to change for each each probe:

XY CLEARANCE
Distance away from edge before rough probing,

Z CLEARANCE
Distance from probe to top of material,

EXTRA DEPTH
Distance to lower probe from top of material,

EDGE WIDTH
Distance along edge wall (away from corner) to start probing.

nota
These distance are always to be set in *machine units* (mm for metric machine, inch for imperial machine).

Preestablecido:

- manual set probe at the intersection of the edges (ie corner) of material as described by the green bullseye on the button. Set it Z CLEARANCE above the top of material. These can be done by eye.
- set EXTRA CLEARANCE to a value that you want the probe to go below the *top* of material. (So the probe will move from its start position down Z Clearance + Extra Clearance distance.)
- set XY CLEARANCE to a value that definitely gives clearance from the wall so when the probe goes down it does not hit anything.
- set EDGE WIDTH to a value that describes the distance measured from the corner, along the wall to where you wish to probe. this edge distance will be used along the X wall and then the Y wall.

Sequence after pressing the probe button:

1. Rapid EDGE WIDTH distance away from corner at the same time moving XY CLEARANCE away from edge. So this is a slightly diagonal move.
2. Move probe down by Z CLEARANCE + EXTRA DEPTH,

3. probe wall twice (rough and fine),
4. move diagonally to the other wall as set by EDGE WIDTH and XY CLEARANCE,
5. probe wall twice,
6. raise probe up by Z CLEARANCE + EXTRA DEPTH (returns to starting height),
7. rapid back to starting corner (now calculated using the probed walls),
8. if auto zero button is enabled, set X and Y of the current user system to zero.

10.5.12.3. Customizing Probe Screen Widget

It is possible to load a customized version of the probe widget.

There should be a folder in the config folder; for screens: named <CONFIG FOLDER>/qtvcp/.
There may be (or can be added) a folder *lib/* and *widgets/*
In the widgets folder you can copy *basic_probe.py* (or *versa_probe.py*) and *probe_subprog.py*
In the lib folder copy *touchoff_subprogram.py*
If these files are found they will be used instead of the originals.
You can modify the files to change behaviour.

10.5.13. Touch plate



Figura 10.32: QtDragon - Touch Plate

You can use a conductive touch plate or equivalent to auto touch off (zero the user coordinate) for the Z position of a tool. There must be a tool loaded prior to probing. In the tool tab or settings tab, set the touch plate height, search and probe velocity and max. probing distance.

nota
When using a conductive plate the search and probe velocity should be the same and slow. If using a tool setter that has spring loaded travel then you can set search velocity faster. LinuxCNC ramps speed down at the maximum acceleration rate, so there can be travel after the probe trip if the speed is set to high.

Place the plate on top of the surface you wish to zero Z on. Connect the probe input wire to the tool (if using a conductive plate). There is a LED to confirm the probe connection is reliable prior to probing. Move the tool manually within the max probe distance. Press the *Touch Plate* button. The machine will probe down twice and the current user offset (G5X) will be zeroed at the bottom of the plate by calculation from the touchplate height setting.

10.5.14. Auto Tool Measurement

10.5.14.1. Descripción general

QtDragon can be setup to do integrated auto tool measurement using the Versa Probe widget and remap code.

This feature assumes the use of two probes in concert:

1. A tool switch sensor, fixed somewhere on the machine (sometimes called a tool-setter), and
2. a spindle probe that is installed temporarily at the beginning of the job (sometimes called an xyz probe or a Renishaw probe).

These techniques are useful for machines that do not have repeatable tool holders and do not have automatic tool changing devices. (For machines with repeatable tool holders, see the section on measuring tool length. For machines with automatic tool changing devices, consult work done under the LinuxCNC repository at `configs/sim/axis/remap/rack-toolchange`.)

To use this feature, you will need to do some additional settings and you may want to use the offered HAL pins to get values in your own ngc remap procedure. Those settings are covered later in the section.

First, this document covers how to use this technique. Second, this document covers how to set up for this technique at the beginning of a production run.

10.5.14.2. Workflow Overview

A detailed workflow walkthrough follows this overview.

Setup steps include:

- Capturando las velocidades de sondeo en la página de configuraciones de sonda versa.
- Enabling ”Use Tool Measurement” on the Versa Probe tab.
- Enabling ”Use Tool Sensor” under Settings.

importante
When fist setting up auto tool measurement, please use caution until you confirm tool change and probe locations - it is easy to break a tool/probe. Abort will be honoured while the probe is in motion.

Tool Measurement in QtDragon is organized into the following steps which will be explained in more detail in the following section:

1. Zero the probe tool by measuring the tool setter with the spindle probe installed
2. Toque de su pieza en X e Y.
3. Mida la altura de su bloque desde la base donde está ubicado su interruptor de herramienta a la cara superior del bloque (incluida mordaza, etc.).
4. In the Versa probe tab, enter the measured value for block height.
5. Ir al modo automático e iniciar su programa.

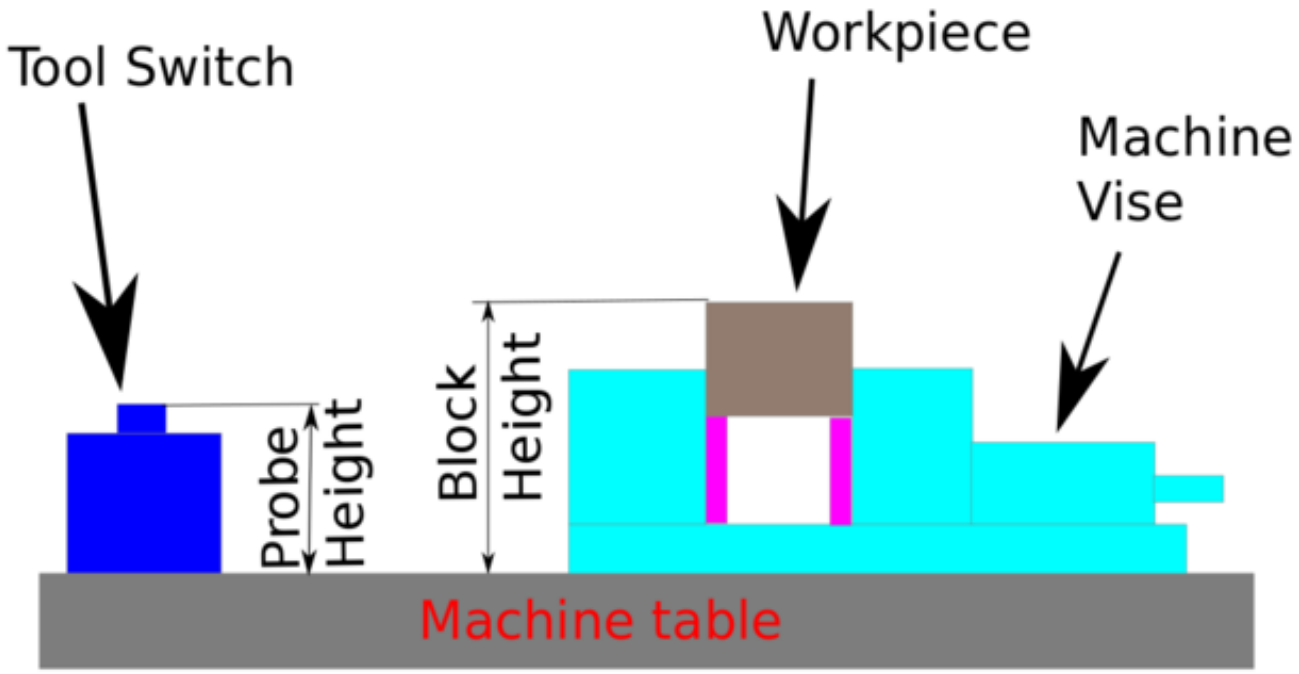


Figura 10.33: Auto tool measurement

With the first given tool change the tool will be measured and the offset will be set automatically to fit the block height. The advantage of this way is, that you do not need a reference tool.

nota
Your program must contain a tool change at the beginning. The tool will be measured, even it has been used before, so there is no danger if the block height has changed. There are several videos on you tube that demonstrate the technique using GMOCCAPY. The GMOCCAPY screen pioneered the technique.

The sequence of events (using the default files in the default setting):

1. Rapid move in Z to position defined in the INI's [TOOL_CHANGE] Z.
2. Rapid move in X and Y to number defined in INI's [TOOL_CHANGE] X and Y.
3. Perform normal M6 tool change, i.e., stop spindle, send message to user to change the tool.
4. Rapid move in X and Y to position defined in the INI's [VERSA_TOOLSETTER] X and Y.
5. Rapid move down in Z to position defined in the INI's [VERSA_TOOLSETTER] Z.
6. Probe down in Z to maximum defined in the INI's [VERSA_TOOLSETTER] MAXPROBE.
7. Changes the offset of the current work coordinate system to match the difference between the previous tool and the currently measured tool.
8. Rapid move up in Z to position defined in the INI's [VERSA_TOOLSETTER] Z_MAX_CLEAR.
9. Rapid move to the X Y position when the tool change was called.
10. Rapid move down to the Z position when the tool change was called.

---

nota
The [TOOL_CHANGE] Z position should be high enough so the tool will not hit the tool probe when moving to the [VERSA_TOOLSETTER] X and Y position.

MAXPROBE distance needs to be high enough for the tool to touch the probe.

10.5.14.3. Detailed Workflow Example

The following setups need only be done once as long as they remain in effect:

1. Under Probe Tool Screens: Ensure reasonable values for "Rapid" and "Search," these are the speeds at which the probing will be performed and are in machine units per minute.
2. Under Probe Tool Screens: Ensure that "Tool Measure" is enabled (this is a button that must be highlighted)
3. Under Settings: Ensure that "Use Tool Sensor" is enabled (this is a tick-box that must be checked)
4. In the Tool Table: Set up a tool for the spindle probe and ensure that its Z offset is set to zero.

---

nota
It is possible to use a non-zero tool length for the tool probe, but this requires extra steps and is easy to make mistakes. The following procedure assumes a zero tool probe length.

The following setup is done before beginning a program that has M6 tool change commands inside it.

1. Physically load the spindle probe into the spindle.
2. Logically load the spindle probe into the spindle with the M61 Qx command where x is the number in the tool table for the spindle probe (there is a button inside the tool table tab that can also be used)

3. Position to the Toolsetter: Use the button under the Probe Screens for ”Go To Toolsetter” to position the spindle above the Toolsetter.
4. Toolsetter Measure: Use the button under the Probe Screens for ”Probe Tool Setter Z Height.” Note that this will set and display on the Probe Settings screen the ”Probe HT” value in ABS coordinates
5. Trote hacia su pieza de trabajo.
6. Workpiece Measure: Use the button under the Probe Screens for ”Probe Z Height of Material:” this will set and display on the Probe Settings screen the ”Block Ht” value in ABS coordinates. (Typically, this will now also be the zero Z for your Work Coordinate System)
7. Set Work Coordinate System (ie, G54, or other): Use the Probe Tool and whatever probe screen or other method is appropriate to set the X, Y, and Z coordinate system needed for your job.
8. If your program begins with a TnM6 command before spinning the spindle, you may leave the spindle probe installed. You may also issue a TnM6 command to change out the spindle probe, and if the program issues the same one, it will skip the tool change.



atención
Take care not to leave the spindle probe in the spindle if a program may start the spindle.

Once those steps are complete, a program with multiple TnM6 toolchanges can be started and will operate with automatic pauses for manual tool change followed by automated tool measurement.

nota
After probing the new tool length using the tool-setter, this remap code uses a G43 which applies the offset to the Work Coordinate system which was in effect when the M6 command was issued. Because remapping has adjusted the Work Coordinate system by the offset between the previous and the current tool, the tool tip will end up at the same point in space as the tip of the previous tool was when the tool change was called.

10.5.14.4. Work Piece Height Probing in QtDragon_hd

The [TOOL_CHANGE] Z position should be high enough so the tool will not hit the tool probe when moving to the [VERSA_TOOLSETTER] X and Y position. MAXPROBE distance needs to be high enough for the tool to touch the probe.

10.5.14.5. Work Piece Height Probing

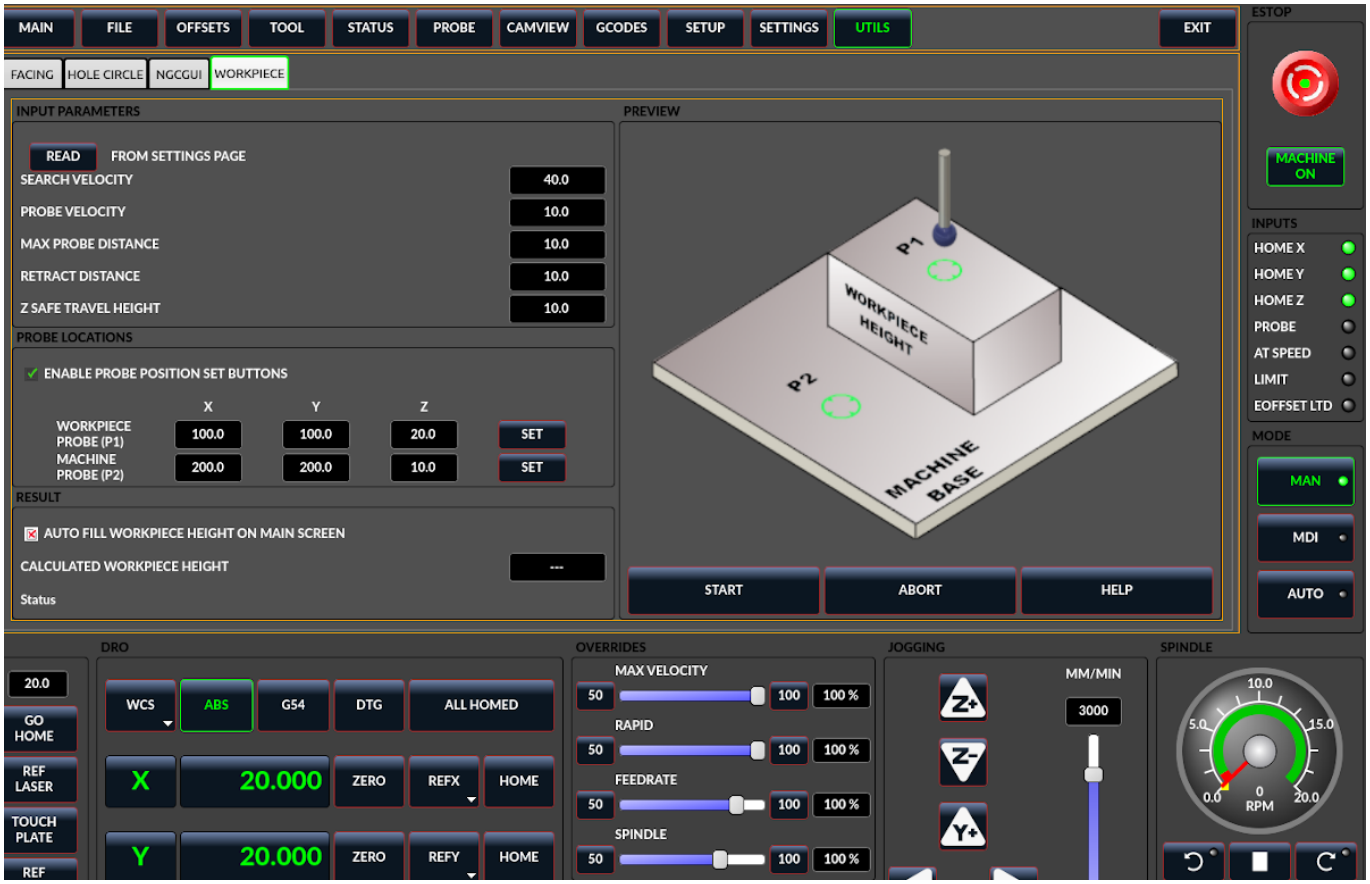


Figura 10.34: QtDragon_hd - Work piece Height probing

This program probes 2 user specified locations in the Z axis and calculates the difference in heights.

---

nota
This is only available in the QtDragon_hd version.

Enable Probe Position Set Buttons

- When checked, the SET buttons are enabled.
- This allows the user to automatically fill in the X, Y and Z parameters with the current position as displayed on the DROs.

Autofill Workpiece Height on Main Screen

- When checked, the calculated height is automatically transferred to the Workpiece Height field in the main screen.
- Otherwise, the main screen is not affected.

Workpiece Probe At

- the X, Y and Z coordinates specify where the first probing routine should start, in current WCS

Machine Probe At

- the X, Y and Z coordinates specify where the second probing routine should start, in current WCS

Z Safe Travel Height

- The machine is raised to the Z safe travel height before jogging to the X and Y coordinates.
- El husillo entonces baja a la coordenada Z especificada.
- It should be selected so that the tool clears all obstructions while jogging.

START button

- The machine will jog to the first location and then probe down.
- The machine then jogs to the second location and probes down again.
- The difference in probed values is reported as Calculated Workpiece Height.
- The parameters for search velocity, probe velocity, maximum probe distance and return distance are read from the main GUI Settings page.

ABORT button

- causes all jog and probe routines currently executing to stop.

HELP button

- displays this help file.
- Any 2 points within the machine operating volume can be specified.
- If the first point is higher than the second, the calculated height will be a positive number.
- If the first point is lower than the second, the calculated height will be a negative number.
- Units are irrelevant in this program. The probed values are not saved and only the difference is reported.



atención
Setting incorrect values can lead to crashes into fixtures on the machine work surface. Initial testing with no tool and safe heights is recommended.

10.5.14.6. Tool Measurement Pins

Versaprobe offers 5 output pins for tool measurement purpose. The pins are used to be read from a remap G-code subroutine, so the code can react to different values.

`qtversaprobe.enable` (HAL_BIT)
Permitir o no medición de herramientas. Refleja el estado del botón en la pantalla.

qtversaprobe.blockheight (HAL_FLOAT)
El valor medido de la cara superior de la pieza de trabajo Refleja entrada en la pantalla.

qtversaprobe.probeheight (HAL_FLOAT)
La altura del interruptor de la sonda asignadora de herramienta. Refleja la entrada en la pantalla.

qtversaprobe.searchvel (HAL_FLOAT)
The velocity to search for the tool probe switch

qtversaprobe.probevel (HAL_FLOAT)
La velocidad para sondear la longitud de la herramienta. Refleja la entrada en la pantalla.

qtversaprobe.backoffdist (HAL_FLOAT)
La distancia que regresa la sonda después del disparo. Refleja la entrada en la pantalla.

10.5.14.7. Tool Measurement INI File Modifications

Modify your INI file to include the following:

QtDragon allows you to select one of two styles of touch probe routines. Versa probe works with a M6 remap to add auto tool probing.

```
[PROBE]
#USE_PROBE = versaprobe
USE_PROBE = basicprobe
```

RS274NGC Section Details
Remap Statement Details
Remap Abort Details

---

nota
These default entries should work fine in most situations. Some systems may need to use *linuxcnc/nc_files/examples/* instead of *linuxcnc/nc_files/*. please check that paths are valid. Custom entries pointing to modified file are possible.

```
[RS274NGC]

# Adjust this paths to point to folders with stdglue.py, qt_auto_tool_probe.ngc and  ↩
    on_abort.ngc
# or similarly coded custom remap files.
SUBROUTINE_PATH = ~/linuxcnc/nc_files/remap-subroutines:\
~/linuxcnc/nc_files/remap_lib

# is the sub, with is called when a error during tool change happens.
ON_ABORT_COMMAND=O <on_abort> call

# The remap code for QtVCP's versaprobe's automatic tool probe of Z
REMAP=M6  modalgroup=6 prolog=change_prolog ngc=qt_auto_probe_tool epilog=change_epilog
```

La posición del sensor de la herramienta y la posición inicial del movimiento de sondeo.
Todo los valores son coordenadas absolutas (G53), excepto MAXPROBE, que se expresa como una longitud absoluta de movimiento.
Todos los valores están en unidades nativas de máquina.
X, Y, y Z establece la ubicación de la sonda asignadora de herramienta.

Auto probe action sequence in the default qt_auto_probe_tool example remap defined above (this behavior can be changed by modifying either the remap statement in the RS274NGC section, or by modifying the qt_auto_probe_tool.ngc code.):

1. rapid move to the INI’s [CHANGE_POSITION] Z position (this is a relative move, it adds this Z value to the current Z coordinate)
2. movimiento rápido a la posición X y Y [CHANGE_POSITION] del INI.
3. wait for manual tool change acknowledgement
4. rapid move to the INI’s [VERSA_TOOLSETTER] X & Y position
5. rapid move to the INI’s [VERSA_TOOLSETTER] Z_MAX_CLEAR Z position
6. fast probe
7. slow probe
8. rapid move to the INI’s [VERSA_TOOLSETTER] Z_MAX_CLEAR Z position

Z_MAX_CLEAR is the Z position to go to before moving to the tool setter when using the *Travel to Toolsetter button*.
Travel to Toolsetter Action sequence:

1. rapid move to [VERSA_TOOLSETTER] Z_MAX_CLEAR Z position
2. rapid move to [VERSA_TOOLSETTER] XY position
3. rapid move to [VERSA_TOOLSETTER] Z position.

Configuraciones de ejemplo:

```
[VERSA_TOOLSETTER]
X = 10
Y = 10
Z = -20
Z_MAX_CLEAR = -2
MAXPROBE =  -20
```

This is not named TOOL_CHANGE_POSITION on purpose - **canon uses that name and will interfere otherwise**. The position to which to move the machine before giving the change tool command. All values are in absolute coordinates. All values are in machine native units.

```
[CHANGE_POSITION]
X = 10
Y = 10
Z = -2
```

The Python section sets up what files LinuxCNC’s Python interpreter looks for, e.g., `toplevel.py` file in the `python` folder in the configuration directory: These default entries should work fine in most situations. Some systems may need to use *linuxcnc/nc_files/examples/* instead of *linuxcnc/nc_files/*. Custom entries pointing to modified file are possible.

```
# The path start point for all remap searches, i.e. Python’s sys.path.append()
PATH_APPEND = ~/linuxcnc/nc_files/remap_lib/python-stdglue/python
# path to the tremap’s ’toplevel file
TOPLEVEL = ~/linuxcnc/nc_files/remap_lib/python-stdglue/python/toplevel.py
```

10.5.14.8. Required HAL Connections

Make sure to connect the tool probe input in your HAL file: If connected properly, you should be able to toggle the probe LED in QtDragon if you press the probe stylus.

```
net probe  motion.probe-input <= <your_input_pin>
```

10.5.15. Run from Line

A G-code program can be started at any line by clicking on the desired line in the G-code display while in AUTO mode. It is the operator's responsibility to ensure the machine is in the desired operational mode. A dialog will be shown allowing the spindle direction and speed to be preset. The start line is indicated in the box labelled LINE, next to the CYCLE START button. The run from line feature can be disabled in the settings page.

nota
LinuxCNC's run-from-line is not very user friendly. E.g., it does not start the spindle or confirm the proper tool. Also, it does not handle subroutines well. If used it is best to start on a rapid move.

10.5.16. Laser buttons

The LASER ON/OFF button in intended to turn an output on or off which is connected to a small laser crosshair projector. When the crosshair is positioned over a desired reference point on the workpiece, the REF LASER button can be pushed, which then sets the X and Y offsets to the values indicated by the LASER OFFSET fields in the Settings page.

10.5.17. Tabs Description

Tabs allow the user to select the most appropriate info/control on the top three panels. If the on screen keyboard is showing and the user wishes to hide it but keep the current tab, they can do that by pressing the *HIDE* button on the virtual keyboard. In QtDragon, there is a splitter handle between the G-code text display and the G-code graphical display. One can use this to split the size between the two areas. This can be set differently in each tab and in each mode. The positions will be remembered.

10.5.17.1. Main tab

This tab displays the graphical representation of the current program. The side buttons will control the display.

- *User View*: Select/restore a user set view of the current program.
- *P*,*X*,*Y*,*Z*: Set standard views.
- *D*: Toggle display of dimensions.
- +, -: Zoom controls.
- *C*: Clear graphics of tool movement lines.

In `qtdragon_hd` there are also macro buttons available on the right side. Up to tens buttons can be defined in the INI.

10.5.17.2. File Tab

You can use this tab to load or transfer programs. Editing of G-code programs can be selected from this tab. With `qtdragon_hd`, this is where you can load the *G-code Ripper*.

10.5.17.3. Offsets Tab

You can monitor/modify system offsets from this tab. There are convenience buttons for zeroing the rotation.G92 and current G5x user offset.

10.5.17.4. Tool Tab

You can monitor/modify tool offsets from this tab. Adding and deleting tools from the tool file can also be done from this tab. When this tab is selected the individual home buttons in the DRO area will change to tool offset setting buttons. They will return to home buttons when you select another tab. Pressing this tool button will drop down a when menu of options:

- Set Current Tool Position
- Adjust Current Tool Position
- Zero Current Tool Position
- Set Tool Offset Directly
- Reset To Last

10.5.17.5. Status Tab

A time-stamped log of important machine or system events will be shown here. Machine events would be more suited to an operator, where the system events may help in debugging problems.

10.5.17.6. Probe Tab

Probing routines options are displayed on this tab. Depending on INI options, this could be VersaProbe or BasicProbe style. They are functionally similar. QtDragon_hd will also show a smaller graphics display window.

10.5.17.7. Camview Tab

If the recognized webcam is connected, this tab will display the video image overlaid with a cross-hair, circle and degree readout. This can be adjusted to suit a part feature for such things as touchoff. The underlying library uses openCV Python module to connect to the webcam.
To adjust the X or Y size aspect ratio (in percent), camera port number, API backend, or requested resolution look in the preference file for:

```
[CUSTOM_FORM_ENTRIES]
Camview xscale = 100
Camview yscale = 100
Camview cam number = 0
Camview cam api = V4L2
Camview cam resolution = 1280,720
```

Scales are in percent, usually the range will be 100 - 200 in one axis.
Negating these scales can be used to flip the image in X, Y or both axes.
API comes from openCV, the available backends will be listed if -V debugging is used. Set to *ANY* for opencv to choose.
Set resolution to DEFAULT for opencv to choose. Available resolutions will be listed if -V debugging is used.

nota
The preference file can only be edited when QtDragon is not running.

10.5.17.8. G-codes Tab

This tab will display a list of LinuxCNC's G-code. if you click on a line, a description of the code will be displayed.

10.5.17.9. Setup Tab

It's possible to load HTML or PDF file (.html / .pdf ending) with setup notes, and will be displayed in the setup tab.
If you load a G-code program and there is an HTML/PDF file of the same name, it will load automatically.
Some program, such as Fusion 360 and Aspire will create these files for you. You can also write your own HTML docs with the included SetUp Writer button.
There are three sub tabs:

- *HTML* - any loaded HTML pages are displayed here. The navigation buttons work on this page.
- *PDF* - any loaded PDF setup pages are displayed here.
- *PROPERTIES* - when a program is loaded its G-code properties are displayed here.

Hay botones de navegación para la página HTML:

- The up arrow returns you to the default HTML page.
- The left arrow moves backward one HTML page.
- The right arrow moves forward one HTML page.

If you wish to include a custom default HTML page, name it *default_setup.html* and place it in your configuration folder.
Custom QtVCP panels can be displayed in this tab by setting the EMBED_TAB_LOCATION option to *tabWidget_setup*.



Figura 10.35: QtDragon - Setup Tab Sample

10.5.17.10. Settings Tab

The settings tab is used to set running options, probing/touchplate/laser/camera offsets and load debugging external programs.

10.5.17.11. Utilities Tab

This tab will display another tab selection of G-code utility programs (and any embedded panels):

- *Facing*: allows quick face milling of a definable area at angles of 0,45 and 90 degrees.
- *Hole Circle*: allows quick setting of a program to drill a bolt circle of definable diameter and number of holes.
- *NGCGUI*: is a QtVCP version of the popular G-code subroutine builder/selector, see Widgets-NGCGUI.
- *Hole Enlarge*: allows milling a preexisting hole larger.

These tabs are detachable from the main screen by pressing the small arrow key on the far right of the tab header.
You can close the panel by pressing the arrow again or closing the window with the x button.
The last size and location of the detached window will be remembered each time the window is closed.
Custom QtVCP panels can be displayed here by setting the EMBED_TAB_LOCATION option to `tabWidget_ut`

10.5.17.12. User Tab

This tab will only be displayed if an embedded panel has been designated for the location `stackedWidget_mai`
If more then one embedded tab has been designated, then pressing the user tab will cycle through them.

10.5.18. Styles

Nearly all aspects of the GUI appearance are configurable via the QtDragon.qss stylesheet file. The file can be edited manually or through the stylesheet dialog widget in the GUI. To call up the dialog, press F12 on the main window. New styles can be applied temporarily and then saved to a new qss file, or overwrite the current qss file.

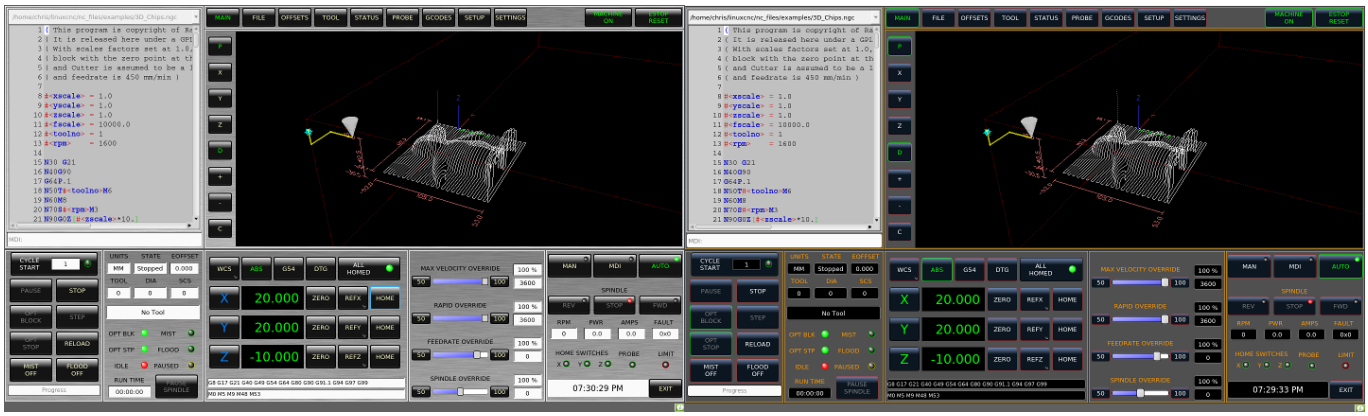


Figura 10.36: QtDragon - Two Style Examples

10.5.19. Internationalisation

It is possible to create translation files for QtDragon to display in the language of the current locale.

To create and or edit a translation file requires that LinuxCNC has been installed as run in place.

The following assumes that the LinuxCNC git directory is ~/linuxcnc-dev.

---

nota
If using QtDragon_hd substitute *qtdragon_hd* for *qtdragon*

Todos los archivos de idioma se guardan en ~/linuxcnc-dev/share/screens/qtdragon/languages.

The qtdragon.py file is a Python version of the GUI file used for translation purposes.

The .ts files are the translation source files for the translations. These are the files that require creating/editing for each language.

The .qm files are the compiled translation files used by pyqt.

The language is determined by an underscore plus the first two letters of the locale, for example if an Italian translation was being done then it would be `_it`. It will be referred to as `_xx` in this document, so `qtdragon_xx.ts` in this document would actually be `qtdragon_it.ts` for an Italian translation.

The default locale for QtDragon is `_en` which means that any translation files created as `qtdragon_en.*` will not be used for translations.

If any of the required utilities (pyuic5, pylupdate5, linguist) are not installed then the user will need to install the required development tools:

```
sudo apt install qttools5-dev-tools pyqt5-dev-tools
```

Cambiar al directorio de idiomas:

```
cd ~/linuxcnc-dev/share/qtvcp/screens/qtdragon/languages
```

If any text changes have been made to the GUI then run the following to update the GUI Python file:

```
pyuic5 ../qtdragon.ui > qtdragon.py
```

The user can either create a new translation source file for a non-existing language translation or modify an existing translation source file due to changes being made to some text in a QtDragon source file. If modifying an existing translation that has had no source file changes then this step is not required.

Crear o editar un archivo .ts:

```
./langfile xx
```

---

nota
this command is a script which runs the following: $ pylupdate5 **.py ../**.py ../../../../../lib/python/qtvcp/lib/qtdragon/*.py -ts qtdragon_xx.ts

---

The editing of the translation is done with the linguist application:

```
lingüista
```

1. Open the TS file and translate the strings

It is not necessary to provide a translation for every text string, if no translation is specified for a string then the original string will be used in the application. The user needs to be careful with the length of strings that appear on widgets as space is limited. If possible try to make the translation no longer than the original.

Cuando termine de editar guarde el archivo:
`Archivo -> Guardar`

Then create the .qm file:
`File -> Release`

QtDragon will be translated to the language of the current locale on the next start so long as a .qm file exists in that language.

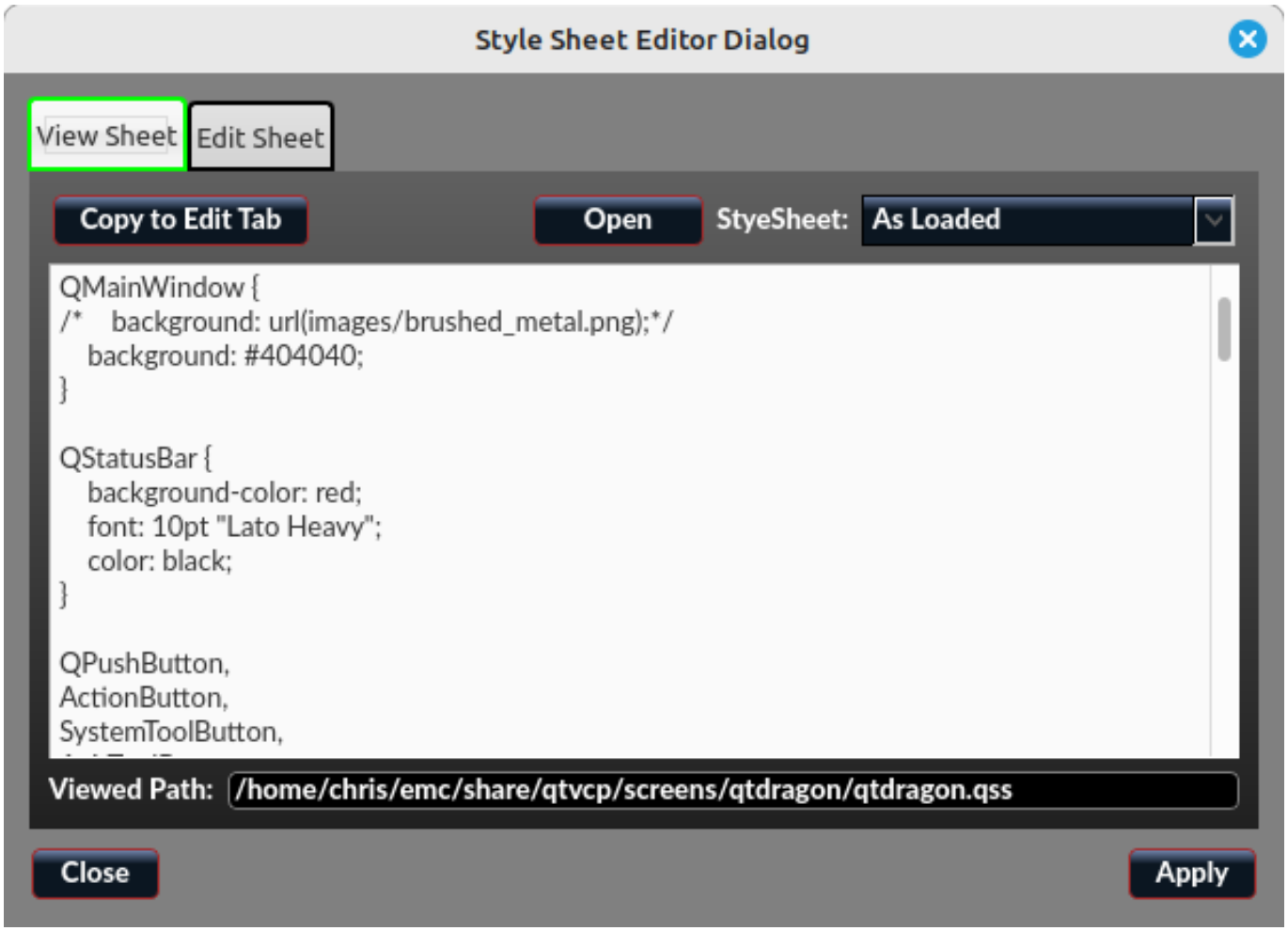
Users are welcome to submit translation files for inclusion into QtDragon. The preferred method is to submit a pull request from the users GitHub account as described in the contributing to LinuxCNC documentation. The only files required to be committed are qtdragon_xx.ts and qtdragon_xx.qm.

10.5.20. Customization

A general overview of Customizing Stock Screens.

10.5.20.1. Stylesheets

Stylesheets can be leveraged to do a fair amount of customization, but you usually need to know a bit about the widget names. Pressing F12 will display a stylesheet editor dialog to load/test/save modification.
The *View Sheet* tab will allow you to select and apply what stylesheet QtDragon will use when it's first loaded.
Press the button *Copy to Edit Tab* to copy the current stylesheet to the edit tab.
The *Edit Sheet* tab allows editing, applying and saving of changes od the displayed stylesheet.



Sometimes these lines will be present and you can change them, otherwise you will need to add them.

For instance, to change the DRO font (look for this entry and change the font name):

```
DROLabel,
StatusLabel#status_rpm {
    border: 1px solid black;
    border-radius: 4px;
    font: 20pt ”Noto Mono”;
}
```

To change the DRO display font and display format:

```
DROLabel {
    font: 25pt ”Lato Heavy”;
```

```
    qproperty-imperial_template: '%9.5f';
    qproperty-metric_template: '%10.4f';
    qproperty-angular_template: '%11.2f';

    /*Adjust the menu options */
    qproperty-showLast: true;
    qproperty-showDivide : false;
    qproperty-showGotoOrigin: false;
    qproperty-showZeroOrigin: false;
    qproperty-showSetOrigin: true;
    qproperty-dialogName: CALCULATOR;
}
```

Change the axis select button's click menu items:

```
AxisToolButton {
    /* Adjust all the menu options */
    qproperty-showLast: false;
    qproperty-showDivide : true;
    qproperty-showGotoOrigin: true;
    qproperty-showZeroOrigin: true;
    qproperty-showSetOrigin: false;
    qproperty-dialog_code_string: CALCULATOR;
}
```

To change the text of the mist button to *air* (add these lines)

```
#action_mist{
    qproperty-true_state_string: "Air\\nOn";
    qproperty-false_state_string: "Air\\nOff";
}
```

To change the Offsets display font and format:

```
ToolOffsetView {
    font: 20pt "Lato Heavy";
    qproperty-imperial_template: '%9.1f';
    qproperty-metric_template: '%10.1f';

}

OriginOffsetView {
    font: 12pt "Lato Heavy";
    qproperty-imperial_template: '%9.1f';
    qproperty-metric_template: '%10.1f';
}
```

To stop the blur effect with dialogs:

```
#screen_options {
    qproperty-focusBlur_option: false;
}
```

To change status highlight/selection colors:

```
#screen_options {
    qproperty-user1Color: white;     /*  default status     */
    qproperty-user2Color: #ff9000;   /*  warning status     */
    qproperty-user3Color: #ff8a96;   /*  critical status    */
    qproperty-user4Color: #ffaa00;   /*  MPG select         */
    qproperty-user5Color: #ff0000;   /*  Cycle Start select */
}
```

Change the G-code text display colors/fonts:

```
EditorBase{
background:black;
qproperty-styleColorCursor:white;
qproperty-styleColorBackground:grey;
qproperty-styleColor0: black;
qproperty-styleColor1: darkblue;
qproperty-styleColor2: blue;
qproperty-styleColor3: red;
qproperty-styleColor4: lightblue;
qproperty-styleColor5: white;
qproperty-styleColor6: lightGreen;
qproperty-styleColor7: yellow ;
qproperty-styleColorSelectionText: white;
qproperty-styleColorSelectionBackground: blue;
qproperty-styleFont0: ”Times,15,-1,5,90,0,0,1,1,0”;
qproperty-styleFont1: ”Times,15,-1,5,90,1,0,1,0,0”;
qproperty-styleFont2: ”Times,15,-1,5,90,0,0,1,1,0”;
qproperty-styleFont3: ”Times,15,-1,5,90,0,0,1,1,0”;
qproperty-styleFont4: ”Times,15,-1,5,90,0,0,1,1,0”;
qproperty-styleFont5: ”Times,15,-1,5,90,0,0,1,1,0”;
qproperty-styleFont6: ”Times,15,-1,5,90,0,0,1,1,0”;
qproperty-styleFont7: ”Times,15,-1,5,90,0,0,1,1,0”;
}
```

To have the manual spindle buttons also incrementally increase/decrease speed:

```
#action_spindle_fwd{
    qproperty-spindle_up_action: true;
}
#action_spindle_rev{
    qproperty-spindle_down_action: true;
}
```

10.5.20.2. Qt Designer and Python code

All aspects of the GUI are fully customization through Qt Designer and/or Python code.
This capability is included with the QtVCP development environment.
The extensive use of QtVCP widgets keeps the amount of required Python code to a minimum, allowing relatively easy modifications.
The LinuxCNC website has extensive documentation on the installation and use of QtVCP libraries.
See QtVCP for more information about QtVCP in general.
Custom modifications can be added by *subclassing* the handler file. This adds code on top of the original.
QtDragon can also utilize QtVCP’s rc file to do minor python code modifications without using a custom handler file.

See Modifying Screens for more information about customization.

Some widget customization is available for basic probe and versa probe.

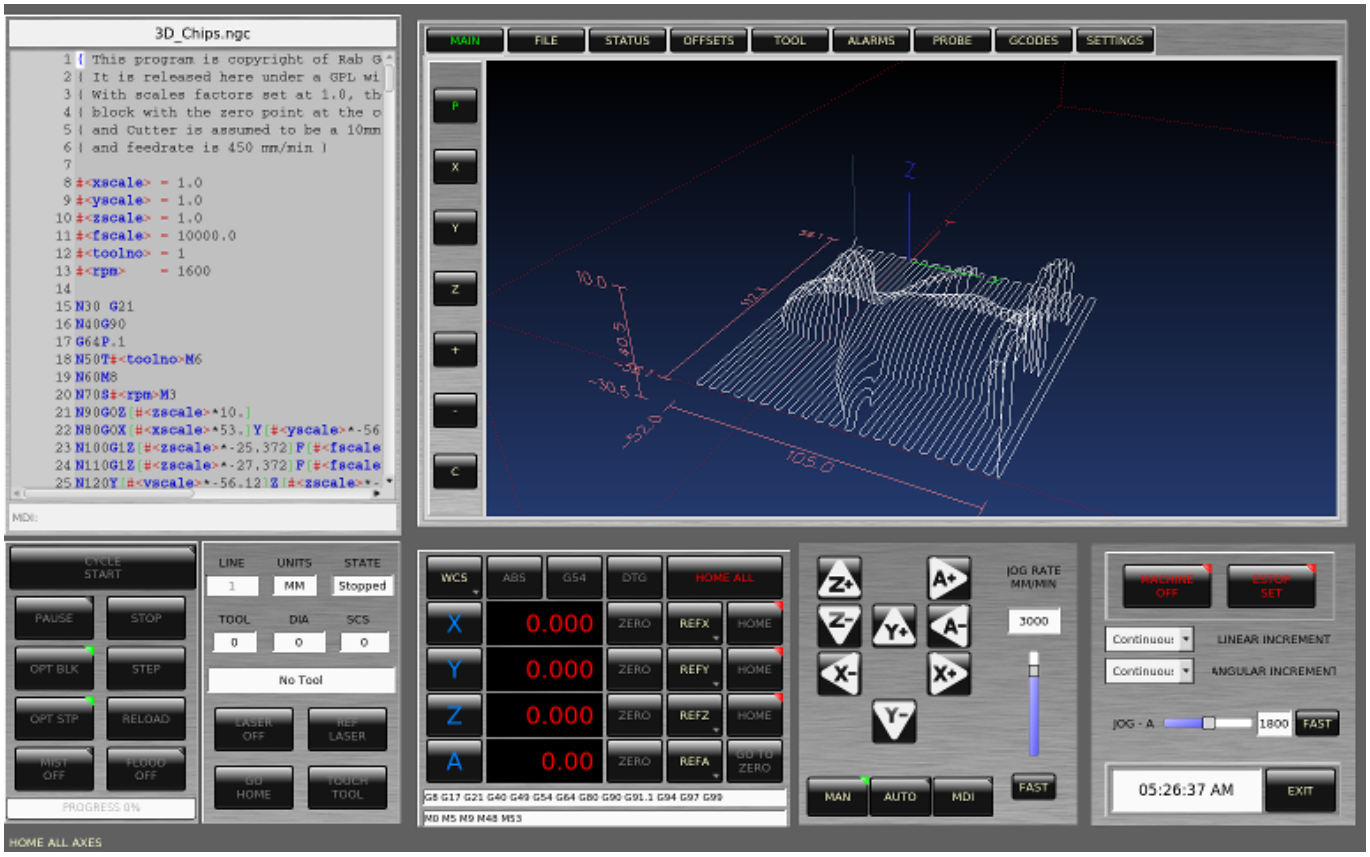


Figura 10.37: QtDragon - Customized QtDragon

10.6. NGCGUI

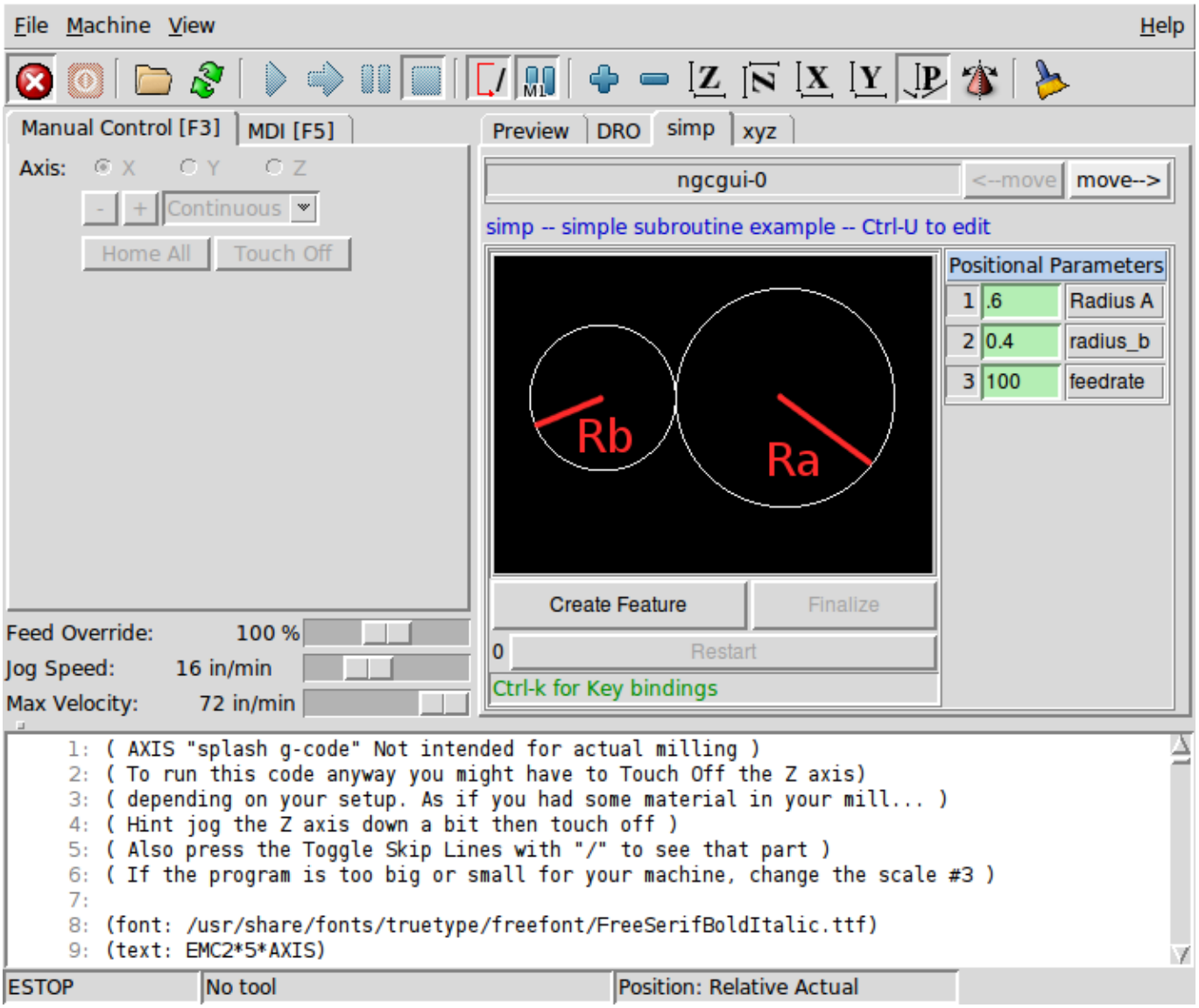


Figura 10.38: NGCGUI embedded into AXIS

10.6.1. Descripción general

* *NGCGUI* is a Tcl application to work with subroutines. It allows you to have a conversational interface with LinuxCNC. You can organize the subroutines in the order you need them to run and concatenate the subroutines into one file for a complete part program. * *NGCGUI* puede ejecutarse como una aplicación independiente o puede integrarse en múltiples páginas de pestañas en la GUI AXIS. * *PyNGCGUI* is an alternate, Python implementation of NGCGUI. * *PyNGCGUI* puede ejecutarse como una aplicación independiente o puede integrarse como una página de pestañas (con su propio conjunto de múltiples pestañas de subrutinas) en cualquier IGU que admita incrustación de aplicaciones GladeVCP; AXIS, Touchy, Gscreen y GMOCCAPY.

Using NGCGUI or PyNGCGUI:

- Se proporcionan pestañas para cada subrutina especificada en el archivo INI.

- Se pueden agregar nuevas pestañas de subrutinas sobre la marcha usando pestaña personalizada.
- Cada pestaña de subrutina proporciona cuadros de entrada para todos los parámetros de la subrutina.
- Los cuadros de entrada pueden tener un valor predeterminado y una etiqueta que se identifican por comentarios especiales en el archivo de subrutina.
- Las invocaciones de subrutina se pueden concatenar juntas para formar un programa de múltiples pasos.
- Se puede usar cualquier subrutina de código G de un solo archivo que se ajuste a las convenciones NGCGUI.
- Any gcmc (G-code-meta-compiler) program that conforms to NGCGUI conventions for tagging variables can be used. (The gcmc executable must be installed separately, see: https://www.vagrearg.org/-content/gcmc)

---

nota
NGCGUI and PyNGCGUI implement the same functions and both process .ngc and .gcmc files that conform to a few NGCGUI-specific conventions. In this document, the term *NGCGUI* generally refers to either application.

10.6.2. Demonstration Configurations

A number of demonstration configurations are located in the sim directory of the Sample Configurations offered by the LinuxCNC configuration picker. The configuration picker is on the system's main menu: Applications > CNC > LinuxCNC

Examples are included for the AXIS, Touchy, gscreen, and GMOCCAPY. These examples demonstrate both 3-axis (XYZ) cartesian configurations (like mills) and lathe (XZ) setups. Some examples show the use of a pop up keyboard for touch screen systems and other examples demonstrate the use of files created for the gcmc (G-code Meta Compiler) application. The touchy examples also demonstrate incorporation of a GladeVCP back plot viewer (gremlin_view).

The simplest application is found as:

```
Sample Configurations/sim/axis/ngcgui/ngcgui_simple
```

A comprehensive example showing gcmc compatibility is at:

```
Sample Configurations/sim/axis/ngcgui/ngcgui_gcmc
```

A comprehensive example embedded as a GladeVCP app and using gcmc is at:

```
Sample Configurations/sim/gscreen/ngcgui/pyngcgui_gcmc
```

The example sim configurations make use of library files that provide example G-code subroutine (.ngc) files and G-code-meta-compiler (.gcmc) files:

- *nc_files/ngcgui_lib*
  - *ngcgui.ngc* - An easy to understand example using subroutines
  - *arc1.ngc* - basic arc using cutter radius compensation
  - *arc2.ngc* - arc speced by center, offset, width, angle (calls arc1)
  - *backlash.ngc* - routine to measure an axis backlash with dial indicator
  - *db25.ngc* - creates a DB25 plug cutout

- *gosper.ngc* - a recursion demo (flowsnake)
- *helix.ngc* - helix or D-hole cutting
- *helix_rtheta.ngc* - helix or D-hole positioned by radius and angle
- *hole_circle.ngc* - equally spaced holes on a circle
- *ihex.ngc* - internal hexagon
- *iquad.ngc* - internal quadrilateral
- *ohex.ngc* - outside hexagon
- *oquad.ngc* - outside quadrilateral
- *qpex_mm.ngc* - demo of qpockets (mm based)
- *qpex.ngc* - demo of qpockets (inch based)
- *qpocket.ngc* - quadrilateral pocket
- *rectangle_probe.ngc* - probe a rectangular area
- *simp.ngc* - a simple subroutine example that creates two circles
- *slot.ngc* - slot from connecting two endpoints
- *xyz.ngc* - machine exerciser constrained to a box shape
- *Custom* - Creates custom tabs
- *ttt* - True Type Tracer, to create texts to be engraved

- *nc_files/ngcgui_lib/lathe*
  - *ngcgui-lathe* - Example lathe subroutine
  - *g76base.ngc* - GUI for G76 threading
  - *g76diam.ngc* - threading speced by major, minor diameters
  - *id.ngc* - bores the inside diameter
  - *od.ngc* - turns the outside diameter
  - *taper-od.ngc* - turns a taper on the outside diameter
  - *Custom* - Creates custom tabs
- *nc_files/gcmc_lib*
  - *drill.gcmc* - drill holes in rectangle pattern
  - *square.gcmc* - simple demo of variable tags for gcmc files
  - *star.gcmc* - gcmc demo illustrating functions and arrays
  - *wheels.gcmc* - gcmc demo of complex patterns

To try a demonstration, select a sim configuration and start the LinuxCNC program.

If using the AXIS GUI, press the *E-Stop*  then *Machine Power*  then *Home All*. Pick a NGCGUI tab, fill in any empty blanks with sensible values and press *Create Feature* then *Finalize*. Finally press the *Run*  button to watch it run. Experiment by creating multiple features and features from different tab pages.

Para crear varias subrutinas concatenadas en un solo archivo, vaya a cada pestaña complete los espacios en blanco, presione *Crear función* y luego, con las teclas de flecha, mueva las pestañas necesarias para ponerlas en orden. Ahora presione *Finalizar* y responda el mensaje para crear

Other GUIs will have similar functionality but the buttons and names may be different.

nota
The demonstration configs create tab pages for just a few of the provided examples. Any GUI with a custom tab can open any of the library example subroutines or any user file if it is in the LinuxCNC subroutine path.
To see special key bindings, click inside an NGCGUI tab page to get focus and then press Control-k.
The demonstration subroutines should run on the simulated machine configurations included in the distribution. A user should always understand the behavior and purpose of a program before running on a real machine.

10.6.3. Library Locations

In LinuxCNC installations installed from deb packages, the simulation configs for NGCGUI use symbolic links to non-user-writable LinuxCNC libraries for:

- *nc_files/ngcgui_lib* NGCGUI-compatible subfiles
- *nc_files/ngcgui_lib/lathe* NGCGUI-compatible lathe subfiles
- *nc_files/gcmc_lib* NGCGUI-gcmc-compatible programs
- *nc_files/ngcgui_lib/utilitysubs* Helper subroutines
- *nc_files/ngcgui_lib/mfiles* User M files

These libraries are located by INI file items that specify the search paths used by LinuxCNC (and NGCGUI):

```
[RS274NGC]
SUBROUTINE_PATH = ../../nc_files/ngcgui_lib:../../nc_files/gcmc_lib:../../nc_files/ ↩
    ngcgui_lib/utilitysubs
USER_M_PATH     = ../../nc_files/ngcgui_lib/mfiles
```

nota
These are long lines (not continued on multiple lines) that specify the directories used in a search patch. The directory names are separated by colons (:). No spaces should occur between directory names.

A user can create new directories for their own subroutines and M-files and add them to the search path(s).

For example, a user could create directories from the terminal with the commands:

```
mkdir /home/myusername/mysubs
mkdir /home/myusername/mymfiles
```

And then create or copy system-provided files to these user-writable directories. For instance, a user might create a NGCGUI-compatible subfile named:

```
/home/myusername/mysubs/example.ngc
```

To use files in new directories, the INI file must be edited to include the new subfiles and to augment the search path(s). For this example:

```
[RS274NGC]
...
SUBROUTINE_PATH = /home/myusername/mysubs:../../nc_files/ngcgui_lib:../../nc_files/gcmc_lib ↩
    :../../nc_files/ngcgui_lib/utilitysubs
USER_M_PATH     = /home/myusername/mymfiles:../../nc_files/ngcgui_lib/mfiles

[DISPLAY]
...
NGCGUI_SUBFILE = example.ngc
...
```

LinuxCNC (and NGCGUI) use the first file found when searching directories in the search path. With this behavior, you can supersede an ngcgui_lib subfile by placing a subfile with an identical name in a directory that is found earlier in the path search. More information can be found in the INI chapter of the Integrators Manual.

10.6.4. Standalone Usage

10.6.4.1. Standalone NGCGUI

Para su uso, escriba en una terminal:

```
ngcgui --help
Usage:
  ngcgui --help | -?
  ngcgui [Options] -D <nc files directory name>
  ngcgui [Options] -i <LinuxCNC INI file name>
  ngcgui [Options]

Options:
  [-S subroutine_file]
  [-p preamble_file]
  [-P postamble_file]
  [-o output_file]
  [-a autosend_file]            (autosend to AXIS default:auto.ngc)
  [--noauto]                    (no autosend to AXIS)
  [-N | --nom2]                 (no m2 terminator (use %))
  [--font [big|small|fontspec]] (default: ”Helvetica -10 normal”)
  [--horiz|--vert]              (default: --horiz)
  [--cwidth comment_width]      (width of comment field)
  [--vwidth varname_width]      (width of varname field)
  [--quiet]                     (fewer comments in outfile)
  [--noiframe]                  (default: frame displays image)
```

nota
As a standalone application, NGCGUI handles a single subroutine file which can be invoked multiple times. Multiple standalone NGCGUI applications can be started independently.

10.6.4.2. Standalone PyNGCGUI

Para su uso, escriba en una terminal:

```
pyngcgui --help
Usage:
pyngcgui [Options] [<sub_filename>]
```

```
Options requiring values:
    [-d | --demo] [0|1|2] (0: DEMO standalone toplevel)
                          (1: DEMO embed new notebook)
                          (2: DEMO embed within existing notebook)
    [-S | --subfile       <sub file name>]
    [-p | --preamble      <preamble file name>]
    [-P | --postamble     <postamble file name>]
    [-i | --ini           <INI file name>]
    [-a | --autofile      <auto file name>]
    [-t | --test          <testno>]
    [-K | --keyboardfile  <glade_file>] (use custom popupkeyboard glade file)
Solo Options:
    [-v | --verbose]
    [-D | --debug]
    [-N | --nom2]         (no m2 terminator (use %))
    [-n | --noauto]       (save but do not automatically send result)
    [-k | --keyboard]     (use default popupkeybaord)
    [-s | --sendtoaxis]   (send generated NGC file to AXIS GUI)
Notes:
      A set of files is comprised of a preamble, subfile, postamble.
      The preamble and postamble are optional.
      One set of files can be specified from cmdline.
      Multiple sets of files can be specified from an INI file.
      If --ini is NOT specified:
         search for a running LinuxCNC and use its INI file.
```

nota
As a standalone application, PyNGCGUI can read an INI file (or a running LinuxCNC application) to create tab pages for multiple subfiles.

10.6.5. Embedding NGCGUI

10.6.5.1. Embedding NGCGUI in AXIS

The following INI file items go in the [DISPLAY] section. (See additional sections below for additional items needed)

- *TKPKG = Ngcgui 1.0* - the NGCGUI package
- *TKPKG = Ngcguittt 1.0* - the True Type Tracer package for generating text for engraving (optional, must follow TKPKG = Ngcgui).
- *NGCGUI_FONT = Helvetica -12 normal* - Sets the font used
- *NGCGUI_PREAMBLE = in_std.ngc* - The preamble file to be added at the beginning of the subroutine. When several subroutines are concatenated, it is only added once.
- *NGCGUI_SUBFILE = simp.ngc* - Creates a tab from the named subroutine.
- *NGCGUI_SUBFILE = ””* - Creates a custom tab
- *#NGCGUI_OPTIONS = opt1 opt2 …* - Opciones de NGCGUI:
  - *nonew* — Prohibe la creación de nuevas pestañas personalizadas
  - *noremove* — Prohibe eliminar una página de pestaña
  - *noauto* — No se ejecuta automáticamente (makeFile, luego ejecución manual)

- *noiframe* — Sin imagen interna, imagen en un nivel máximo por separado

- *TTT = truetype-tracer* - nombre del programa truetype tracer (debe estar en la RUTA del usuario)
- *TTT_PREAMBLE = in_std.ngc* - Opcional, especifica el nombre del archivo para el preámbulo utilizado para subfiles ttt creados. (alternativo: mm_std.ngc)

nota
The optional truetype tracer items are used to specify an NGCGUI-compatible tab page that uses the application truetype-tracer. The truetype-tracer application must be installed independently and located in the user PATH.

10.6.5.2. Embedding PyNGCGUI as a GladeVCP tab page in a GUI

The following INI file items go in the [DISPLAY] section for use with the AXIS, Gscreen, or Touchy GUIs. (See additional sections below for additional items needed)

EMBED_ Items

- `EMBED_TAB_NAME = PyNGCGUI` - name to appear on embedded tab
- `EMBED_TAB_COMMAND = gladevcp -x {XID} pyngcgui_axis.ui` - invokes GladeVCP
- `EMBED_TAB_LOCATION = name_of_location` - where the embedded page is located

nota
The EMBED_TAB_LOCATION specifier is not used for the AXIS GUI. While PyNGCGUI can be embedded in AXIS, integration is more complete when using NGCGUI (using TKPKG = Ngcgui 1.0). To specify the EMBED_TAB_LOCATION for other GUIs, see the DISPLAY Section of the INI Configuration Chapter.

nota
The truetype tracer GUI front-end is not currently available for GladeVCP applications.

10.6.5.3. Additional INI File items required for NCGUI or PyNGCGUI

The following INI file items go in the [DISPLAY] section for any GUI that embeds either NGCGUI or PyNGCGUI.

- *NGCGUI_FONT = Helvetica -12 normal* - specifies the font name,size, normal|bold
- *NGCGUI_PREAMBLE = in_std.ngc* - the preamble file to be added in front of the subroutines. When concatenating several common subroutine invocations, this preamble is only added once. For mm-based machines, use mm_std.ngc
- *NGCGUI_SUBFILE = filename1.ngc* - creates a tab from the filename1 subroutine
- *NGCGUI_SUBFILE = filename2.ngc* - creates a tab from the filename2 subroutine
- *... etc.*
- *NGCGUI_SUBFILE = gcmcname1.gcmc* - creates a tab from the gcmcname1 file
- *NGCGUI_SUBFILE = gcmcname2.gcmc* - creates a tab from the gcmcname2 file

- *... etc.*
- *NGCGUI_SUBFILE = ””* - creates a custom tab that can open any subroutine in the search path
- *NGCGUI_OPTIONS = opt1 opt2 ...* - NGCGUI options
  - *nonew* - disallow making a new custom tab
  - *noremove* - disallow removing any tab page
  - *noauto* - no autosend (use makeFile, then save or manually send)
  - *noiframe* - no internal image, display images on separate top level widget
  - *nom2* - do not terminate with m2, use % terminator. This option eliminates all the side effects of m2 termination
- *GCMC_INCLUDE_PATH = dirname1:dirname2* - search directories for gcmc include files

This is an example of embedding NGCGUI into AXIS. The subroutines need to be in a directory specified by the [RS274NGC]SUBROUTINE_PATH. Some example subroutines use other subroutines so check to be sure you have the dependences, if any, in a SUBROUTINE_PATH directory. Some subroutines may use custom M-files which must be in a directory specified by the [RS274NGC]USER_M_PATH.

The G-code-meta-compiler (gcmc) can include statements like:

```
include(”filename.inc.gcmc”);
```

By default, gcmc includes the current directory which, for LinuxCNC, will be the directory containing the LinuxCNC INI file. Additional directories can be prepended to the gcmc search order with the GCMC_INCLUDE_PATH item.

Sample AXIS-GUI-based INI

```
[RS274NGC]
...
SUBROUTINE_PATH   = ../../nc_files/ngcgui_lib:../../ngcgui_lib/utilitysubs
USER_M_PATH       = ../../nc_files/ngcgui_lib/mfiles

[DISPLAY]
TKPKG             = Ngcgui    1.0
TKPKG             = Ngcguittt 1.0
# Ngcgui must precede Ngcguittt

NGCGUI_FONT       = Helvetica -12 normal
# specify filenames only, files must be in [RS274NGC]SUBROUTINE_PATH
NGCGUI_PREAMBLE   = in_std.ngc
NGCGUI_SUBFILE    = simp.ngc
NGCGUI_SUBFILE    = xyz.ngc
NGCGUI_SUBFILE    = iquad.ngc
NGCGUI_SUBFILE    = db25.ngc
NGCGUI_SUBFILE    = ihex.ngc
NGCGUI_SUBFILE    = gosper.ngc
# specify ”” for a custom tab page
NGCGUI_SUBFILE    = ””
#NGCGUI_SUBFILE   = ”” use when image frame is specified if
#                     opening other files is required
#                     images will be put in a top level window
NGCGUI_OPTIONS    =
#NGCGUI_OPTIONS   = opt1 opt2 ...
# opt items:
#   nonew      -- disallow making a new custom tab
#   noremove   -- disallow removing any tab page
#   noauto     -- no auto send (makeFile, then manually send)
#   noiframe   -- no internal image, image on separate top level
```

```
GCMC_INCLUDE_PATH = /home/myname/gcmc_includes

TTT               = truetype-tracer
TTT_PREAMBLE      = in_std.ngc

PROGRAM_PREFIX    = ../../nc_files
```

nota
The above is not a complete AXIS GUI INI — the items show are those used by NGCGUI. Many additional items are required by LinuxCNC to have a complete INI file.

10.6.5.4. Truetype Tracer

Ngcgui_ttt provides support for truetype-tracer (v4). It creates an AXIS tab page which allows a user to create a new NGCGUI tab page after entering text and selecting a font and other parameters. (Truetype-tracer must be installed independently).

To embed ngcgui_ttt in AXIS, specify the following items in addition to NGCGUI items:

```
Item:    [DISPLAY]TKPKG = Ngcgui_ttt version_number
Example: [DISPLAY]TKPKG = Ngcgui_ttt 1.0
Note:    Obligatorio, especifica la carga de ngcgui_ttt en una página de pestaña de eje  ↩
    llamada ttt.
         Debe seguir el elemento TKPKG = Ngcgui.

Item:    [DISPLAY]TTT = path_to_truetype-tracer
Example: [DISPLAY]TTT = truetype-tracer
Note:    Opcional, si no se especifica, intente usar /usr/local/bin/truetype-tracer.
         Especifique con ruta absoluta o como un nombre ejecutable simple,
         en cuyo caso el entorno PATH del usuario se usará para encontrar el programa.

Item:    [DISPLAY]TTT_PREAMBLE = preamble_filename
Example: [DISPLAY]TTT_PREAMBLE = in_std.ngc
Note:    Opcional, especifica el nombre de archivo para el preámbulo utilizado para los  ↩
    archivos subttt creados.
```

10.6.5.5. INI File Path Specifications

NGCGUI uses the LinuxCNC search path to find files. The search path begins with the standard directory specified by:

```
[DISPLAY]PROGRAM_PREFIX = directory_name
```

followed by multiple directories specified by:

```
[RS274NGC]SUBROUTINE_PATH = directory1_name:directory1_name:directory3_name ...
```

Directories Directories may be specified as absolute paths or relative paths.

- Example: `[DISPLAY]PROGRAM_PREFIX = /home/myname/linuxcnc/nc_files`
- Example: `[DISPLAY]PROGRAM_PREFIX = ~/linuxcnc/nc_files`
- Example: `[DISPLAY]PROGRAM_PREFIX = ../../nc_files`

Absolute Paths An absolute path beginning with a ”/” specifies a complete filesystem location. A path beginning with a ”~/” specifies a path starting from the user’s home directory. A path beginning with ”~username/” specifies a path starting in username’s home directory.

Rutas relativas Relative paths are based on the startup directory which is the directory containing the INI file. Using relative paths can facilitate relocation of configurations but requires a good understanding of Linux path specifiers.

- `./d0` is the same as d0, e.g., a directory named d0 in the startup directory
- `../d1` refers to a directory d1 in the parent directory
- `../../d2` refers to a directory d2 in the parent of the parent directory
- `../../../d3` etc.

Multiple directories can be specified with [RS274NGC]SUBROUTINE_PATH by separating them with colons. The following example illustrates the format for multiple directories and shows the use of relative and absolute paths.

Multiple Directories Example:

```
[RS274NGC]SUBROUTINE_PATH = ../../nc_files/ngcgui_lib:../../nc_files/ngcgui_lib/utilitysubs ↩
    :/tmp/tmpngc
```

This is one long line, do not continue on multiple lines. When LinuxCNC and/or NGCGUI searches for files, the first file found in the search is used.

LinuxCNC (and NGCGUI) must be able to find all subroutines including helper routines that are called from within NGCGUI subfiles. It is convenient to place utility subs in a separate directory as indicated in the example above.

The distribution includes the ngcgui_lib directory and demo files for preambles, subfiles, postambles and helper files. To modify the behavior of the files, you can copy any file and place it in an earlier part of the search path. The first directory searched is [DISPLAY]PROGRAM_PREFIX. You can use this directory but it is better practice to create dedicated directory(ies) and put them at the beginning of the [RS274NGC]SUBROUTINE_PATH.

In the following example, files in /home/myname/linuxcnc/mysubs will be found before files in ../../nc_files/ngc

Adding User Directory Example:

```
[RS274NGC]SUBROUTINE_PATH = /home/myname/linuxcnc/mysubs:../../nc_files/ngcgui_lib:../../ ↩
    nc_files/ngcgui_lib/utilitysubs‘
```

New users may inadvertently try to use files that are not structured to be compatible with NGCGUI requirements. NGCGUI will likely report numerous errors if the files are not coded per its conventions. Good practice suggests that NGCGUI-compatible subfiles should be placed in a directory dedicated to that,purpose and that preamble, postamble, and helper files should be in separate directory(ies) to discourage attempts to use them as subfiles. Files not intended for use as subfiles can include a special comment: ”(not_a_subfile)” so that NGCGUI will reject them automatically with a relevant message.

10.6.5.6. Summary of INI File item details for NGCGUI usage

[RS274NGC]SUBROUTINE_PATH = dirname1:dirname2:dirname3 ...
Ejemplo: `[RS274NGC]SUBROUTINE_PATH = ../../nc_files/ngcgui_lib:../../nc_files/ngcgui_li`
Nota: Opcional, pero muy útil para organizar subarchivos y archivos de utilidad.

[RS274NGC]USER_M_PATH = dirname1:dirname2:dirname3 ...
Ejemplo: `[RS274NGC]USER_M_PATH = ../../nc_files/ngcgui_lib/mfiles`
Nota: Opcional, necesario para localizar archivos M de usuario personalizados.

[DISPLAY]EMBED_TAB_NAME = name to display on embedded tab page
Ejemplo: `[DISPLAY]EMBED_TAB_NAME = Pyngcgui`
Nota: Las entradas: `EMBED_TAB_NAME`, `EMBED_TAB_COMMAND`, `EMBED_TAB_LOCATION` definen una aplicación integrada para varias GUIs LinuxCNC.

[DISPLAY]EMBED_TAB_COMMAND = programname followed by arguments
Ejemplo: `[DISPLAY]EMBED_TAB_COMMAND = gladevcp -x {XID} pyngcgui_axis.ui`
Nota: Para aplicaciones GladeVCP, vea el Capítulo GladeVCP.

[DISPLAY]EMBED_TAB_LOCATION = name_of_location
Ejemplo: `[DISPLAY]EMBED_TAB_LOCATION = notebook_main`
Nota: Vea archivos INI de ejemplo para ubicaciones posibles.
No requerido para la IGU AXIS.

[DISPLAY]PROGRAM_PREFIX = dirname
Ejemplo: `[DISPLAY]PROGRAM_PREFIX = ../../nc_files`
Nota: Obligatorio y necesario para numerosas funciones de LinuxCNC.
Es el primer directorio utilizado en la búsqueda de archivos.

[DISPLAY]TKPKG = NGCGUI version_number
Ejemplo: `[DISPLAY]TKPKG = Ngcgui 1.0`
Nota: Solo se requiere para la incrustación de IGU AXIS.
Especifica la carga de las pestañas de AXIS NGCGUI.

[DISPLAY]NGCGUI_FONT = font_descriptor
Ejemplo: [DISPLAY]NGCGUI_FONT = Helvetica -12 normal
Nota: Opcional, font_descriptor es un especificador de fuente compatible con tcl con elementos para fonttype -fontsize fontweight.
El valor predeterminado es: Helvetica -10 normal.
Los tamaños de fuente mas pequeños pueden ser útiles para pantallas pequeñas.
Los tamaños de fuente mas grandes pueden ser útiles para aplicaciones de pantalla táctil.

[DISPLAY]NGCGUI_SUBFILE = subfile_filename
Ejemplo: `[DISPLAY]NGCGUI_SUBFILE = simp.ngc`
Ejemplo: `[DISPLAY]NGCGUI_SUBFILE = square.gcmc`
Ejemplo: `[DISPLAY]NGCGUI_SUBFILE = ""`
Nota: Use uno o más elementos para especificar subarchivos compatibles con NGCGUI o programas gcmc que requieren una página de pestañas al inicio.
Se creará una pestaña "Personalizada" cuando el nombre de archivo sea "".
Un usuario puede usar una pestaña "Personalizada" para explorar el sistema de archivos e identificar archivos de preámbulo, subarchivo y epílogo.

[DISPLAY]NGCGUI_PREAMBLE = preamble_filename
Ejemplo: `[DISPLAY]NGCGUI_PREAMBLE = in_std.ngc`
Nota: Opcional, cuando se especifica, el archivo se antepone a un subarchivo.
Los archivos creados con pestañas "Personalizadas" usan el preámbulo especificado con la pagina.

[DISPLAY]NGCGUI_POSTAMBLE = postamble_filename
Ejemplo: `[DISPLAY]NGCGUI_POSTAMBLE = bye.ngc`
Nota: Opcional, cuando se especifica, el archivo se agrega a un subarchivo.
Los archivos creados con pestañas "Personalizadas" usan el postámbulo especificado con la pagina.

[DISPLAY]NGCGUI_OPTIONS = opt1 opt2 ...
Ejemplo: `[DISPLAY]NGCGUI_OPTIONS = nonew noremove`

Nota: Opciones múltiples separadas por espacios en blanco.
Por defecto, NGCGUI configura páginas de pestañas para que:
1) a user can make new tabs;
2) a user can remove tabs (except for the last remaining one);
3) finalized files are automatically sent to LinuxCNC;
4) an image frame (iframe) is made available to display an image for the subfile (if an image is provided);
5) the NGCGUI result file sent to LinuxCNC is terminated with an M2 (and incurs M2 side-effects).

The options `nonew`, `noremove`, `noauto`, `noiframe`, `nom2` respectively disable these default behaviors.

Por defecto, si un archivo de imagen (.png,.gif,jpg,pgm) se encuentra en el mismo directorio que el subarchivo, la imagen se muestra en el iframe. Especificando la opción `noiframe` pone a disposición botones adicionales para seleccionar un preámbulo, subarchivo y epílogo y casillas de verificación adicionales. Selecciones de las casillas de verificación siempre están disponibles con teclas especiales:
`Ctrl-R` Alternar ”Retener valores en la lectura de subarchivo”,
`Ctrl-E` Alternar ”Expandir subrutina”,
`Ctrl-a` Alternar ”Envío automático”,
`Ctrl-k` enumera todas las teclas y funciones.

If `noiframe` is specified and an image file is found, the image is displayed in a separate window and all functions are available on the tab page. The `NGCGUI_OPTIONS` apply to all NGCGUI tabs except that the `nonew`, `noremove`, and `noiframe` options are not applicable for ”Custom” tabs. Do not use ”Custom” tabs if you want to limit the user’s ability to select subfiles or create additional tab pages.

[DISPLAY]GCMC_INCLUDE_PATH = dirname1:dirname2:...
Ejemplo: `[DISPLAY]GCMC_INCLUDE_PATH =/home/myname/gcmc_includes:/home/myname/gcmc_inclu`
Nota: Opcional, cada directorio se incluirá cuando se invoque gcmc usando la opción: `--include nombre-de-directorio`.

10.6.6. File Requirements for NGCGUI Compatibility

10.6.6.1. Single-File Gcode (.ngc) Subroutine Requirements

An NGCGUI-compatible subfile contains a single subroutine definition. The name of the subroutine must be the same as the filename (not including the .ngc suffix). LinuxCNC supports named or numbered subroutines, but only named subroutines are compatible with NGCGUI. For more information see the O-Codes chapter.

La primera línea sin comentarios debe ser una sentencia `sub`.
La última línea sin comentarios debe ser una sentencia `endsub`.

examp.ngc:

```
(info: info_text_to_appear_at_top_of_tab_page)
; comment line beginning with semicolon
( comment line using parentheses)
o<examp> sub
  BODY_OF_SUBROUTINE
o<examp> endsub
; comment line beginning with semicolon
( comment line using parentheses)
```

The body of the subroutine should begin with a set of statements that define local named parameters for each positional parameter expected for the subroutine call. These definitions must be consecutive

beginning with #1 and ending with the last used parameter number. Definitions must be provided for each of these parameters (no omissions).

Parameter Numbering

```
#<xparm> = #1
#<yparm> = #2
#<zparm> = #3
```

LinuxCNC considers all numbered parameters in the range #1 thru #30 to be calling parameters, so NGCGUI provides entry boxes for any occurrence of parameters in this range. It is good practice to avoid use of numbered parameters #1 through #30 anywhere else in the subroutine. Using local, named parameters is recommended for all internal variables.

Each defining statement may optionally include a special comment and a default value for the parameter.

Statement Prototype

```
#<vname> = #n (=default_value)
or
#<vname> = #n (comment_text)
or
#<vname> = #n (=default_value comment_text)
```

Parameter Examples

```
#<xparm> = #1 (=0.0)
#<yparm> = #2 (Ystart)
#<zparm> = #3 (=0.0 Z start setting)
```

If a default_value is provided, it will be entered in the entry box for the parameter on startup. If comment_text is included, it will be used to identify the input instead of the parameter name.

Global Named Parameters Notes on global named parameters and NGCGUI:

(global named parameters have a leading underscore in the name, like #<_someglobalname>)

As in many programming languages, use of globals is powerful but can often lead to unexpected consequences. In LinuxCNC, existing global named parameters will be valid at subroutine execution and subroutines can modify or create global named parameters.

Se desaconseja pasar información a subrutinas utilizando parámetros con nombres globales, dado que dicho uso requiere el establecimiento y mantenimiento de un contexto global bien definido que es difícil de mantener. Usando los parámetros numerados de #1 a #30 como entradas de subrutina debería ser suficiente para satisfacer una amplia gama de requerimientos de diseño. NGCGUI supports some input global named parameter but their usage is obsolete and not documented here.

Si bien no se recomiendan los parámetros con nombre global de entrada, las subrutinas LinuxCNC deben usar parámetros globales con nombre para devolver resultados. Los subarchivos ngcgui-compatible están destinados al uso de GUI; los valores de retorno no son un requisito común. Sin embargo, ngcgui es útil como herramienta de prueba para subrutinas que devuelven parámetros con nombre global y es común que los subfiles compatibles con ngcgui llamen archivos de subrutina de utilidad que devuelven resultados con parámetros globales con nombre.

To support these usages, NGCGUI ignores global named parameters that include a colon (:) character in their name. Use of the colon (:) in the name prevents NGCGUI from making entryboxes for these parameters.

Global Named Parameters Example

```
o<examp> sub
...
#<_examp:result> = #5410       (return the current tool diameter)
```

```
...
o<helper> call [#<x1>] [#<x2>] (call a subroutine)
#<xresult> = #<_helper:answer> (immediately localize the helper global result)
#<_helper:answer> = 0.0        (nullify global named parameter used by subroutine)
...
o<examp> endsub
```

In the above example, the utility subroutine will be found in a separate file named helper.ngc. The helper routine returns a result in a global named parameter named #<_helper:answer.

For good practice, the calling subfile immediately localizes the result for use elsewhere in the subfile and the global named parameter used for returning the result is nullified in an attempt to mitigate its inadvertent use elsewhere in the global context. A nullification value of 0.0 may not always be a good choice.

NGCGUI supports the creation and concatenation of multiple features for a subfile and for multiple subfiles. It is sometimes useful for subfiles to determine their order at runtime, so NGCGUI inserts a special global parameter that can be tested within subroutines. The parameter is named #<_feature:>. Its value begins with a value of 0 and is incremented for each added feature.

Additional Features A special *info* comment can be included anywhere in an NGCGUI-compatible subfile. The format is:

```
(info: info_text)
```

The info_text is displayed near the top of the NGCGUI tab page in AXIS.

Files not intended for use as subfiles can include a special comment so that NGCGUI will reject them automatically with a relevant message.

```
(not_a_subfile)
```

An optional image file (.png,.gif,.jpg,.pgm) can accompany a subfile. The image file can help clarify the parameters used by the subfile. The image file should be in the same directory as the subfile and have the same name with an appropriate image suffix, e.g. the subfile example.ngc could be accompanied by an image file examp.png. NGCGUI attempts to resize large images by subsampling to a size with maximum width of 320 and maximum height of 240 pixels.

None of the conventions required for making an NGCGUI-compatible subfile preclude its use as general purpose subroutine file for LinuxCNC.

La distribución LinuxCNC incluye una biblioteca (directorio ngcgui_lib) que incluye archivos de utilidad y subfiles compatibles con NGCGUI de ejemplo para ilustrar las características de las subrutinas LinuxCNC y el uso de NGCGUI. Otra biblioteca (gcmc_lib) proporciona ejemplos de archivos de subrutina para el metacompilador de código G (gcmc).

Additional user sumitted subroutines can be found on the Forum in the Subroutines Section.

10.6.6.2. Gcode-meta-compiler (.gcmc) file requirements

Files for the Gcode-meta-compiler (gcmc) are read by NGCGUI and it creates entry boxes for variables tagged in the file. When a feature for the file is finalized, NGCGUI passes the file as input to the gcmc compiler and, if the compile is successful, the resulting G-code file is sent to LinuxCNC for execution. The resulting file is formatted as single-file subroutine; .gcmc files and .ngc files can be intermixed by NGCGUI.

The variables identified for inclusion in NGCGUI are tagged with lines that will appear as comments to the gcmc compiler.

Variable Tags Formats

```
//ngcgui: varname1 =
//ngcgui: varname2 = value2
//ngcgui: varname3 = value3, label3;
```

Variable Tags Examples

```
//ngcgui: zsafe =
//ngcgui: feedrate = 10
//ngcgui: xl = 0, x limit
```

Para estos ejemplos, el cuadro de entrada para varname1 no tendrá un valor predeterminado, el cuadro de entrada para varname2 tendrá un valor predeterminado de value2, y el cuadro de entrada para varname 3 tendrá un valor predeterminado de 3 y una etiqueta label3 (en lugar de varname3). Los valores predeterminados deben ser números.

To make it easier to modify valid lines in a gcmc file, alternate tag line formats accepted. The alternate formats ignore trailing semicolons (;) and trailing comment markers (//). With this provision, it is often makes it possible to just add the //ngcgui: tag to existing lines in a .gcmc file.

Alternate Variable Tags Formats

```
//ngcgui: varname2 = value2;
//ngcgui: varname3 = value3; //, label3;
```

Alternate Variable Tags Examples

```
//ngcgui: feedrate = 10;
//ngcgui: xl = 0; //, x limit
```

An info line that will appear at the top of a tab page may be optionally included with a line tagged as:

Info tag

```
//ngcgui: info: text_to_appear_at_top_of_tab_page
```

When required, options can be passed to the gcmc compiler with a line tagged:

Option line tag format

```
//ngcgui: -option_name [ [=] option_value]
```

Option line tag Examples

```
//ngcgui: -I
//ngcgui: --imperial
//ngcgui: --precision 5
//ngcgui: --precision=6
```

Options for gcmc are available with the terminal command:

```
gcmc --help
```

A gcmc program by default uses metric mode. The mode can be set to inches with the option setting:

```
//ngcgui: --imperial
```

A preamble file, if used, can set a mode (g20 or g21) that conflicts with the mode used by a gcmc file. To ensure that the gcmc program mode is in effect, include the following statement in the .gcmc file:

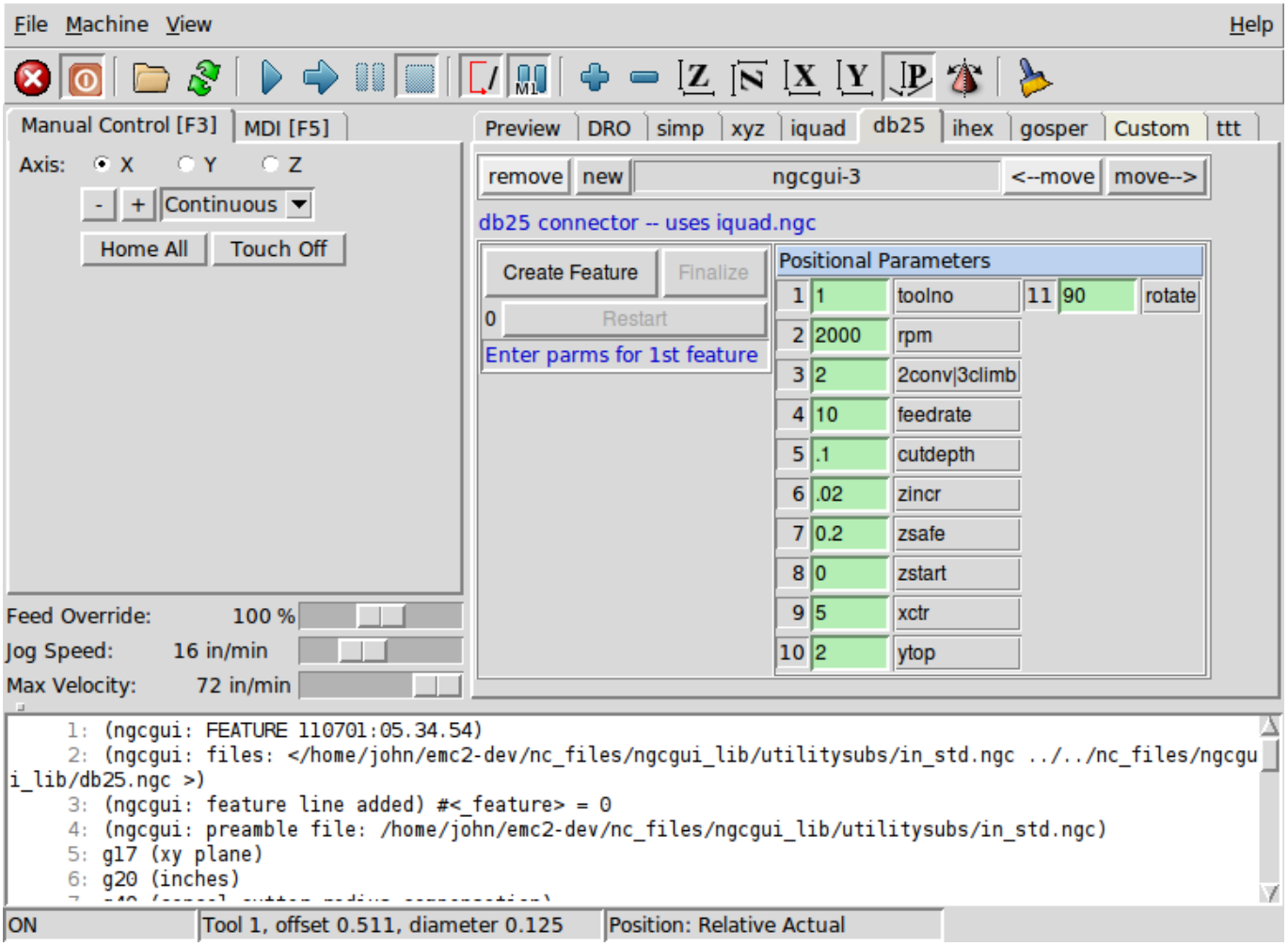
```
include(”ensure_mode.gcmc”)
```

and provide a proper path for gcmc include_files in the INI file, for example:

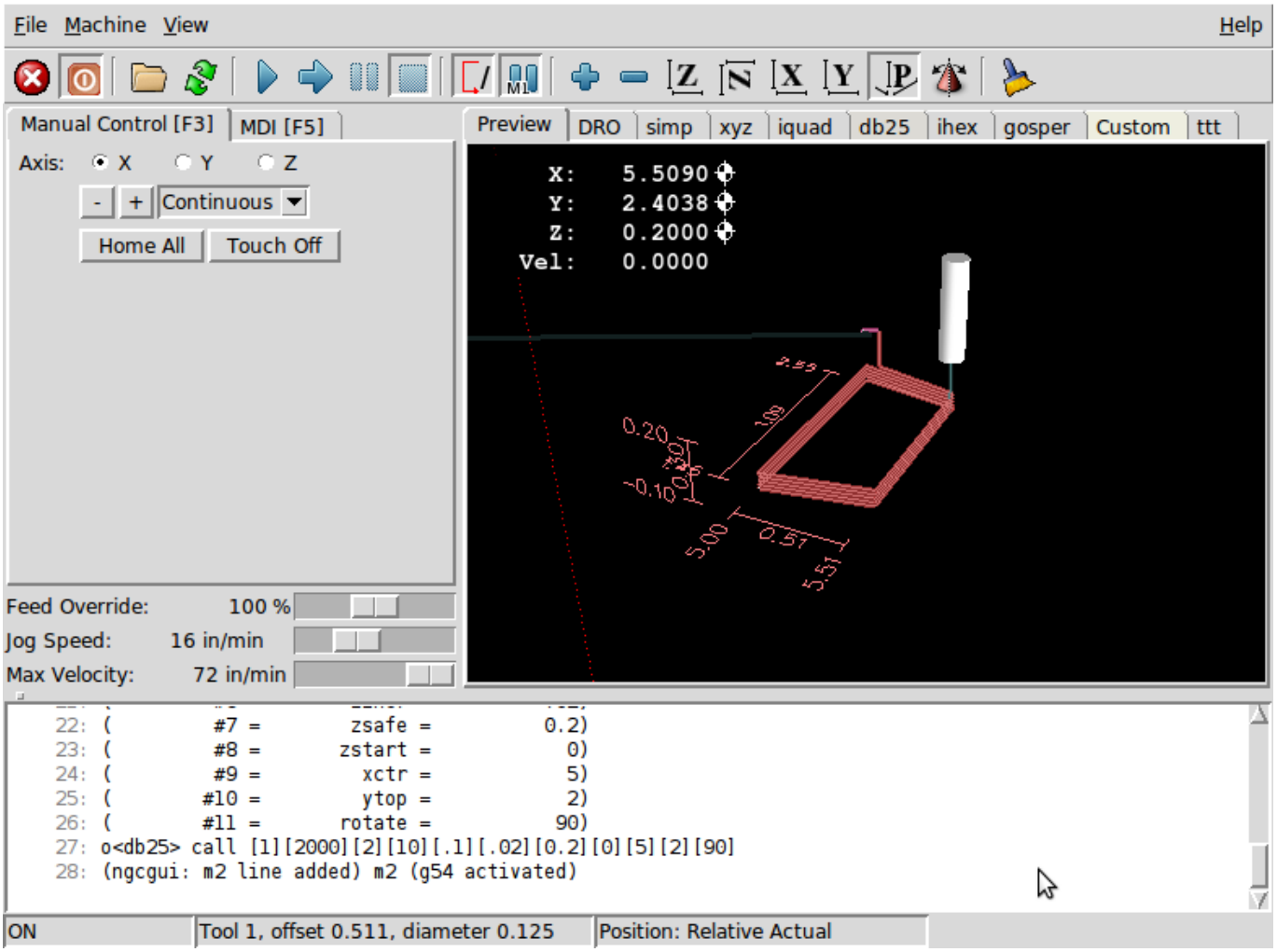
```
[DISPLAY]
GCMC_INCLUDE_PATH = ../../nc_files/gcmc_lib
```

10.6.7. DB25 Example

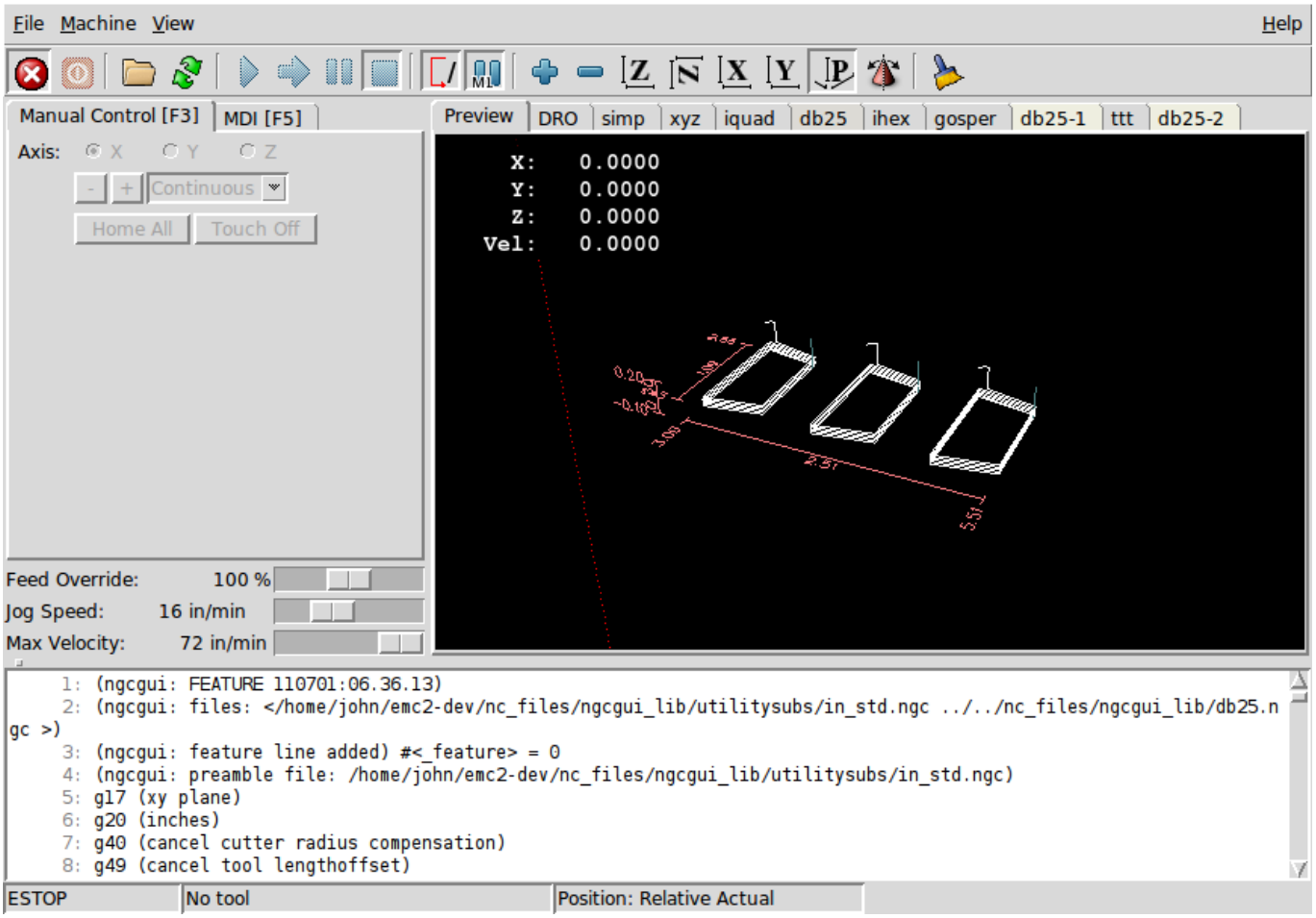
The following shows the DB25 subroutine. In the first photo you see where you fill in the blanks for each variable.



This photo shows the backplot of the DB25 subroutine.



This photo shows the use of the new button and the custom tab to create three DB25 cutouts in one program.



10.6.8. Creating a subroutine

- For creating a subroutine for use with NGCGUI, the filename and the subroutine name must be the same.
- The file must be placed in the subdirectory pointed to in the INI file.
- On the first line there may be a comment of type `info:`
- The subroutine must be surrounded by the `sub` and `endsub` tags.
- The variables used must be numbered variables and must not skip number.
- Comments and presets may be included.

Subroutine Skeleton Example

```
(info: simp -- simple exemple de sous-programme -- Ctrl-U pour éditer)
o<simp> sub
  #<ra>       = #1 (=.6 Rayon A) ;Example de paramètre avec un commentaire
  #<radius_b> = #2 (=0.4)         ;Example de paramètre sans commentaire
  #<feedrate> = #3 (Feedrate)     ;Example de paramètre sans preset
  g0x0y0z1
  g3 i#<ra> f#<feedrate>
  g3 i[0-#<radius_b>]
o<simp> endsub
```

10.7. GUI TkLinuxCNC

10.7.1. Introducción

TkLinuxCNC es una de las primeras interfaces gráficas para LinuxCNC. Esta escrita en Tcl y usa el toolkit Tk para visualización. Estar escrita en Tcl la hace muy portable (corre en una multitud de plataformas). Se puede visualizar una ventana separada de trazo posterior como se muestra.

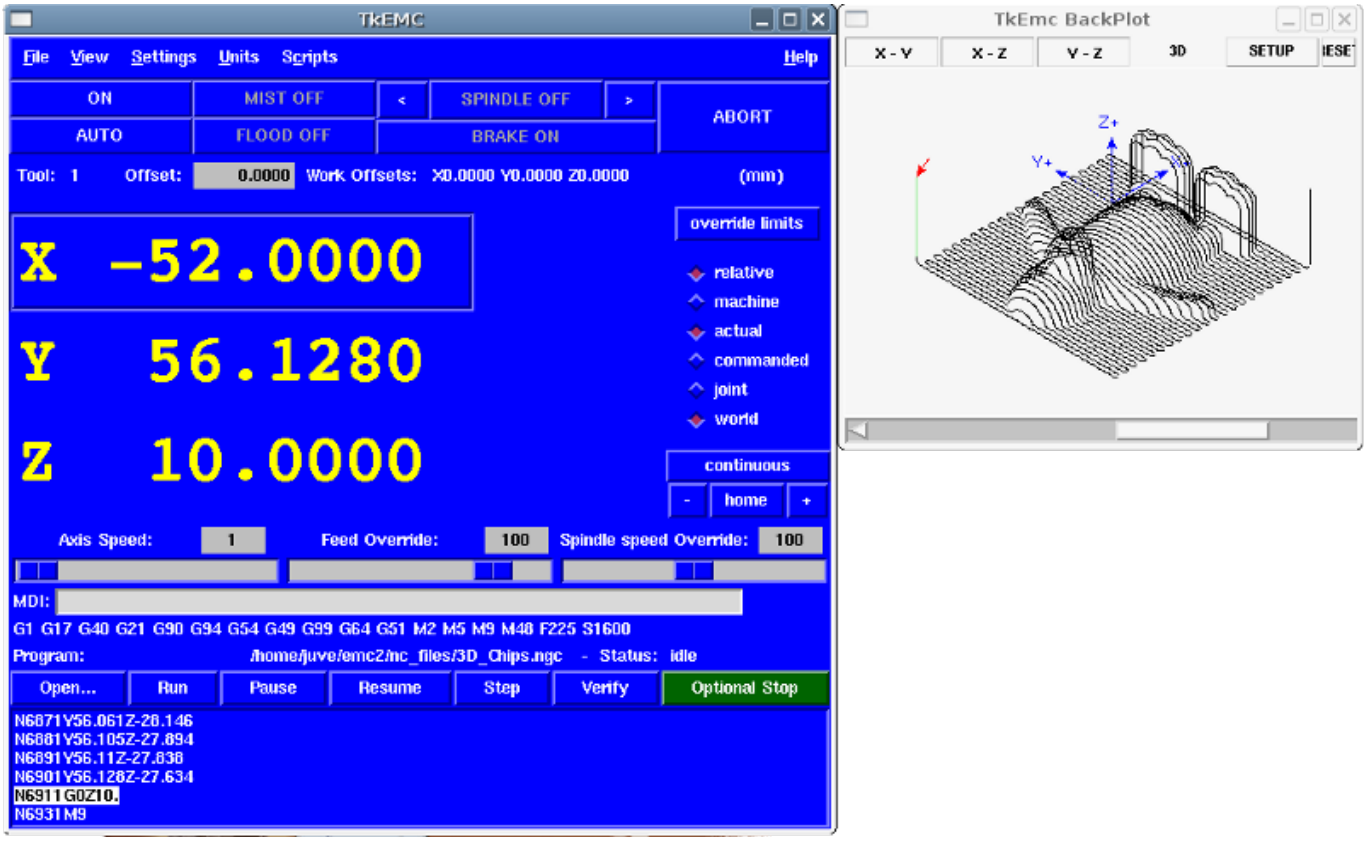


Figura 10.39: Ventana de TkLinuxCNC

10.7.2. Primeros pasos

Para seleccionar TkLinuxCNC como front-end de LinuxCNC edite el archivo INI. En la sección *[DISPLAY]* cambie la línea *DISPLAY* para leer

```
DISPLAY = tklinuxcnc
```

Luego, iniciar LinuxCNC y seleccionar ese archivo INI. La configuración de ejemplo *sim/tklinuxcnc.ini* ya está configurada para usar TkLinuxCNC como su interfaz de usuario.

Cuando se lance LinuxCNC se abrirá la <<cap:tklinuxcnc-display,ventana de TkLinuxCNC.

10.7.2.1. Una sesión típica con TkLinuxCNC

1. Iniciar LinuxCNC y seleccionar un archivo de configuración.

2. Limpiar el estado de *E-STOP* y encender la máquina (presionando F1 y luego F2).
3. Llevar a *casa* cada eje.
4. Cargar el archivo a ser fresado.
5. Colocar el material a fresar en la mesa.
6. Establecer los offsets apropiados para cada eje trotando y ya sea haciendo homing otra vez o haciendo clic derecho en el nombre de un eje y capturando un valor de offset. [1]
7. Ejecutar el programa.
8. Para fresar el mismo archivo, regresar al paso 6. Para fresar un archivo distinto regresar al paso 4. Al terminar, salir de LinuxCNC.

10.7.3. Elementos de la ventana de TkLinuxCNC

La ventana de TkLinuxCNC contiene los siguientes elementos:

- Una barra de menús que permite realizar varias acciones
- Un conjunto de botones que permiten cambiar el modo de trabajo actual, inicio/paro del husillo y otras E/S relevantes
- Barra de estado para varias visualizaciones relacionadas con offsets
- Área de visualización de coordenadas
- Un conjunto de deslizantes que controlan la *Velocidad de trote*, *Porcentaje de alimentación* y *Porcentaje de velocidad del husillo* los cuales permiten incrementar o decrementar estas configuraciones
- Caja de texto para entrada manual de datos *MDI*
- Barra de estado para mostrar códigos G y M activos y palabras F y S activas
- Botones relacionados con el intérprete
- Una área de visualización de texto que muestra el código G fuente del archivo cargado

10.7.3.1. Botones principales

De izquierda a derecha, los botones son:

- Activación de máquina: *ESTOP* > *RESTABLECER ESTOP* > *ON*
- Alternar refrigerante de niebla
- Decrementar velocidad del husillo
- Establecer dirección del husillo *HUSILLO OFF* > *HUSILLO EN DIRECTA* . *HUSILLO EN REVERSA*
- Incrementar la velocidad del husillo
- Abortar

después, en la segunda línea:

- Modo de operación: *MANUAL* > *MDI* > *AUTO*
- Alternador de inundación de refrigerante
- Alternador de control de freno del husillo

[1] Para algunas de estas acciones puede ser necesario cambiar el modo en el que se ejecuta LinuxCNC actualmente.

10.7.3.2. Barra de estado de offsets

La barra de estado de offsets muestra la herramienta actualmente seleccionada (con Txx M6), el offset de longitud de herramienta (si esta activo) y los offsets de trabajo (establecidos al hacer click derecho en las coordenadas).

10.7.3.3. Área de visualización de coordenadas

La parte principal de la pantalla muestra la posición actual de la herramienta. El color de la lectura de posición depende del estado del eje. Si el eje no está en casa el eje se muestra en letras amarillas. Una vez en casa se mostrará en letras verdes. Si hay un error con el eje actual, TkLinuxCNC usará letras rojas para indicarlo, por ejemplo si se ha alcanzado un interruptor de límite.

Para interpretar apropiadamente esos números, refiérase a los botones de opción de la derecha. Si la posición es *Máquina*, entonces el número mostrado esta en el sistema coordenado de la máquina. Si es *Relativo*, entonces el número mostrado esta en el desplazamiento del sistema coordenado. Opciones mas abajo pueden ser *Real* o *Comandada*. *Real* se refiere a la retroalimentación de los codificadores (si tiene una máquina servo), y *Comandada* se refiere a la posición ordenada enviada a los motores. Estos valores pueden diferir por razones distintas: error de seguimiento, banda muerta, resolución del codificador o tamaño de paso. Por ejemplo, si ordena un movimiento a X 0.0033 en su fresadora, pero un paso de su motor paso a paso es 0.00125, la posición *Comandada* será 0.0033, pero la posición *Real* será 0.0025 (2 pasos) o 0.00375 (3 pasos).

Otro grupo de botones de opción permite elegir entre las vistas *articulación* y *universal*. Estos no tienen mucho sentido en una máquina normal (p. ej. cinemáticas triviales), pero son de ayuda en máquinas con cinemáticas no-triviales como robots o plataformas Stewart. (puede leer más acerca de cinemáticas en el Manual del integrador).

Trazo posterior Cuando hay movimiento en la máquina, deja un rastro llamado trazo posterior. Se puede iniciar la ventana de trazo posterior seleccionando Ver→Backplot.

10.7.3.4. Intérprete TkLinuxCNC / Control de programa automático

Intérprete TkLinuxCNC / control de programa

Botones de control Los botones en la parte inferior de TkLinuxCNC se usan para controlar la ejecución de un programa:

+ * *Abrir* para cargar un programa, * *Verificar* para revisar si tiene errores, * *Ejecutar* para iniciar el corte real, * *Pausa* para detener la ejecución, * *Reanudar* para reanudar la ejecución de un programa pausado, * *Paso* para avanzar una línea en el programa y * *Parada opcional* para alternar el interruptor de parada opcional (si el botón esta verde se detendrá la ejecución del programa al encontrar cualquier M1).

Área de visualización de texto Cuando el programa está en ejecución se resalta en blanco la línea que se ejecuta actualmente. El área de texto se desplazará automáticamente para mostrar la línea actual.

10.7.3.5. Control manual

Teclas implícitas TkLinuxCNC permite mover manualmente la máquina. A esta acción se le conoce como *trotar*. Primero, seleccionar el eje a mover haciendo clic en él. Luego, hacer clic y mantenga presionado el botón + o - dependiendo de la dirección de movimiento deseada. Los primeros cuatro ejes también pueden moverse con el teclado, con teclas de flecha (X e Y), teclas Pág. Arriba y Pág. Abajo (Z) y las teclas *[* y *]* (A/4to).

+ Si se selecciona *Continuo*, el movimiento continuará mientras el botón o la tecla esta presionado(a). Si se selecciona otro valor, la máquina se moverá exactamente la distancia mostrada cada vez que se hace clic en el botón o se presiona la tecla. Los valores disponibles son:

+

```
1.0000, 0.1000, 0.0100, 0.0010, 0.0001
```

+ Al presionar *Home* o la tecla HOME, se llevará a casa el eje seleccionado. Dependiendo de la configuración, esto puede simplemente establecer el valor del eje a la posición absoluta 0.0, o puede hacer que la máquina se mueva a una ubicación de casa específica usando *interruptores de casa*. Ver el capítulo Homing para más información.

+ Al presionar *Ajuste de límites* se le permitirá temporalmente a la máquina trotar fuera de los límites definidos en el archivo INI. (Nota: Si el botón *Ajuste de límites* está activo se mostrará en color rojo).

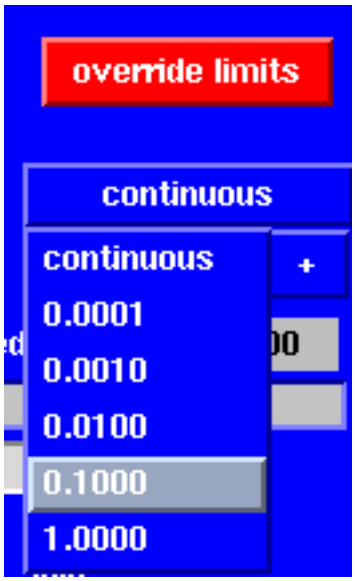


Figura 10.40: Ejemplo de ajuste de límites e incrementos de trote en TkLinuxCNC

El grupo Husillo Los botones de la primera fila seleccionan la dirección de giro del husillo: en sentido contrario a las agujas del reloj, detenido y en el sentido de las agujas del reloj. Los botones siguientes aumentan o disminuyen la velocidad de rotación. El botón en la segunda fila permite que el freno del husillo sea accionado o liberado. Dependiendo de la configuración de la máquina, no todos los elementos en este grupo pueden tener algún efecto.

El grupo Refrigerante Dos botones permiten encender y apagar los refrigerantes *Niebla* e *Inundación*. Dependiendo de la configuración de la máquina, no todos los elementos en este grupo pueden aparecer.

10.7.3.6. Entrada de código

Manual Data Input (también llamada MDI), permite ingresar programas en código G manualmente, una línea a la vez. Cuando la máquina no está encendida, y no esta en modo MDI, los controles de captura de código no están disponibles.



Esto permite ingresar un comando de código G para ser ejecutado. El comando se ejecuta al presionar Enter.

Códigos G activos Se muestran los *códigos modales* que están activos en el intérprete. Por ejemplo, *G54* indica que el *offset G54* se aplica a todas las coordenadas que se introduzcan.

10.7.3.7. Velocidad de trote

La velocidad de trote puede ser modificada al mover este deslizante. Los números de arriba se refieren a unidades del eje / segundo. Se puede hacer clic en la caja de texto con el número. Al hacer clic aparecerá una ventana emergente, donde se puede ingresar un número.

10.7.3.8. Porcentaje de alimentación

Al mover este deslizante, se puede modificar la velocidad de alimentación programada. Por ejemplo, si un programa solicita *F60* y el deslizante se establece en 120 %, entonces la velocidad de alimentación resultante será 72. Se puede hacer clic en la caja de texto con el número; al hacer clic aparecerá una ventana emergente donde se puede ingresar un número.

10.7.3.9. Porcentaje de velocidad del husillo

El deslizante de porcentaje de velocidad del husillo funciona exactamente como el deslizante de alimentación, pero controla la velocidad del husillo. Por ejemplo, si un programa solicita S500 (velocidad del husillo de 500 RPM) y el control deslizante se establece en 80 %, entonces la velocidad del husillo resultante será 400 RPM. El deslizante tienen valores mínimo y máximo definidos en el archivo INI; si faltasen, el deslizante permanecerá en 100 %. Se puede hacer clic en el cuadro de texto con el número, al hacerlo aparecerá una ventana emergente que permitirá capturar un número.

10.7.4. Controles del teclado

Casi todas las acciones en TkLinuxCNC se pueden realizar con el teclado. Muchos de los atajos de teclado no están disponibles cuando se está en modo MDI.

Los métodos abreviados de teclado más utilizados se muestran en la siguiente tabla.

Cuadro 10.7: Atajos de teclado más comunes

Teclas	Acción
F1	Alterna parada de emergencia
F2	Enciende/apaga la máquina
`, 1 .. 9, 0	Ajusta el porcentaje de alimentación de 0 % a 100 %
X, `	Activa el primer eje
Y, 1	Activa el segundo eje
Z, 2	Activa el tercer eje
A, 3	Activa el cuarto eje
Casa	Envía el eje activo a casa
Izquierda, Derecha	Trota el primer eje
Arriba, Abajo	Trota el segundo eje
Pág. arriba, Pág. abajo	Trota el tercer eje
[,]	Trota el cuarto eje

Cuadro 10.7: (continued)

Teclas	Acción
ESC	Detiene la ejecución

10.8. QtPlasmaC

10.8.1. Preamble

Except where noted, this guide assumes the user is using the latest version of QtPlasmaC. Version history can be seen by visiting this link which will show the latest available version. The installed QtPlasmaC version is displayed in the title bar. See Update QtPlasmaC for information on updating QtPlasmaC.

10.8.2. License

QtPlasmaC and all of its related software are released under GPLv2.

10.8.3. Introducción

The development branch version of QtPlasmaC is a GUI for plasma cutting which utilises the plasmac component for controlling a plasma table using the master branch (development) version of LinuxCNC (v2.10) using the Debian Bullseye or later distribution.

The QtPlasmaC GUI supports up to five axes and uses the QtVCP infrastructure provided with LinuxCNC.

The standard theme is based on a design by user ”pinder” on the LinuxCNC Forum and the colors are able to be changed by the user.

The development branch version of the QtPlasmaC GUI will run on any hardware that is supported by the master branch version of LinuxCNC (v2.10) provided there are enough hardware I/O pins to fulfill the requirements of a plasma configuration.

There are three available formats:

- 16:9 with a minimum resolution of 1366 x 768
- 9:16 with a minimum resolution of 768 x 1366
- 4:3 with a minimum resolution of 1024 x 768

Screenshot examples of QtPlasmaC are below:

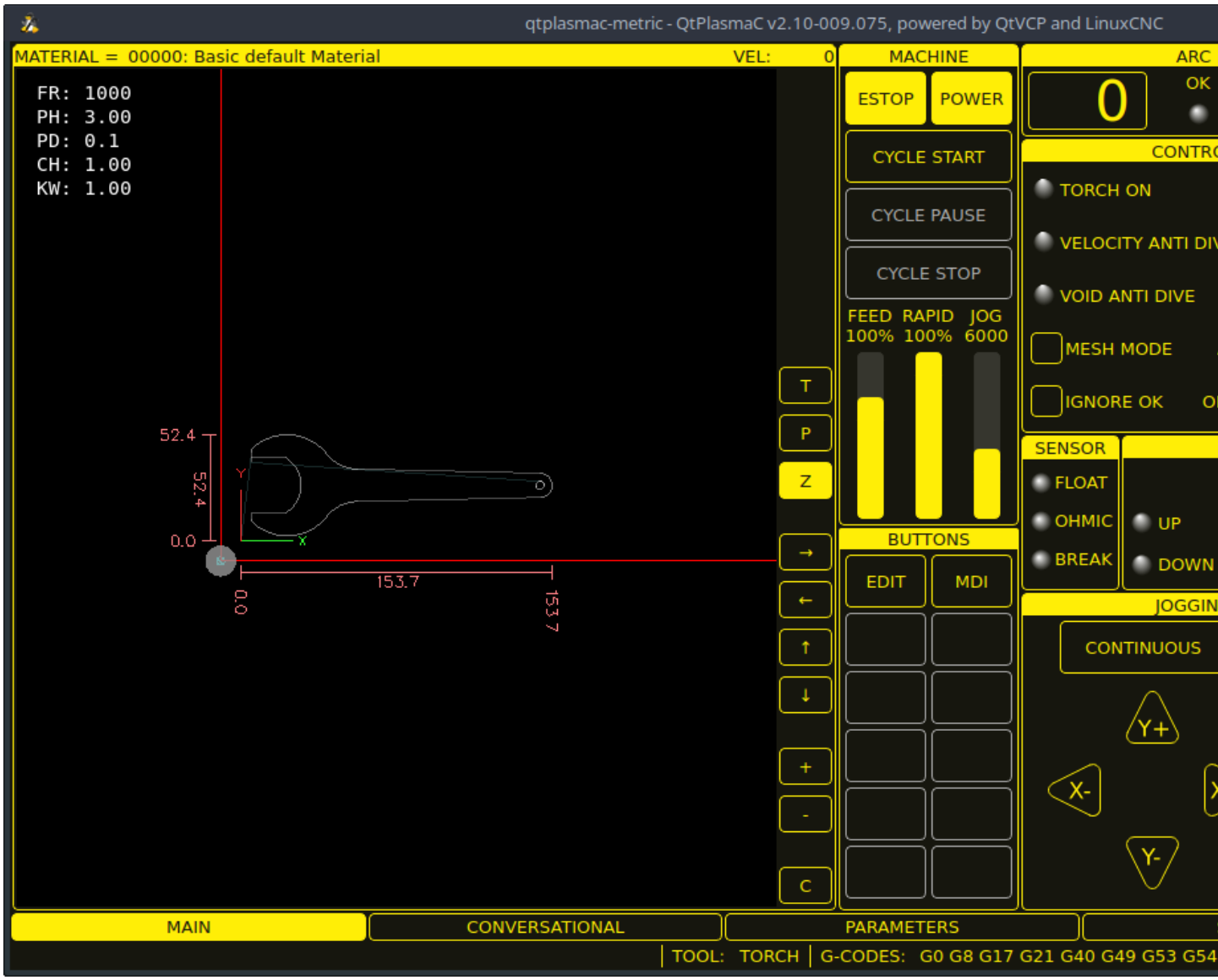
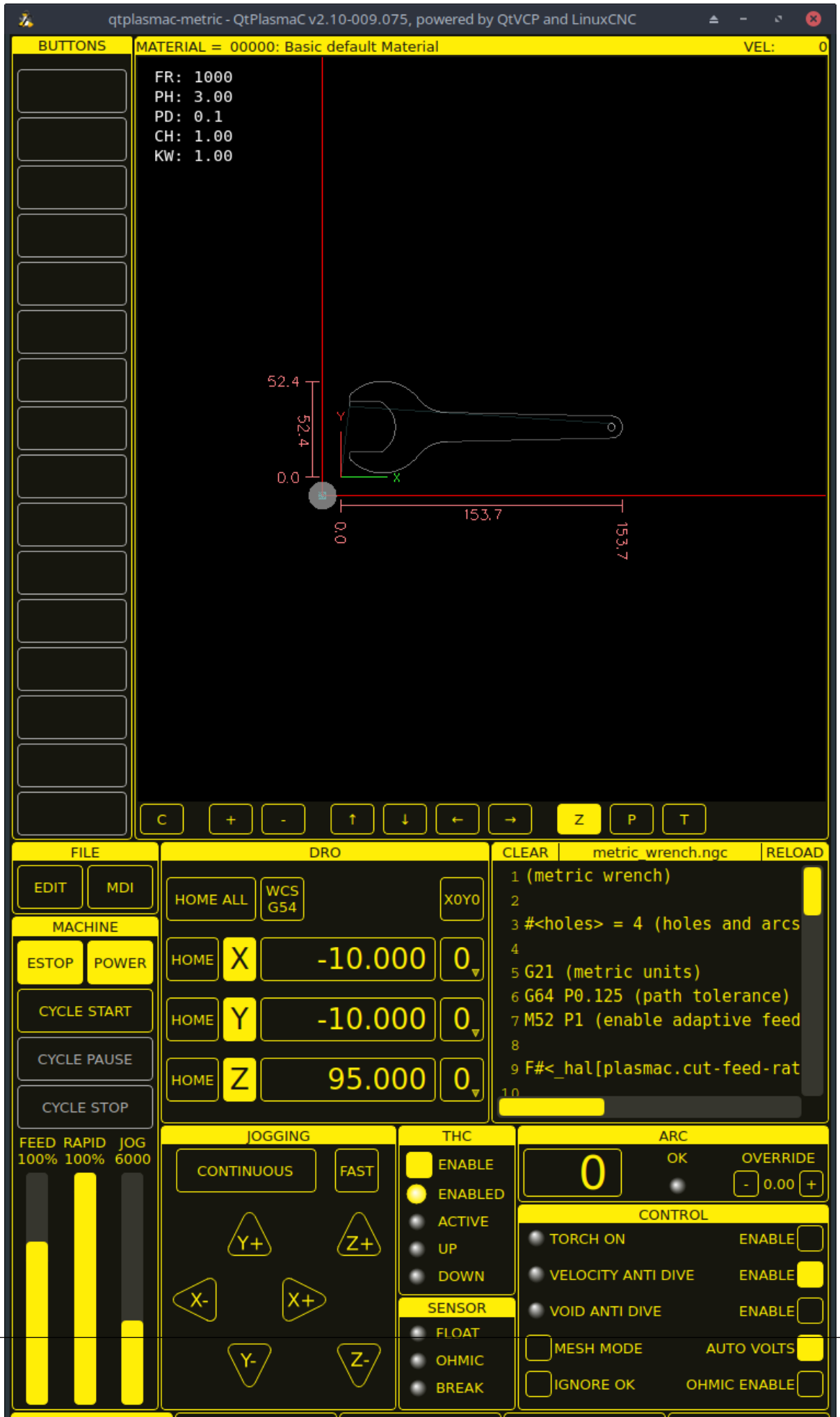


Figura 10.41: **16:9**


qtplasmac-metric - QtPlasmaC v2.10-009.075, powered by QtVCP and LinuxCNC
BUTTONS
MATERIAL = 00000: Basic default Material
VEL: 0
FR: 1000
PH: 3.00
PD: 0.1
CH: 1.00
KW: 1.00
52.4
52.4
0.0
153.7
0.0
153.7
C
+
-
Z
P
T
FILE
EDIT
MDI
MACHINE
ESTOP
POWER
CYCLE START
CYCLE PAUSE
CYCLE STOP
FEED 100%
RAPID 100%
JOG 6000
DRO
HOME ALL
WCS G54
X0Y0
HOME X -10.000 0
HOME Y -10.000 0
HOME Z 95.000 0
CLEAR
metric_wrench.ngc
RELOAD
1 (metric wrench)
3 #<holes> = 4 (holes and arcs
5 G21 (metric units)
6 G64 P0.125 (path tolerance)
7 M52 P1 (enable adaptive feed
9 F#<_hal[plasmac.cut-feed-rat
JOGGING
CONTINUOUS
FAST
Y+
Z+
X-
X+
Y-
Z-
THC
ENABLE
ENABLED
ACTIVE
UP
DOWN
SENSOR
FLOAT
OHMIC
BREAK
ARC
0
OK
OVERRIDE
0.00
CONTROL
TORCH ON
ENABLE
VELOCITY ANTI DIVE
ENABLE
VOID ANTI DIVE
ENABLE
MESH MODE
AUTO VOLTS
IGNORE OK
OHMIC ENABLE

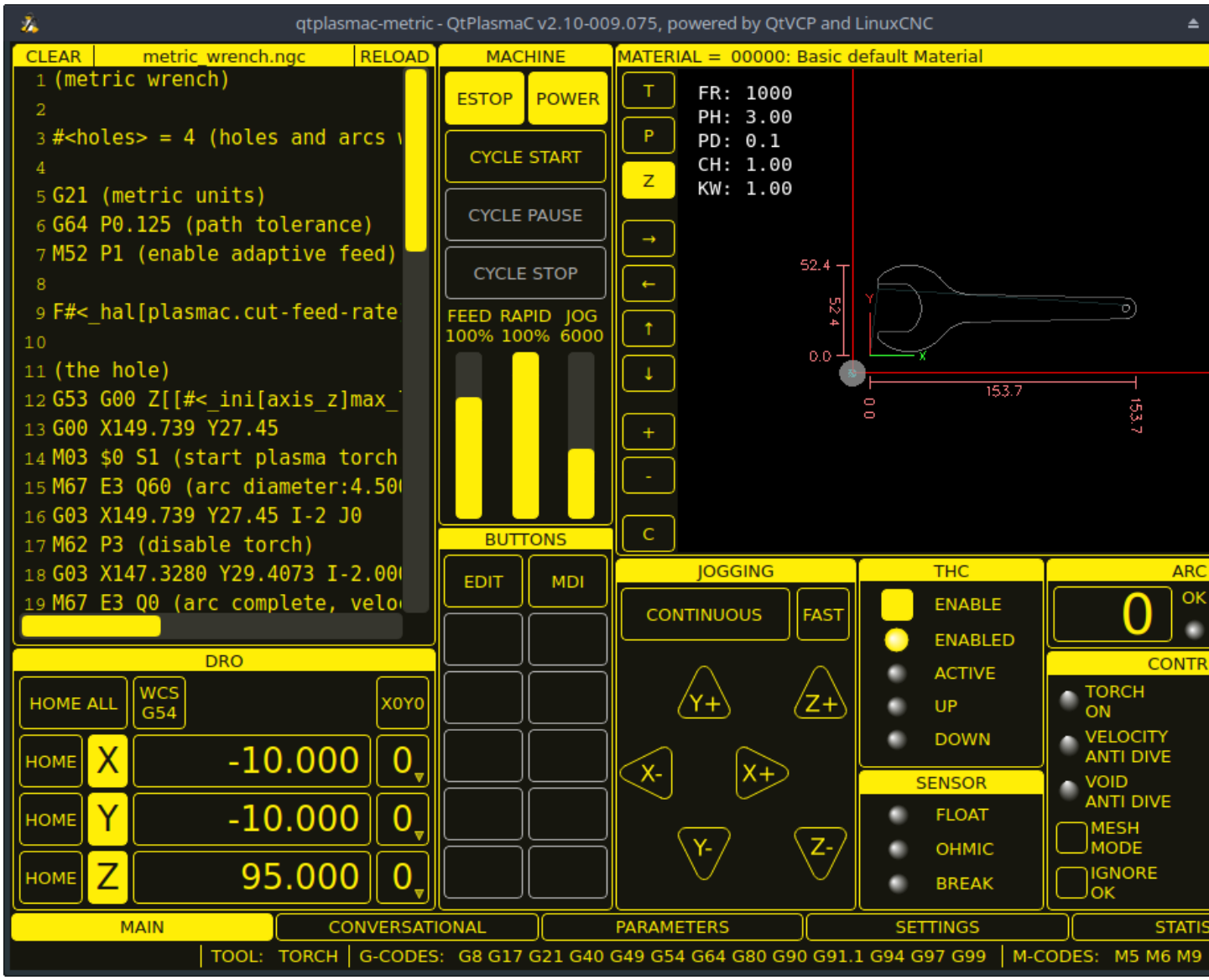


Figura 10.43: **4:3**

10.8.4. Installing LinuxCNC

The preferred method for installing LinuxCNC is via an ISO image as described below.

---

nota
It is possible to install and run LinuxCNC on a variety of Linux distributions however that is beyond the scope of this User Guide. If the user wishes to install a Linux distribution other than those recommended, they will first need to install their preferred Linux distribution and then install the master branch version of LinuxCNC (v2.10) along with any required dependencies. It should also be noted that Bullseye is the earliest Debian distribution that is supported by the master branch of LinuxCNC (v2.10). Buster is no longer supported.

10.8.4.1. If The User Does Not Have Linux Installed

Installation instructions are available from here.

Following these instructions will yield a machine with the current stable branch of LinuxCNC (v2.9) on Debian 12 (Bookworm). The user will then have to follow the appropriate instructions to upgrade to the master branch version of LinuxCNC (v2.10).

10.8.4.2. Package Installation (Buildbot) If The User Has Linux on Debian 12 (Bookworm)

Follow the instructions from the Updating LinuxCNC on Debian Bookworm section from here.

10.8.4.3. Package Installation (Buildbot) If The User Has Linux on Debian 12 (Bookworm) or Debian 11 (Bullseye)

A package installation (Buildbot) uses prebuilt packages from the LinuxCNC Buildbot.

Add the GPG keys and add the repository to the sources list to suit the Debian version.

The below stanza would add the master branch (v2.10) Bookworm repository.

```
deb     http://buildbot2.highlab.com/ bookworm master-uspace
```

10.8.4.4. Run In Place Installation If The User Has Linux Installed

A run in place installation runs LinuxCNC from a locally compiled version usually located at ~/linuxcnc-dev, instructions for building a run in place installation are available from here.

10.8.5. Creating A QtPlasmaC Configuration

Prior to creating a QtPlasmaC configuration, it is important that the user has a firm understanding of the operating modes available, as well as the I/O's that are required for successful plasma operation.

10.8.5.1. Modes

QtPlasmaC requires the selection of one of following three operating modes:

Modo	Descripción
0	Uses an external arc voltage input to calculate both Arc Voltage (for Torch Height Control) and Arc OK.
1	Uses an external arc voltage input to calculate Arc Voltage (for Torch Height Control). Uses an external Arc OK input for Arc OK.
2	Uses an external Arc OK input for Arc OK. Use external up/down signals for Torch Height Control.



importante
If the plasma power source has an Arc OK (Transfer) output then it is recommended to use that for Arc OK rather than the soft (calculated) Arc OK provided by mode 0. It may also be possible to use a reed relay as an alternative method to establish an Arc OK signal when the power source does not provide one.

nota
For fine tuning of Mode 0 Ark OK see Tuning Mode 0 Arc OK in the Advanced Topics section of the manual.

10.8.5.2. Available I/Os

nota
This section only touches on the hardware I/O's required for QtPlasmaC. Base machine requirements such as limit switches, home switches, etc. are in addition to these.

Name	Modes	Descripción
Arc Voltage	0, 1	Analog input; **optional.** HAL pin name `plasmac.arc-voltage-in` Connected to the velocity output of an encoder equipped breakout board. This signal is used to read the arc voltage to determine the necessary corrections to maintain the torch distance from the work piece during cutting.
Arc OK	1, 2	Digital input; **optional.** HAL pin name `plasmac.arc-ok-in` Connected from the Arc OK output of the plasma power source to an input on the breakout board. This signal is used to determine if the cutting arc has been established and it is ok for the machine to move (sometimes called arc transfer).
Float Switch	0, 1, 2	Digital input; **optional, see info below table:** HAL pin name `plasmac.float-switch` Connected from a breakout board input to a switch on the floating head. This signal is used to mechanically probe the work piece with the torch and set Z zero at the top of the work piece. If used and no ohmic probe is configured, this is the probing method. If used and an ohmic probe is configured, this is the fallback probing method.
Ohmic Probe	0, 1, 2	Digital input; **optional, see info below table:** HAL pin name `plasmac.ohmic-probe` Connected from to the ohmic probe's output to a breakout board input. This signal is used to probe electronically by completing a circuit using the work piece and the torch consumables and set Z zero at the top of the work piece. If used, this is the primary probing method. If an ohmic probe fails to locate the work piece, and there is no float switch is present, probing will continue until the torch breaks away or the minimum Z limit is reached.
Ohmic Probe Enable	0, 1, 2	Digital output; **optional, see info below table:** HAL pin name `plasmac.ohmic-enable` Connected from a breakout board output to an input to control the ohmic probe's power.
Breakaway Switch	0, 1, 2	Digital input; **optional, see info below table:** HAL pin name `plasmac.breakaway` Connected from a breakout board input to a torch breakaway detection switch. This signal senses if the torch has broken away from its cradle.

Name	Modes	Descripción
Torch On	0, 1, 2	Digital output; **required.** HAL pin name `plasmac.torch-on` Connected from a breakout board output to the torch-on input of the plasma power supply. This signal is used to control the plasma power supply and start the arc.
Move Up	2	Digital input; **optional.** HAL pin name `plasmac.move-up` Connected from the up output of the external THC control to a break out board input. This signal is used to control the Z axis in an upward motion and make necessary corrections to maintain the torch distance from the work piece during cutting.
Move Down	2	Digital input; **optional.** HAL pin name `plasmac.move-down` Connected from the down output of the external THC control to a break out board input. This signal is used to control the Z axis in a downward motion and make necessary corrections to maintain the torch distance from the work piece during cutting.
Scribe Arming	0, 1, 2	Digital output; **optional.** HAL pin name `plasmac.scribe-arm` Connected from a breakout board output to the scribe arming circuit. This signal is used to place the scribe into position on the work piece .
Scribe On	0, 1, 2	Digital output; **optional.** HAL pin name `plasmac.scribe-on` Connected from a breakout board output to the scribe-on circuit. This signal is used to turn the scribing device on.
Laser On	0, 1, 2	Digital output; **optional.** HAL pin name `qtplasmac.laser_on` This signal is used to turn the alignment laser on.

Only one of either **Float Switch** or **Ohmic Probe** is required. If both are used, then **Float Switch** will be a fallback if **Ohmic Probe** is not sensed.

If **Ohmic Probe** is used, then **Ohmic Probe Enable** is required to be checked on the QtPlasmaC GUI.

Breakaway Switch is not mandatory because the **Float Switch** is treated the same as a breakaway when not probing. If they are two separate switches, and there are not enough inputs on the breakout board, they could be combined and connected as a **Float Switch**.

---

nota
The minimum I/O requirement for a QtPlasmaC configuration to function are: **Arc Voltage** input OR **Arc OK** input, **Float Switch** input, and **Torch On** output. To reiterate, in this case QtPlasmaC will treat the float switch as a breakaway switch when it is not probing.

10.8.5.3. Configuración recomendada:

Refer to the Heights Diagrams for a visual representation of the terms below.

- **[AXIS_Z] MIN_LIMIT** should be just below top of the slats with allowances for float_switch_travel and over travel tolerance. For example, if the user's float switch takes 4 mm (0.157") to activate then set the Z minimum to 5 mm (0.2") plus an allowance for overrun (either calculated using the equation below or allow 5 mm (0.2") below the lowest slat).

- **[AXIS_Z] MAX_LIMIT** should be the highest the user wants the Z axis to travel (it must not be lower than Z HOME_OFFSET).
- **[AXIS_Z] HOME** should be set to be approximately 5 mm-10 mm (0.2”-0.4”) below the maximum limit.
- **Floating Head** - it is recommended that a floating head be used and that it has enough movement to allow for overrun during probing. Overrun can be calculated using the following formula:

```
o = 0.5 * a * (v / a)^2
```

where: o = overrun, a = acceleration in units/s^2 and v = velocity in units/s.

Metric example: given a Z axis MAX_ACCELERATION of 600 mm/s^2 and MAX_VELOCITY of 60 mm/s, the overrun would be 3 mm.

Imperial example: given a Z axis MAX_ACCELERATION of 24 in/s^2 and MAX_VELOCITY of 2.4 in/s, the overrun would be 0.12 in.

On machines that will utilize an ohmic probe as the primary method of probing, it is highly recommended to install a switch on the floating head as a backup means of stopping Z motion in the event of ohmic probe failure due to dirty surfaces.

10.8.5.4. Configuring

LinuxCNC provides two configuration wizards which can be used to build a machine configuration. The choice of these wizards is dependent on the hardware used to control the machine.

If the user wishes to use a Run In Place installation then prior to running one of the following commands they will need to run the following command from a terminal:

```
source ~/linuxcnc-dev/scripts/rip-environment
```

If using a Package installation, then no additional action is required.

Si usa un puerto paralelo, use el Asistente StepConf ejecutando el comando `stepconf` en una ventana de terminal a lanzándola usando la entrada de menú del escritorio **Application -> CNC -> StepConf Wizard**.

Si usa un tablero Mesa Electronics, use el Asistente PnCconf ejecutando el comando `pncconf en una ventana de terminal o lanzándolo usando la entrada de menú de escritorio **Application -> CNC -> PnCConf**.

If using a Pico Systems board, this LinuxCNC forum thread may be helpful.

The machine specific settings are not described here, refer to the documentation for the particular configuration wizard that is being used.

There are LinuxCNC forum sections available for these wizards:

StepConf Wizard

PnCconf Wizard

Fill in the required entries to suit the machine wiring/breakout board configuration.

QtPlasmaC adds two pages to the LinuxCNC configuration wizards for QtPlasmaC specific parameters, the two pages are QtPlasmaC options and User Buttons. Complete each of the wizards QtPlasmaC page to suit the machine that is being configured and the user button requirements.

Note that PnCConf options allow user selection of Feed Override, Linear Velocity, and Jog Increments, whereas in StepConf these are automatically calculated and set.

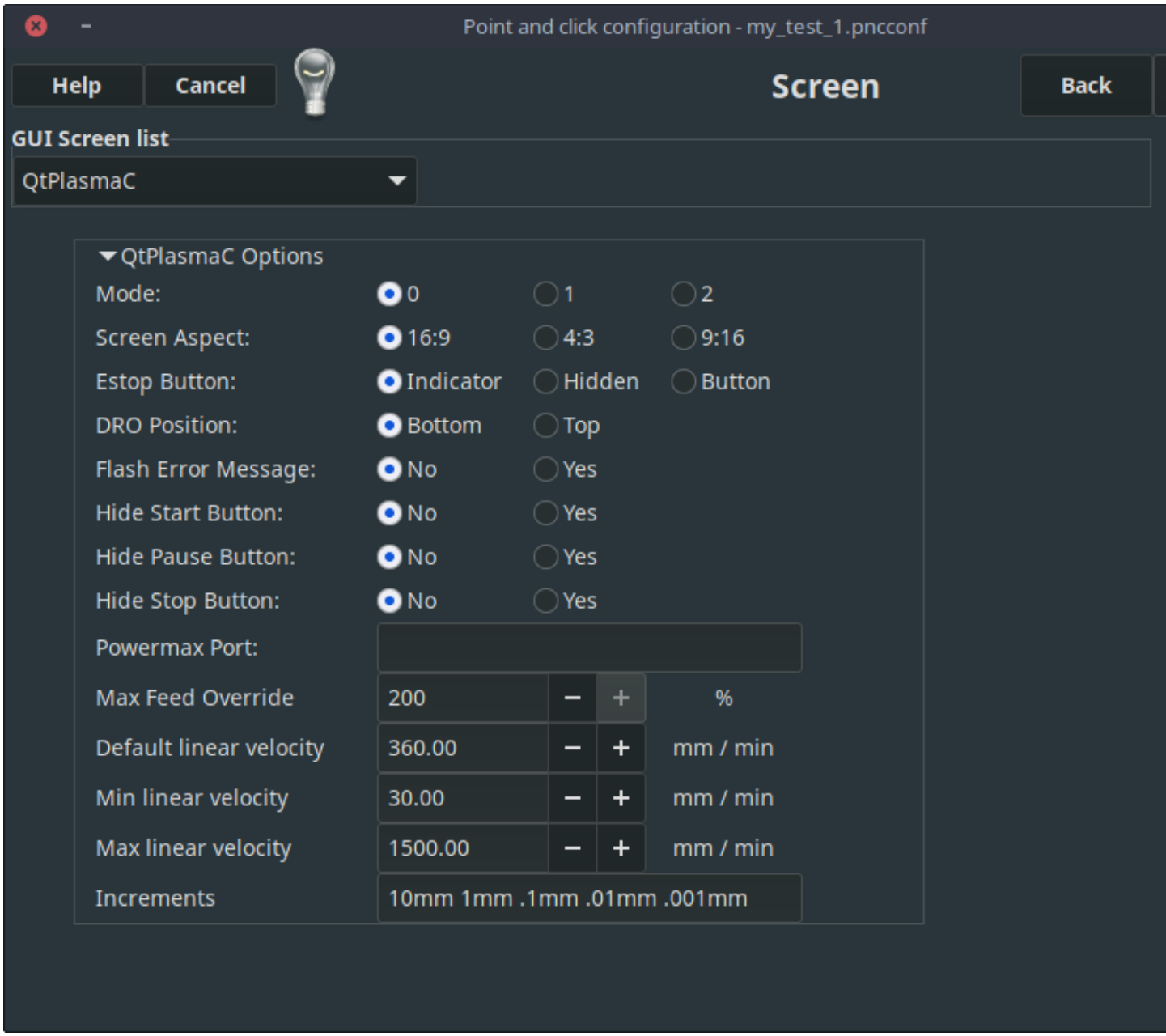


Figura 10.44: PnCConf QtPlasmaC Options

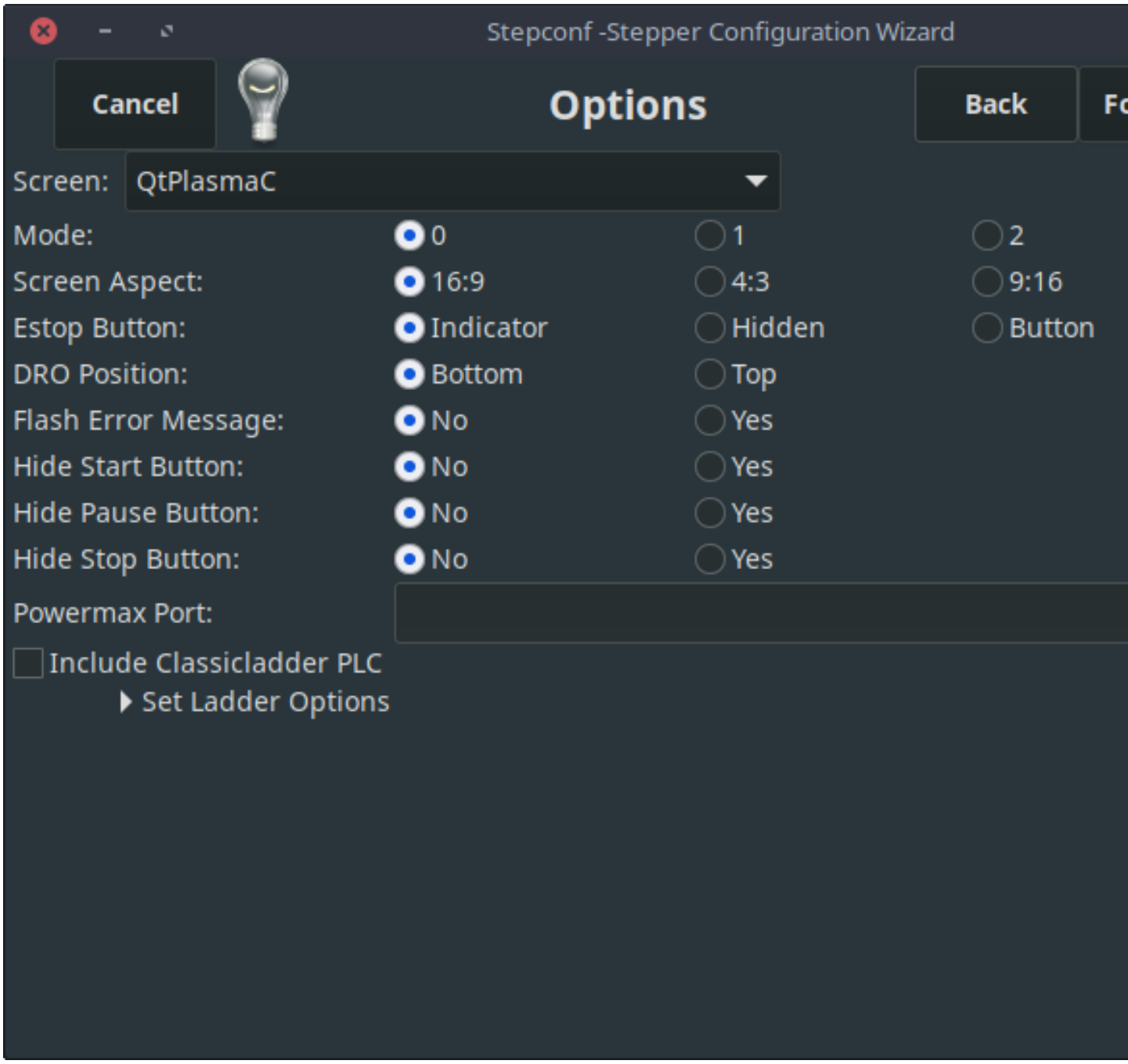


Figura 10.45: StepConf QtPlasmaC Options

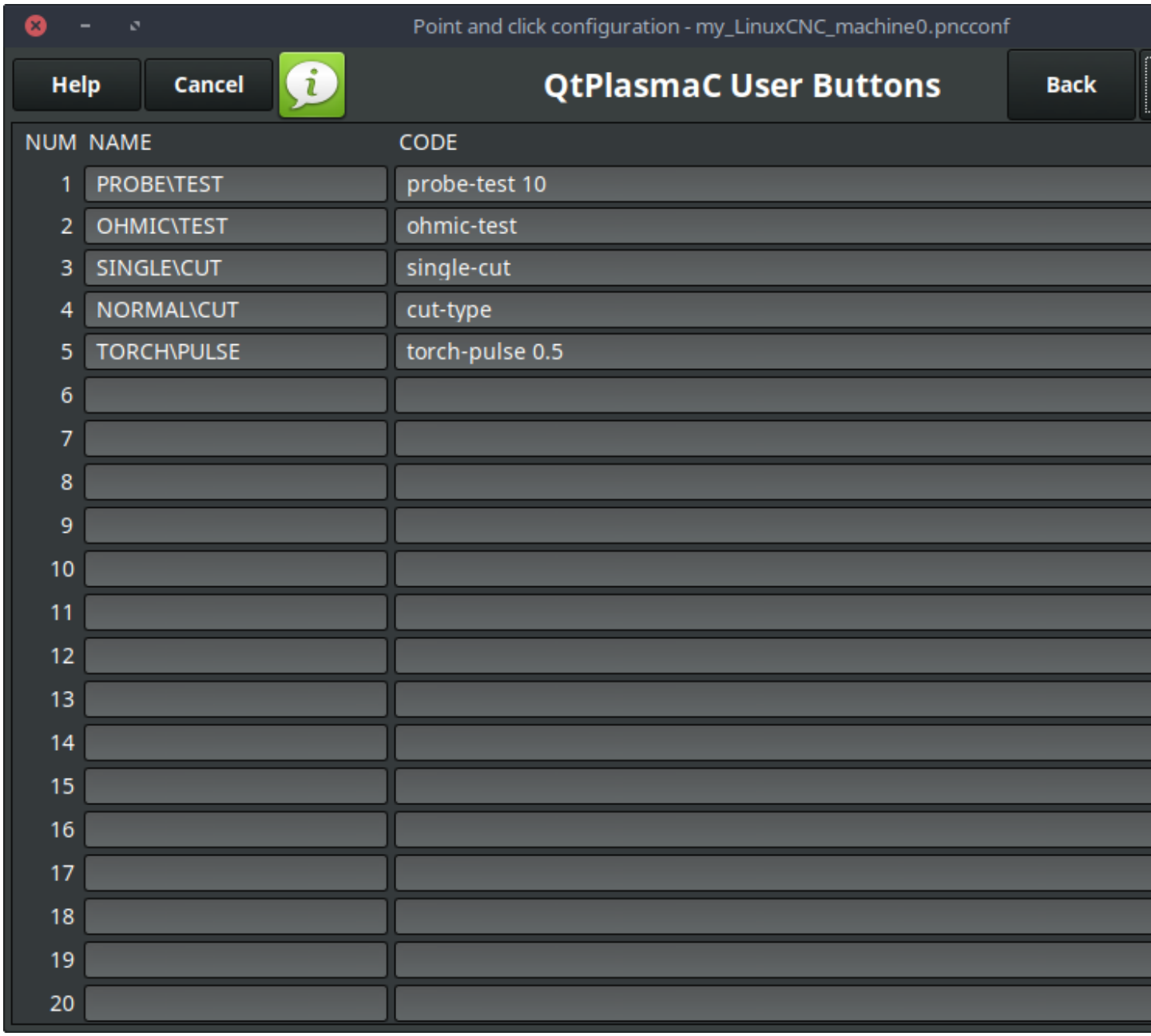


Figura 10.46: QtPlasmaC User Buttons

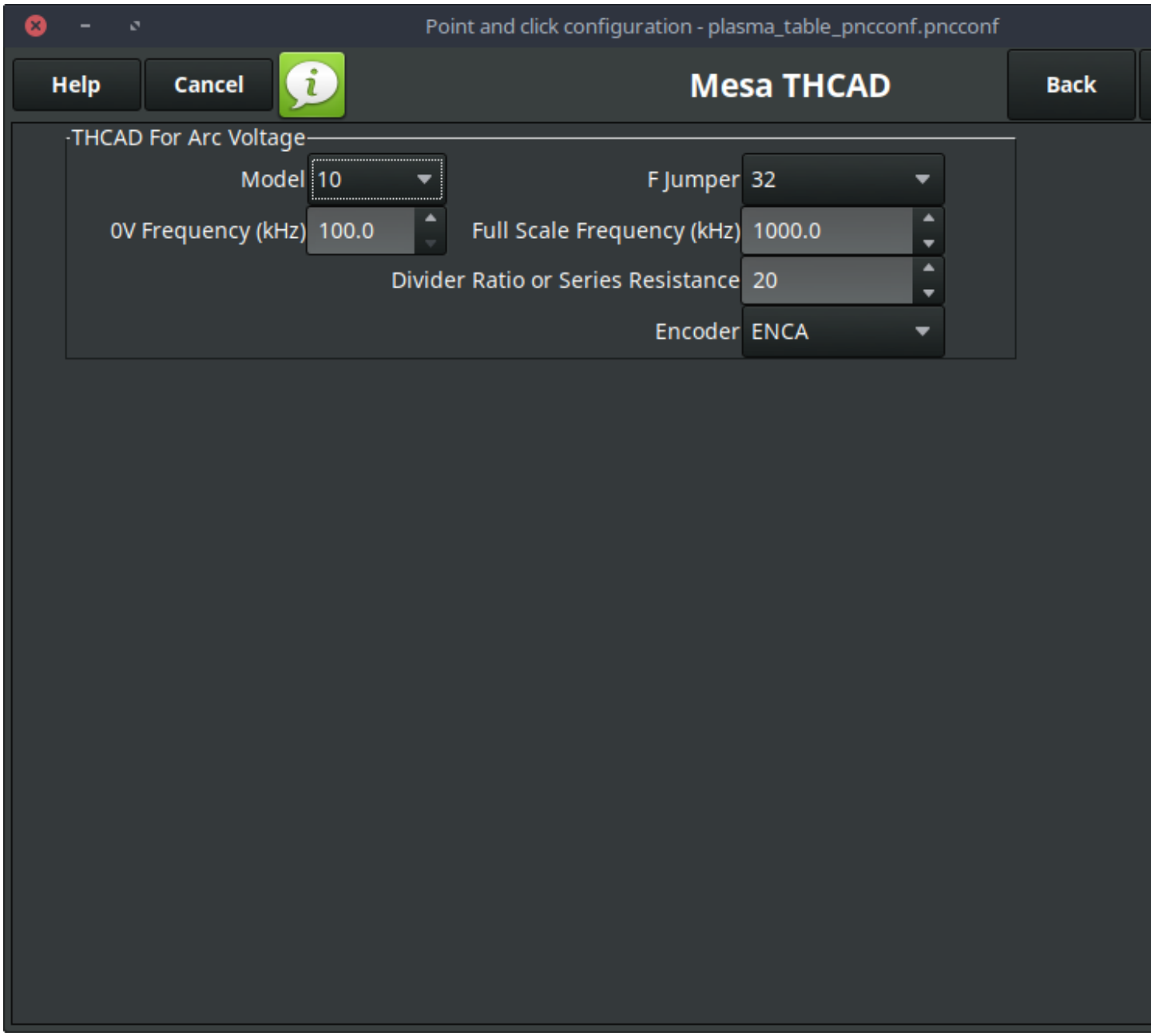


Figura 10.47: QtPlasmaC THCAD

The THCAD screen will only appear if a Plasma Encoder is selected in the card screen. The the dedicated section on Mesa THCAD for more information.

When the configuration is complete, the wizard will save a copy of the configuration that may be loaded and edited at a later time, a working QtPlasmaC configuration will be created in the following directory: ~/linuxcnc/configs/<*machine_name*>.

The way the newly created QtPlasmaC configuration can be run from the terminal command line slightly differs depending on the way LinuxCNC was installed:

For a package installation (Buildbot):

```
linuxcnc ~/linuxcnc/configs/_<machine_name>_/_<machine_name>_.ini
```

For a run in place installation:

```
~/linuxcnc-dev/scripts/linuxcnc ~/linuxcnc/configs/_<machine_name>_/_<machine_name>_.ini
```

After running the above command LinuxCNC should be running with the QtPlasmaC GUI visible.



importante
BEFORE PROCEEDING, THE USER SHOULD BE ABLE TO HOME THE MACHINE, ZERO EACH AXIS, JOG ALL AXES TO SOFT LIMITS WITHOUT CRASHING, AND RUN TEST G-CODE PROGRAMS WITHOUT ANY ERRORS.

ONLY WHEN this criteria is met should the user proceed with the QtPlasmaC initial setup.

nota
It is possible to create a sim configuration using StepConf but it is not possible to have tandem joints in the sim configuration.

10.8.5.5. Qt Dependency Errors

If any Qt dependency errors are encountered while attempting to run the QtPlasmaC configuration, the user may need to run the QtVCP installation script to resolve these issues.

For a package installation (Buildbot) enter the following command in a terminal window:

```
/usr/lib/python3/dist-packages/qtvcp/designer/install_script
```

For a run in place installation enter the following command in a terminal window:

```
~/linuxcnc-dev/lib/python/qtvcp/designer/install_script
```

10.8.5.6. Initial Setup

The following heights diagrams will help the user visualize the different heights involved in plasma cutting and how they are measured. There are two different scenarios based on if the user chooses to use **Probe Height** only, or if the user chooses to use **Slat Height** AND **Material Thickness**.

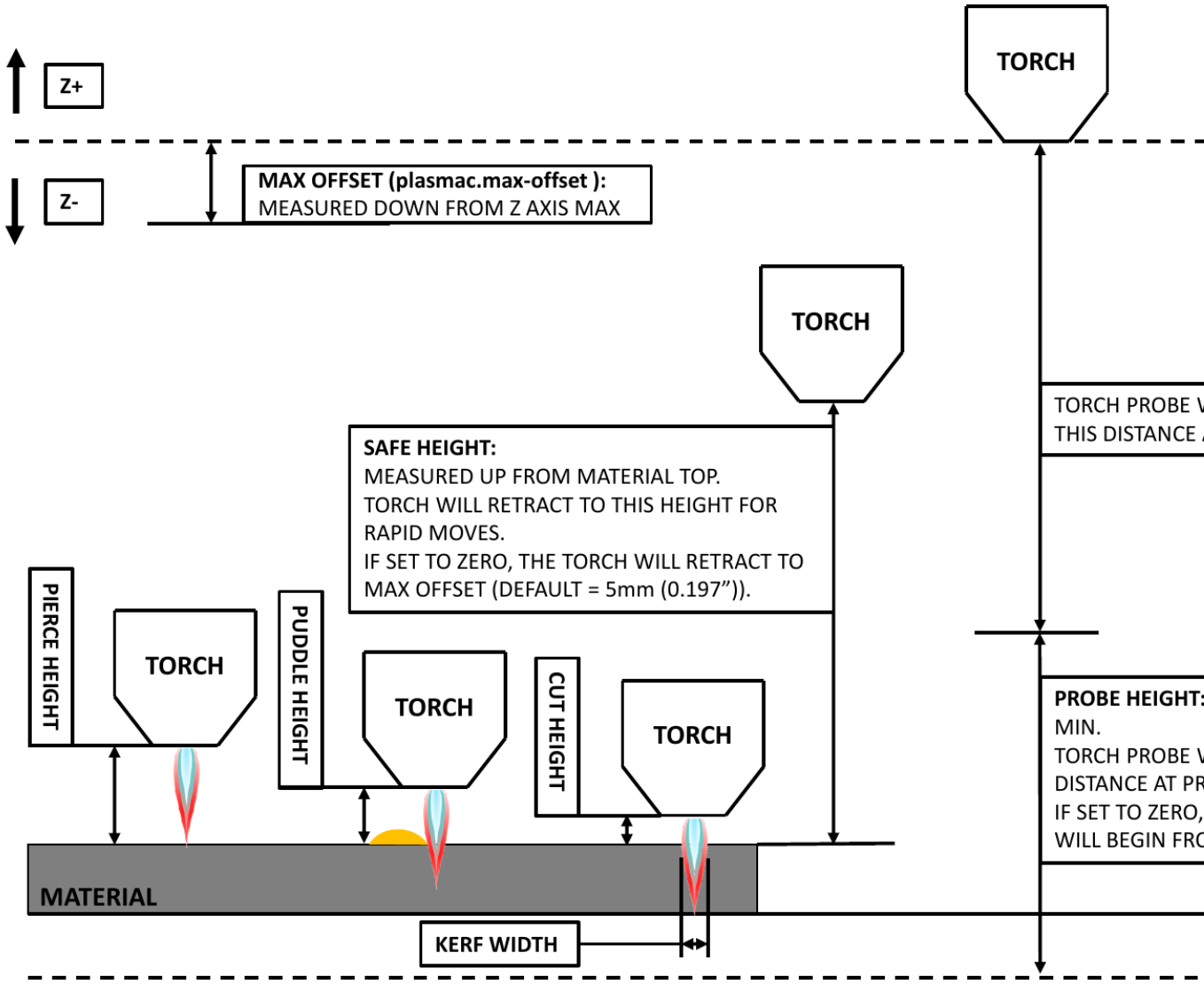


Figura 10.48: **Probe Height Only**

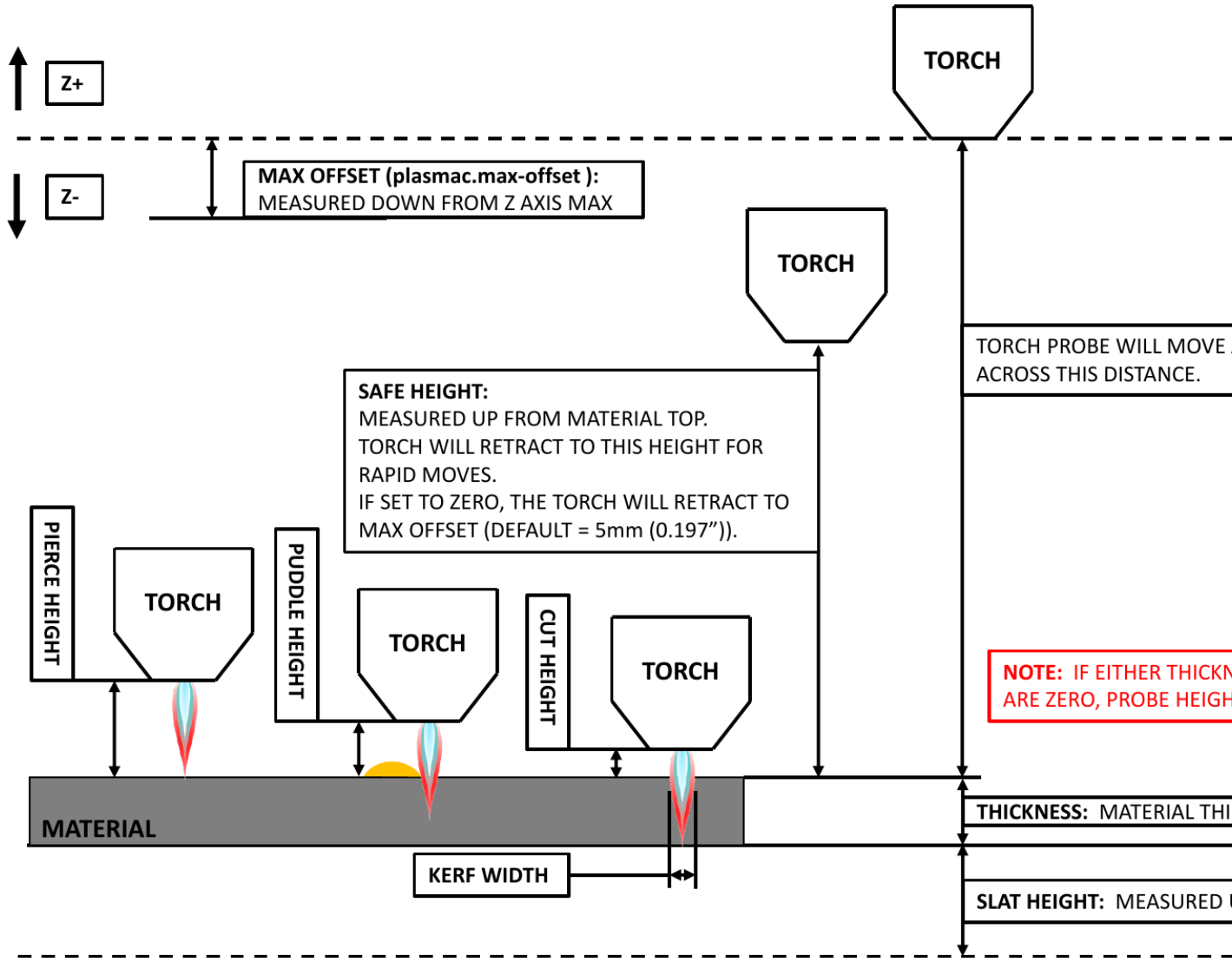


Figura 10.49: **Slat Height and Material Thickness**

Click on the Parameters Tab to view the **CONFIGURATION** section which shows the user settable parameters. It is necessary to ensure every one of these settings is tailored to the machine.

To set the Z axis DRO relative to the Z axis MINIMUM_LIMIT, the user should perform the following steps. It is important to understand that in QtPlasmaC, touching off the Z axis DRO has no effect on the Z axis position while running a G-code program. These steps simply allow the user to more easily set the probe height as after performing the steps, the displayed Z axis DRO value will be relative to Z axis MINIMUM_LIMIT.

---

nota
The user should be familiar with the recommended Z Axis Settings.

---

1. Home the Z axis.

2. Ensure there is nothing below the torch then jog the Z axis down until it stops at the Z axis MINIMUM_LIMIT then click the 0 next to the Z axis DRO to **Touch Off** with the Z axis selected to set the Z axis at zero offset. This step only serves to allow the user to more easily visualize and adjust **Probe Height** this value is measured from the Z axis MINIMUM_LIMIT up.
3. Lleva el eje Z a casa nuevamente.

Probe Test

If the machine is equipped with a float switch, then the user will need to set the offset in the **CONFIGURATION** section of the **PARAMETERS** tab. This will be done by running a ”Probe Test” cycle.

1. Check that the Probe Speed and the Probe Height in the **CONFIGURATION** section of the **PARAMETERS** tab are correct. QtPlasmaC can probe at the full Z axis velocity so long as the machine has enough movement in the float switch to absorb any overrun. If the machine is suitable, the user could set the Probe Height to a value near the Z axis minimum and do all probing at full speed.
2. If the machine is not already homed and in the home position, home the machine.
3. Place some material on the slats under the torch.
4. Press the **PROBE TEST** button.
5. The Z axis will probe down, find the material then move up to the specified **Pierce Height** as set by the currently selected material. The torch will wait in this position for the time set in the <*machine_name*>.prefs file. The default probe test hold time is 10 seconds, this value may be edited in the <*machine_name*>.prefs file. After this the torch will return to the starting height.
6. Measure the distance between the material and the tip of the torch while the torch is waiting at **Pierce Height**.
7. If the measurement is greater than the **Pierce Height** of the currently selected material, then reduce the ”Float Travel” in the **CONFIGURATION** section of the **PARAMETERS** tab by the difference between the measured value and the specified value. If the measurement is less than **Pierce Height** of the currently selected material, then increase the ”Float Travel” in the **CONFIGURATION** section of the **PARAMETERS** tab by the difference between the specified value and the measured value.
8. After the adjustments to the ”Float Travel” have been made, repeat the process from #4 above until the measured distance between the material and the torch tip matches the **Pierce Height** of the currently selected material.
9. If the table has a laser or camera for sheet alignment, a scribe, or uses offset probing then the required offsets need to be applied by following the procedure described in Peripheral Offsets. The LASER and/or CAMERA buttons will not be visible until the user has set the appropriate offset(s) and they are recorded in the <*machine_name*>.prefs file.
10. CONGRATULATIONS! The user should now have a working QtPlasmaC Configuration.

---

nota
If the amount of time between the torch contacting the material and when the torch moves up and comes to rest at the Pierce Height seems excessive, see the probing section for a possible solution.



importante
IF USING A **Mesa Electronics THCAD** THEN THE **Voltage Scale** VALUE WAS OBTAINED MATHEMATICALLY. IF THE USER INTENDS TO USE CUT VOLTAGES FROM A MANUFACTURE’S CUT CHART THEN IT WOULD BE ADVISABLE TO DO MEASUREMENTS OF ACTUAL VOLTAGES AND FINE TUNE THE **Voltage Scale** AND **Voltage Offset**.



aviso
PLASMA CUTTING VOLTAGES CAN BE LETHAL, IF THE USER IS NOT EXPERIENCED IN DOING THESE MEASUREMENTS GET SOME QUALIFIED HELP.

10.8.6. Migrating to QtPlasmaC From PlasmaC (AXIS or GMOCCAPY)

Automated migration to QtPlasmaC from PlasmaC is no longer supported. The user will either need to convert the PlasmaC configuration manually, or create a new configuration using the configuration wizard.

10.8.7. Other QtPlasmaC Setup Considerations

10.8.7.1. Low-pass Filter

The plasmac HAL component has a built in low-pass filter that if used is applied to the **plasmac.arc-voltage-in** input pin to filter any noise that could cause erroneous voltage readings. The low-pass filter should only be used after using Halscope to determine the required frequency and whether the amplitude of the noise is large enough to cause any issues. For most plasma machines low-pass is not required and should not be used unless it is required.

The HAL pin assigned to this filter is **plasmac.lowpass-frequency** and is set to 0 (disabled) by default. To apply a low-pass filter to the arc-voltage, the user would edit the following entry in the custom.hal file in the machine's configuration directory to add the appropriate cutoff frequency as measured in Hertz (Hz).

For example:

```
setp plasmac.lowpass-frequency 100
```

El ejemplo anterior dará una frecuencia de corte de 100 Hz.

10.8.7.2. Contact Bounce

Contact bounce from mechanical relays, switches, or external interference may cause some inconsistent behavior of the following switches:

- Float Switch
- Ohmic Probe
- Breakaway Switch
- Arc OK (for modes 1 & 2)

Due to the fact that the software is capable of sampling rates faster than the contact bounce period, it is possible that the software may see contact bounce as several changes in input states occurring in a very small time period, and incorrectly interpret this as a very quick on-off of the input. One method of mitigating contact bounce is to "debounce" the input. To summarize debounce, it requires the input state to be stable at the opposite state of the output state for consecutive delay periods before changing the state of the output.

Debounce delay periods can be changed by editing the appropriate debounce value in the custom.hal file in the <*machine_name*> config directory.

Each increment of delay adds one servo thread cycle to the debounce time. For example: given a servo thread period of 1000000 (measured in nano seconds), a debounce delay of 5 would equate to 5000000 ns, or 5 ms.

For the Float and Ohmic switches this equates to a 0.001 mm (0.00004") increase in the probed height result.

It is recommended to keep the debounce values as low as possible while still achieving consistent results. Using Halscope to plot the inputs is a good way to establish the correct value.

For QtPlasmaC installations, debounce is achieved by using the HAL dbounce component which is a later alternative to the original debounce component. This new version allows for the loading and naming of individual debounce instances and is compatible with Twopass HAL file processing.

All four signals above have an individual debounce component so the debounce periods can be catered individually to each input. Any changes made to these values in the custom.hal file will not be overwritten by later updates of QtPlasmaC.

The default delay for all four inputs is five servo thread periods. In most cases this value will work quite well. If any of the inputs do not use mechanical switches, it may be possible to either reduce or remove the delay for those inputs.

If debounce is required for other equipment like home or limit switches etc. then more dbounce components may added in any of the HAL files without any regard to the signals listed here.

10.8.7.3. Contact Load

Mechanical relays and switches usually require a minimum current passing through the contacts for reliable operation. This current varies with the material that the contacts in the device are made from.

Depending on the specified minimum contact current and the current drawn by the input device there may be a need to provide a method to increase the current through the contacts.

Most relays using gold contacts will not require any additional current for reliable operation.

There are two different methods available to provide this minimum current if it is required:

1. A 0.1 µF film capacitor placed across the contacts.
2. A 1200 Ω 1 W resistor across the load (see calculations below).

Schematics are shown at contact load schematics.

More information on contact switching load can be seen on page VI of the finder General Technical Information document.

Calculations:

If using a Mesa card, the input resistance of a 7I96 is 4700 Ω (symbol R)(always consult the product manual associated with the revision being used as these values sometimes vary between revisions), giving a contact current of 5.1 mA (symbol I) assuming a supply voltage (symbol U) of 24 V (I = U/R)[2].

As an example, the typical relay used in a Hypertherm Powermax 65 plasma cutter (TE T77S1D10-24) requires a minimum contact load of 100 mA @ 5 VDC which will dissipate 0.5 W (P = I * V). If using a 24 VDC power supply this would then equate to a minimum current of 20.8 mA. Because there is less current drawn by the Mesa input than is required by the relay there needs to be an increase in the current.

The resistance can be calculated using $R = U_s / (I_m - I_i)$ where:

- R = calculated resistance

[2] In the US, the letter V is commonly used as a symbol (Voltage) and as a unit (Volt).

- U_s = supply voltage
- I_m = minimum current required
- I_i = input current

Using a 7I96 with an input current of 5.1 mA gives a calculated value of 1529 Ω (= 24 V / (.0208 - .0051) A). This could then be rounded down to a commonly available 1500 Ω resistor, giving a small safety margin.

The power dissipation can by calculated using $P = U_s^2 / R_s$ where:

- P = power
- U_s = supply voltage
- R_s = selected resistance

This gives a value of 0.38 W. This could then be rounded up to 1 W, giving a good safety margin. The final selection would be a 1500 Ω 1 W resistor.

10.8.7.4. Desktop Launcher

If a link to the launch the configuration was not created when creating the config, the user could create a desktop launcher to the config by right clicking on the desktop and selecting Create Launcher or similar. This will bring up a dialog box to create a launcher. Give the icon a nice short name, enter anything for the command and click OK.

After the launcher appears on the desktop, right click on it and then edit it with the user's editor of choice. Edit the file so it looks similar to:

```
[Desktop Entry]
Comment=
Terminal=false
Name=LinuxCNC
Exec=sh -c "linuxcnc $HOME/linuxcnc/configs/<machine_name>/<machine_name>.ini"
Type=Application
Icon=/usr/share/pixmaps/linuxcncicon.png
```

If the user would like a terminal window to open behind the GUI window, then change the Terminal line to:

```
Terminal=true
```

Displaying a terminal can be handy for error and information messages.

10.8.7.5. QtPlasmaC Files

After a successful QtPlasmaC installation, the following files are created in the configuration directory:

Filename	**Función**
<*machine_name*>.ini	Archivo de configuración para la máquina.
<*machine_name*>.hal	HAL para la máquina.
<*machine_name*>.prefs	Archivo de configuración de parámetros y preferencias específicas para QtPlasmaC.
custom.hal	HAL file for user customization.

Filename	Función
custom_postgui.hal	HAL file for user customization which is run after the GUI has initialized.
shutdown.hal	HAL file which is run during the shutdown sequence.
tool.tbl	Tool table used to store offset information for additional tools (scribe, etc.) used by the QtPlasmaC configuration.
qtplasmac	Link to the directory containing common QtPlasmaC support files.
backup	Directory for backups of config files.

nota
<*machine_name*> is whatever name the user entered into the ”Machine Name” field of the configuration wizard program.

nota
Custom commands are allowed in custom.hal and the custom_postgui.hal files as they are not overwritten during updates.

After running a new configuration for the first time the following files will be created in the configuration directory:

Filename	Función
<machine_name>_material.cfg	File for storing the material settings from the MATERIAL section of the PARAMETERS Tab.
update_log.txt	File for storing log of major updates. Major updates are those that make any modification to a user’s configuration.
qtvcp.prefs	File containing the QtVCP preferences.
qtplasmac.qss	File storing the style sheet for the currently loaded session of QtPlasmaC.

nota
The configuration files (<*machine_name*>.ini and <*machine_name*>.hal) that are created by configuration wizard are notated to explain the requirements to aid in manual manipulation of these configurations. They may be edited with any text editor.

nota
The <*machine_name*>.prefs file is plain text and may be edited with any text editor.

10.8.7.6. Archivo INI

QtPlasmaC has some specific <*machine_name*>.ini file variables as follows:

[FILTER] Section

These variables are mandatory.

```
PROGRAM_EXTENSION = .ngc,.nc,.tap G-code File (*.ngc, *.nc, *.tap)
ngc               = qtplasmac_gcode
nc                = qtplasmac_gcode
tap               = qtplasmac_gcode
```

[RS274NGC] Section

These variables are mandatory.

```
RS274NGC_STARTUP_CODE = G21 G40 G49 G80 G90 G92.1 G94 G97 M52P1
SUBROUTINE_PATH       = ./:../../nc_files
USER_M_PATH           = ./:../../nc_files
```

nota
for a imperial config replace G21 above with G20.

nota
both the above paths show the minimum requirements.



importante
SEE PATH TOLERANCE FOR RS274NGC_STARTUP_CODE INFORMATION RELATED TO G64.

[HAL] Section

These variables are mandatory.

```
HALUI           = halui (required)
HALFILE         = _<machine_name>_.hal (the machine HAL file)
HALFILE         = plasmac.tcl (the standard QtPlasmaC HAL file)
HALFILE         = custom.hal (Users custom HAL commands)
POSTGUI_HALFILE = postgui_call_list.hal (required)
SHUTDOWN        = shutdown.hal (shutdown HAL commands)
```

nota
The user could place custom HAL commands in the custom.hal file as this file is not overwritten by QtPlasmaC updates.

[DISPLAY] Section

This variable is mandatory.

```
DISPLAY = qtvcp qtplasmac      (use 16:9 resolution)
        = qtvcp qtplasmac_9x16 (use 9:16 resolution)
        = qtvcp qtplasmac_4x3  (use 4:3 resolution)
```

There are multiple QtVCP options that are described here: QtVCP INI Settings

For example, the following would start a 16:9 resolution QtPlasmaC screen in full screen mode:

```
DISPLAY = qtvcp -f qtplasmac
```

[TRAJ] Section

This variable is mandatory.

```
SPINDLES = 3
```

[AXIS_X] Section

These variables are mandatory.

```
MAX_VELOCITY     = double the value in the corresponding joint
MAX_ACCELERATION = double the value in the corresponding joint
OFFSET_AV_RATIO  = 0.5
```

[AXIS_Y] Section

These variables are mandatory.

```
MAX_VELOCITY     = double the value in the corresponding joint
MAX_ACCELERATION = double the value in the corresponding joint
OFFSET_AV_RATIO  = 0.5
```

[AXIS_Z] Section

These variables are mandatory.

```
MIN_LIMIT        = just below the top of the table's slats
MAX_VELOCITY     = double the value in the corresponding joint
MAX_ACCELERATION = double the value in the corresponding joint
OFFSET_AV_RATIO  = 0.5
```

nota
With the exception of tube cutting with an angular A, B, or C axis, QtPlasmaC uses the LinuxCNC External Offsets feature for all Z axis motion, and for moving the X and/or Y axis for a consumable change or a cut recovery while paused. For more information on this feature, please read External Axis Offsets in the LinuxCNC documentation.

10.8.8. QtPlasmaC GUI Overview

The following sections will give a general overview of the QtPlasmaC layout.

10.8.8.1. Exiting QtPlasmaC

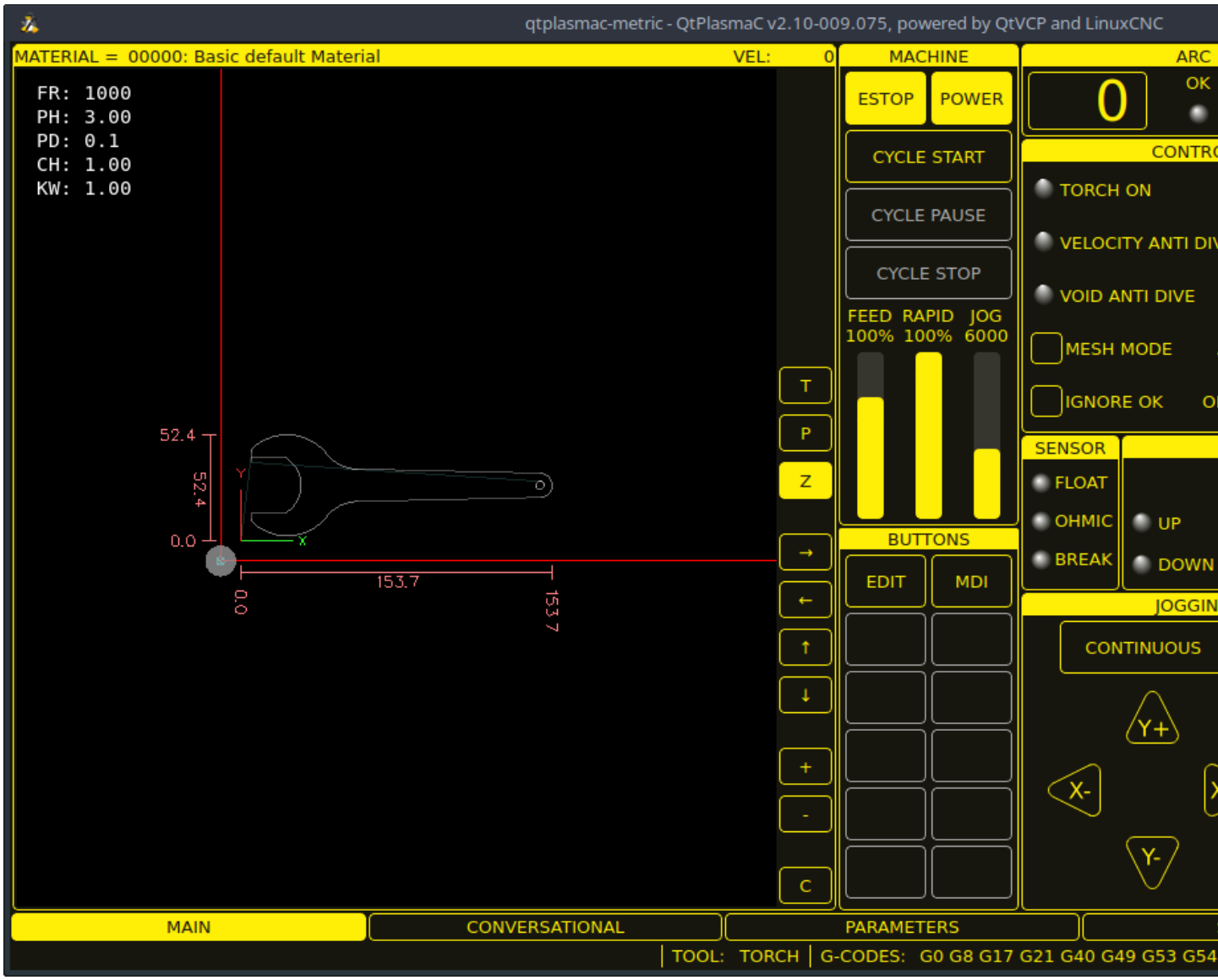
Exiting or shutting down QtPlasmaC is done by either:

1. Click the window shutdown button on the window title bar
2. Long press the **POWER** button on the MAIN Tab.

A shutdown warning can be displayed on every shutdown by checking the **Exit Warning** checkbox on the SETTINGS Tab.

10.8.8.2. MAIN Tab

Screenshot example of the QtPlasmaC MAIN Tab in **16:9** aspect ratio:



Some functions/features are only used for particular modes and are not displayed if they are not required by the chosen QtPlasmaC mode.

Cuadro 10.12: Features of the **PREVIEW WINDOW**

Name	**Descripción**
Material	The top header is clickable in this area to reveal a drop-down menu. It is used to manually select the current material cut parameters. If there are no materials in the material file, then only the default material will be displayed.
VEL:	Displays the current feed rate of the table, including both rapid and cutting moves. While cutting, if a Velocity Reduction is active, this label updates to reflect the percentage of the original feed rate being used. For example, ”VEL@20 %:” means the table is cutting at 20 % of the programmed feed rate - an 80 % reduction.
FR:	If ”View Material” is selected on the SETTINGS Tab, this displays the currently selected material’s Feed Rate.
PH:	If ”View Material” is selected on the SETTINGS Tab, this displays the currently selected material’s Pierce Height.

Cuadro 10.12: (continued)

Name	Descripción
PD:	If "View Material" is selected on the SETTINGS Tab, this displays the currently selected material's Pierce Delay.
CH:	If "View Material" is selected on the SETTINGS Tab, this displays the currently selected material's Cut Height.
CA:	If "View Material" is selected on the SETTINGS Tab, and RS485 communications are enabled, this displays the currently selected material's Cut Amperage.
T	This button changes the preview to a top down full table view.
P	This button changes the preview to an isometric view.
Z	This button changes the preview to a top down view.
→	This button pans the preview right.
←	This button pans the preview left.
↑	This button pans the preview up.
↓	This button pans the preview down.
+	This button zooms the preview.
-	This button zooms the preview.
C	This button clears the live plot.

Cuadro 10.13: **MACHINE** representation

Name	Descripción
ESTOP	Setting Estop type = 0 in the **[GUI_OPTIONS]** section of the <*machine_name*>.prefs file, will change this button to an indicator of the hardware E-stop's status only. This is the default behavior. Setting the option Estop type = 1 in the **[GUI_OPTIONS]** section of the <*machine_name*>.prefs file, will hide this button. Setting the option Estop type = 2 in the **[GUI_OPTIONS]** section of the <*machine_name*>.prefs file, will enable this button to act as a GUI E-stop.
POWER	This button turns the GUI on and allows QtPlasmaC/LinuxCNC to control the hardware. Pressing and holding the **POWER** button for longer than two seconds will bring up a dialog to exit the QtPlasmaC application.
CYCLE START	This button starts the cycle for any loaded G-code file.
CYCLE PAUSE	This button pauses the cycle for any loaded G-code file. If a cycle is paused, this button will display **CYCLE RESUME** and flash. Pressing **CYCLE RESUME** will resume the cycle.
CYCLE STOP	This button stops any actively running or paused cycle. This includes: - G-code Programs - Torch pulse if the pulse was started during **CYCLE PAUSE** (this will cancel the paused G-code program execution as well) - Probe Test - Framing - Manual Cut
FEED	This slider overrides the feed rate for all feed moves. Any value other than 100 % will cause the label to flash. Clicking the label will return the slider to 100 %.
RAPID	This slider overrides the rapid rate for all rapid moves. Any value other than 100 % will cause the label to flash. Clicking the label will return the slider to 100 %.

Cuadro 10.13: (continued)

Name	Descripción
JOG	This slider sets the jog rate. Clicking the label will return the slider to the default linear velocity as set in the <*machine_name*>.ini file.

BUTTONS The Button Panel contains buttons useful for the operation of the machine.

The **EDIT** and **MDI** buttons are permanent, all other buttons are user programmable in the <*machine_name*>.prefs file.

See custom user buttons for detailed information on custom user buttons.

Name	Descripción
EDIT	This button opens a G-code editor for the currently loaded program.
MDI	This button places QtPlasmaC into Manual Data Input (MDI) mode which will display the MDI HISTORY and an entry box over top of the G-code window. Once pressed, this button will display ”MDI CLOSE”. Pressing **MDI CLOSE** will close the MDI. Please see the MDI section for additional MDI information.
OHMIC TEST	This button will enable the Ohmic Probe Enable output signal and if the Ohmic Probe input is sensed, the LED indicator in the SENSOR Panel will light. The main purpose of this is to allow a quick test for a shorted torch tip.
PROBE TEST	This button will initiate a Probe Test.
SINGLE CUT	This button will show the dialog box to start an automatic Single Cut.
NORMAL CUT	This button will toggle between Cut Types (NORMAL CUT and PIERCE ONLY).
TORCH PULSE	This button will initiate a Torch Pulse.

Cuadro 10.15: **ARC**

Name	Modes	Descripción
Arc Voltage	0, 1	Displays the actual arc voltage.
OK	0, 1, 2	Indicates the status of the Arc OK signal.
+	0, 1	Each press of this button will raise the target voltage by the THC Threshold voltage (The distance changed will be Height Per Volt * THC Threshold voltage).
-	0, 1	Each press of this button will lower the target voltage by the THC Threshold voltage (The distance changed will be Height Per Volt * THC Threshold voltage).
OVERRIDE	0, 1	Clicking this label will return any voltage override to 0.00.

Cuadro 10.16: **CONTROL**

Name	Modes	Descripción
TORCH ON	0, 1, 2	Indicates the status of the Torch On output signal.

Cuadro 10.16: (continued)

Name	Modes	Descripción
TORCH ON ENABLE	0, 1, 2	This box toggles between Enabling and Disabling the torch. This box defaults to unfilled (disabled) when QtPlasmaC is first run. This box must be filled to change it to ”Torch Enabled” before material cutting can commence. If this box is not filled, then running a loaded program will cause the machine to run the cycle without firing the torch. This is sometimes referred to as a ”dry run”. If the user has a laser installed, then it is also possible to dry run with the laser. See the LASER section for detailed instructions.
VELOCITY ANTI DIVE	0, 1, 2	Indicates that the THC is locked at the current height due to the cut velocity falling below the Velocity Anti Dive (VAD) Threshold percentage set on the PARAMETERS Tab.
VELOCITY ANTI DIVE ENABLE	0, 1, 2	This box toggles between Enabling and Disabling VELOCITY ANTI DIVE.
VOID ANTI DIVE	0, 1	Indicates that the THC is locked due to a void being sensed.
VOID ANTI DIVE ENABLE	0, 1	This box toggles between Enabling and Disabling VOID ANTI DIVE.
MESH MODE	0, 1, 2	This box will enable or disable Mesh Mode for the cutting of expanded metal. This check box may be enabled or disabled at any time during normal cutting. Mesh mode: - Will require an Arc OK signal to start machine motion. - Will disable the THC. - Will not stop machine motion if the Arc OK signal is lost. - Will automatically select CPA mode if PowerMax communications are being used. For more information see Mesh Mode (expanded metal).
AUTO VOLTS	0, 1	This box will enable or disable Auto Volts.
IGNORE OK	0, 1, 2	This box will determine if QtPlasmaC ignores the Arc OK signal. This check box may be enabled or disabled at any time during normal cutting. Additionally, this mode may be enabled or disabled via proper M codes in a running program. Ignore Arc OK mode: - Will not require an Arc OK signal be received before starting machine motion after the ”Torch On” signal is given. - Will disable the THC. - Will not stop machine motion if the Arc OK signal is lost. For more information see Ignore Arc Ok.
OHMIC PROBE	0, 1, 2	This box enables or disables the ohmic probe input. If the Ohmic Probe input is disabled, the Ohmic Probe LED will still show the status of the probe input, but the Ohmic Probe results will be ignored.
RS485	0, 1, 2	This box will enable or disable the communications to a PowerMax. This button is only visible if a PM_PORT option is configured in the `[POWERMAX]` section of the *<machine_name>*.prefs file.

Cuadro 10.16: (continued)

Name	Modes	Descripción
Status	0, 1, 2	When PowerMax communications are enabled, this will display one of the following: **CONNECTING**, **CONNECTED**, **COMMS ERROR**, or a **Fault Code**. For more information, see the PowerMax Communications section.

Cuadro 10.17: **SENSOR**

Name	Descripción
FLOAT	Indicates that the float switch is activated.
OHMIC	Indicates that the probe has sensed the material.
BREAK	Indicates that the torch breakaway sensor is activated.

Cuadro 10.18: **THC**

Name	Descripción
ENABLE	This box determines whether the THC will be enabled or disabled during a cut.
ENABLED	This LED indicates whether the THC is enabled or disabled.
ACTIVE	This LED indicates that the THC is actively controlling the Z axis.
UP	This LED indicates that the THC is commanding the Z axis to raise.
DOWN	This LED indicates that the THC is commanding the Z axis to lower.

nota
During Paused Motion, this section will become CUT RECOVERY

Name	Descripción
CONTINUOUS	This drop-down button will change the jog increment. Options are determined by the values in the **[DISPLAY]** section of the <*machine_name*>.ini file and begin with the label ”INCREMENTS =”.
FAST	This button will toggle between FAST which is the default linear velocity in the <*machine_name*>.ini file or SLOW which is 10 % of the default value.
Y+	Este botón mueve el eje Y en dirección positiva.
Y-	Este botón mueve el eje Y en dirección negativa.
X+	Este botón mueve el eje X en dirección positiva.
X-	Este botón mueve el eje X en dirección negativa.
Z+	Este botón mueve el eje Z en dirección positiva.
Z-	Este botón mueve el eje Z en dirección negativa.

nota
During Paused Motion, this section will be shown on top of the JOGGING panel. The following section will cover each button encountered in this panel. Please see CUT RECOVERY for a detailed description of the cut recovery functionality.

Name	Descripción
PAUSED MOTION FEED SLIDER	In the event of a paused program, this interface allows X/Y motion to follow the programmed path in the reverse or forward direction. This slider's range is from 1 %-100 % of the Cut Feed Rate for the currently selected material.
FEED	This displays the paused motion feed rate.
REV	In the event of a paused program, this button will move the machine in reverse along the programmed path until it reaches the last M3 command that was either executed or that QtPlasmaC was attempting to execute before the program became paused.
FWD	In the event of a paused program, this button will move the machine forward along the programmed path indefinitely until the program's end, skipping over M3 commands.
CANCEL MOVE	This button will cancel any Cut Recovery movement that was made and return the torch to the position the Cut Recovery movement was initiated. Note that if FWD or REV were used to move the torch, CANCEL will not return to the position of the torch when the pause occurred.
MOVE x.xxx	This displays the amount of travel that will be incurred with each press of an arrow key, in the direction the arrow key was pressed. This value displayed below MOVE represents the Kerf Width of the currently selected material.
DIRECTIONAL ARROWS	These buttons will move the torch in the direction indicated by a distance of one Kerf Width (of the currently selected material) per press.

Cuadro 10.21: **G-CODE WINDOW**

Name	Descripción
CLEAR	This button will clear the currently opened program. If a file is open, the default material will be selected. If no file is open, the preview will be reset to a top down full table view. The torch (T0) will be selected if it was not the active tool. Previous error messages, and the error status will be cleared. Cut type will be set to NORMAL CUT.
OPEN	This button will open a FILE OPEN panel over the PREVIEW WINDOW.
RELOAD	Este botón recargará el archivo de código G cargado actualmente.

Cuadro 10.22: **DRO**

Name	Descripción
HOME ALL	This button will home all of the axes in the order set by HOME_SEQUENCE in the <*machine_name*>.ini file.
WCS G54	Esta lista desplegable cambiará el offset de trabajo actual.
CAMERA	This button will display a CAMVIEW panel on top of the PREVIEW WINDOW and will allow the user to set an origin with or without rotation. See the CAMERA section for detailed instructions. This button will not be visible until a CAMERA offset is set in the <*machine_name*>.prefs file.
LASER	This button will allow the user to use a laser to set an origin with or without rotation. See the LASER section for detailed instructions. This button will not be visible until a LASER offset is set in the <*machine_name*>.prefs file.
X0 Y0	Este botón establecerá la posición actual a X0 Y0.
HOME [AXIS]	This button will home the corresponding axis.

Cuadro 10.22: (continued)

Name	Descripción
0 [AXIS]	This drop-down button will display the following options: **Zero** - zeros the axis. **Set** - launches a dialog box to manually input the axis' coordinate. **Divide By 2** - divides the currently displayed coordinate in the DRO by two. **Set To Last** - sets the axis to the previously set coordinate.

10.8.8.3. Preview Views

The QtPlasmaC preview screen has the ability to be switched between different views and displays, as well as zooming in and out, and panning horizontally and vertically.

When QtPlasmaC is first started, the Z (top down) view will be selected as the default view for a loaded G-code file, but the full table view will be displayed.

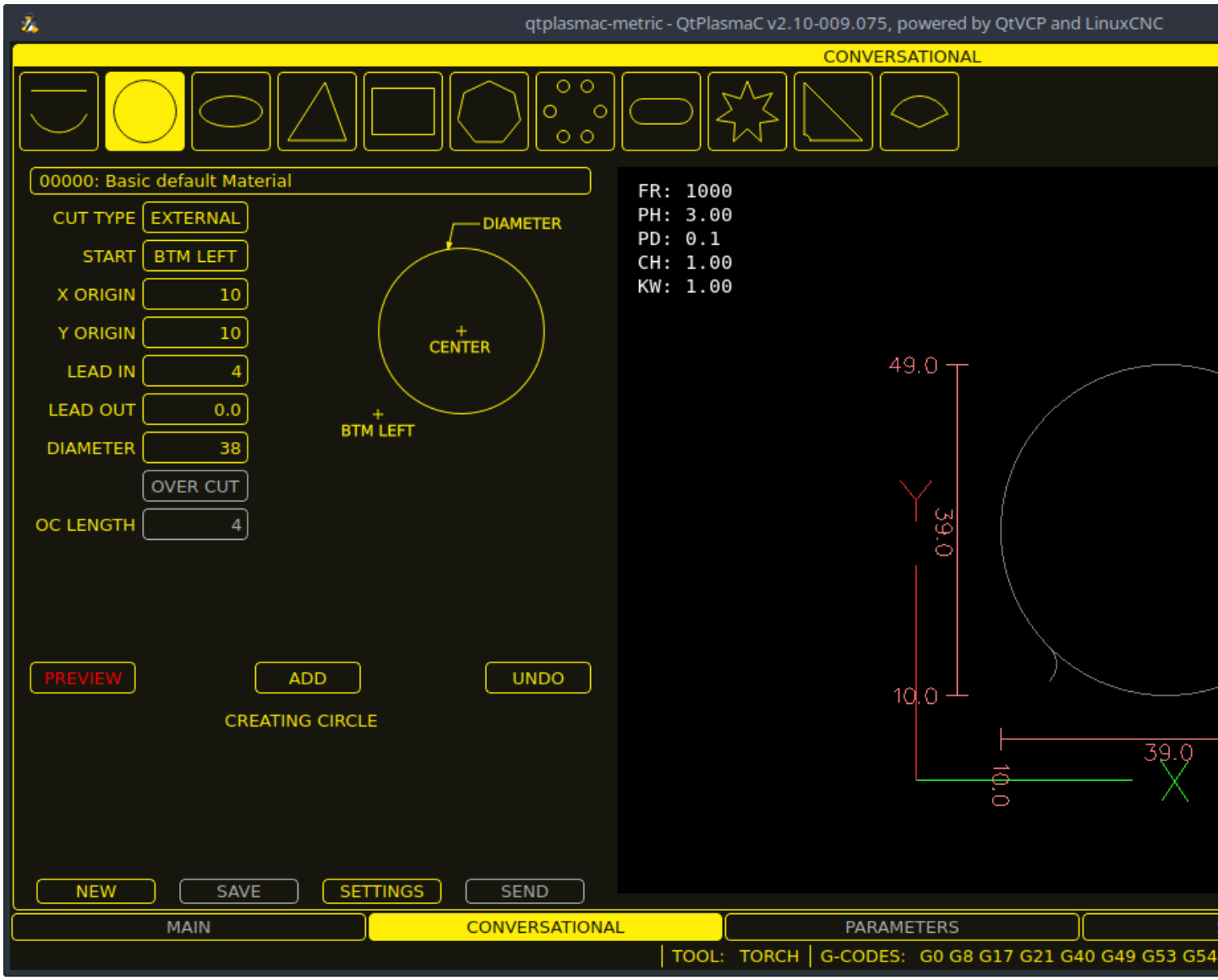
When a G-code file is loaded, the display will change to the selected view.

Whenever there is no G-code file loaded, the full table will automatically be displayed irrespective of which view is currently selected (the highlighted button representing the currently selected view will not change).

If a full table is displayed due to no G-code file being loaded and the user wishes to change the view orientation, then pressing either Z or P will change the display to the newly selected view. If the user then wishes to display the full table while maintaining the currently selected view as the default view for a loaded G-code file, then pressing CLEAR will achieve this and allow the selected view orientation to prevail the next time a G-code file is loaded.

10.8.8.4. CONVERSATIONAL Tab

Screenshot example of the QtPlasmaC CONVERSATIONAL Tab in **16:9** aspect ratio:



The CONVERSATIONAL Tab enables the user to quickly program various simple shapes for quick cutting without the need for CAM software.

See Conversational Shape Library for detailed information on the Conversational feature.

It is possible to hide this tab so the conversational feature cannot be used by an operator. This may be achieved either by wiring the pin to a physical key-switch or similar or it may also be set in a HAL file using the following command:

```
setp qtplasmac.conv_disable 1
```

10.8.8.5. PARAMETERS Tab

Screenshot example of the QtPlasmaC PARAMETERS Tab in **16:9** aspect ratio:



Some functions/features are only used for particular modes and are not displayed if they are not required by the chosen QtPlasmaC mode.

This tab is used to display configuration parameters that are modified infrequently.

It is possible to hide this tab so machine settings cannot be modified by unauthorized personnel. This may be achieved either by wiring the pin to a physical key-switch or similar or it may also be set in a HAL file using the following command:

```
setp qtplasmac.param_disable 1
```

Cuadro 10.23: **CONFIGURATION - ARC**

Name	Modes	Descripción
Start Fail Timer	0, 1, 2	This sets the amount of time (in seconds) QtPlasmaC will wait between commanding a ”Torch On” and receiving an Arc OK signal before timing out and displaying an error message.
Max Starts	0, 1, 2	This sets the number of times QtPlasmaC will attempt to start the arc.

Cuadro 10.23: (continued)

Name	Modes	Descripción
Retry Delay	0, 1, 2	This sets the time (in seconds) between an arc failure and another arc start attempt.
Voltage Scale	0, 1	This sets the arc voltage input scale and is used to display the correct arc voltage. For initial setup, see Calibration Values.
Voltage Offset	0, 1	This sets the arc voltage offset and is used to display zero volts when there is zero arc voltage input. For initial setup, see Calibration Values.
Height Per Volt	0, 1, 2	This sets the distance the torch would need to move to change the arc voltage by one volt. Used for manual height manipulation only.
OK High Volts	0	This sets the voltage threshold below which Arc OK signal is valid.
OK Low Volts	0	This sets the voltage threshold above which the Arc OK signal is valid.

nota
When setting the OK Low Volts and OK High Volts in Mode 0, the cut voltage of a stable arc must be greater than the OK Low Volts value but lower than the OK High Volts value for QtPlasmaC to receive a valid Arc OK signal. To further clarify, to have a valid Arc OK, the arc voltage must fall between the two limits.

Cuadro 10.24: **CONFIGURATION - PROBING**

Name	Descripción
Float Travel	This sets the amount of travel the float switch moves before completing the float switch circuit. This distance can be measured by using the Probe Test button, and the method described in Initial Setup.
Probe Speed	This sets the speed at which the torch will probe to find the material after it moves to the Probe Height.
Probe Height	This sets the height above the Z axis minimum limit that Probe Speed begins. If set to zero, then the torch will move at Probe Speed from the current position. This may be adventageous when using Slat Height and Material Thickness as the machine will default to Probe Height if either Slat Height or Material Thickness are zero. Refer to the Heights Diagrams for a visual representation.
Slat Height	This sets the height of the slats, measured up from Z axis minimum limit. This must be used in conjunction with Material Thickness. If either Slat Height or Material Thickness are zero then the machine will default to Probe Height. Refer to the Heights Diagrams for a visual representation.
Ohmic Offset	This sets the distance above the material the torch will should go after a successful ohmic probe. It is mainly used to compensate for high probing speeds.
Ohmic Retries	This sets the number of times QtPlasmaC will retry a failed ohmic probe before falling back to the float switch for material detection.
Skip IHS	This sets the distance threshold used to determine if an Initial Height Sense (probe) can be skipped for the current cut, see IHS Skip.
Offset Speed	This sets the speed at which the probe will move to the offset position in the X axis and Y axis.

nota
If the amount of time between the torch contacting the material and when the torch moves up and comes to rest at the Pierce Height seems excessive, see the probing section for a possible solution.

Cuadro 10.25: **CONFIGURATION - SAFETY**

Name	Descripción
Safe Height	This sets the height above the material that the torch will retract to before executing rapid moves. If set to zero, then Max Offset (plasmac.max-offset) will be used for the safe height. This defaults to 5mm (0.197”). Refer to the Heights Diagrams for a visual representation.

Cuadro 10.26: **CONFIGURATION - SCRIBING**

Name	Descripción
Arm Delay	This sets the delay (in seconds) from the time the scribe command is received to the activation of the scribe. This allows the scribe to reach surface of the material before activating the scribe.
On Delay	This sets the delay (in seconds) to allow the scribe mechanism to start before beginning motion.

Cuadro 10.27: **CONFIGURATION - SPOTTING**

Name	Descripción
Threshold	This sets the arc voltage at which the delay timer will begin. 0 V starts the delay when the torch on signal is activated.
Time On	This sets the length of time (in milliseconds) the torch is on after threshold voltage is reached.

Cuadro 10.28: **CONFIGURATION - PIERCE ONLY**

Name	Descripción
X Offset	Moves the pierce point this distance along the X axis when piercing in Pierce Only mode.
Y Offset	Moves the pierce point this distance along the Y axis when piercing in Pierce Only mode.

Cuadro 10.29: **CONFIGURATION - MOTION**

Name	Descripción
Setup Speed	The Z axis velocity for setup moves (movements to Probe Height, Pierce Height, Cut Height, etc.).

nota
Setup Speed has no effect on THC speed which is capable of the velocity displayed in the Max. Speed field.

Cuadro 10.30: **CONFIGURATION - THC**

Name	Modes	Descripción
Delay	0, 1, 2	This sets the delay (in seconds) measured from the time the Arc OK signal is received until Torch Height Controller (THC) activates. This is only available when Auto THC is not enabled.
Sample Counts	0, 1	This sets the number of consecutive arc voltage readings within THC Sample Threshold required to activate the Torch Height Controller (THC). This is only available when Auto THC is enabled.
Sample Threshold	0, 1	This sets the maximum voltage deviation allowed for THC Sample Counts. This is only available when Auto THC is enabled.
Threshold	0, 1	This sets the voltage variation allowed from the target voltage before for THC makes movements to correct the torch height.
Speed (PID-P)	0, 1, 2	This sets the Proportional gain for the THC PID loop. This roughly equates to how quickly the THC attempts to correct changes in height.
VAD Threshold	0, 1, 2	(Velocity Anti Dive) This sets the percentage of the current cut feed rate the machine can slow to before locking the THC to prevent torch dive.
Void Slope	0, 1	(Void Anti Dive) This sets the size of the change in cut voltage per seconds necessary to lock the THC to prevent torch dive (higher values need greater voltage change to lock THC).
PID-I	0, 1	This sets the Integral gain for the THC PID loop. Integral gain is associated with the sum of errors in the system over time and is not always needed.
PID-D	0, 1	This sets the Derivative gain for the THC PID loop. Derivative gain works to dampen the system and reduce over correction oscillations and is not always needed.

Two methods of THC activation are available and are selected with the **Auto Activation** check-button. Both methods begin their calculations when the current velocity of the torch matches the cut feed rate specified for the selected material:

1. Delay Activation (the default) is selected when **Auto Activation** is unchecked. This method uses a time delay set with the **Delay** parameter.
2. Auto Activation is selected when **Auto Activation** is checked. This method determines that the arc voltage is stable by using the **Sample Counts** and **Sample Threshold** parameters.

nota
PID loop tuning is a complicated process and is outside the scope of this User Guide. There are many sources of information available to assist with understanding and tuning PID loops. If the THC is not making corrections fast enough, it is recommended to increase the P gain in small increments until the system operates favorably. Large P gain adjustments can result in over correction and oscillations.

SAVE & RELOAD Buttons The **SAVE** button will save the currently displayed parameters to the <*machine_name*>.prefs file.

The **RELOAD** button will reload all the parameters from the <*machine_name*>.prefs file.

Cuadro 10.31: **MATERIAL** - The parameters which are active for the current cut.

Name	**Descripción**
Material	The top drop-down menu is used to manually select the current material cut parameters. If there are no materials in the material file, then only the default material will be displayed.
Thickness	This sets the thickness for the currently selected material. This must be used in conjunction with Slat Height. If either Slat Height or Material Thickness are zero then the machine will default to Probe Height. Refer to the Heights Diagrams for a visual representation.
Kerf Width	This sets the kerf width for the currently selected material. Refer to the Heights Diagrams for a visual representation.
Pierce Height	This sets the pierce height for the currently selected material. Refer to the Heights Diagrams for a visual representation.
Pierce Delay	This sets the pierce delay (in seconds) for the currently selected material.
Cut Height	This sets the cut height for the currently selected material. Refer to the Heights Diagrams for a visual representation.
Cut Feed Rate	This sets the cut feed rate for the currently selected material.
Cut Amps	This sets the cut amperage for the currently selected material. This is a visual indicator to the operator only, unless PowerMax communications are being used.
Cut Volts	This sets the cut voltage for the currently selected material.
Puddle Height	Expressed as a percentage of Pierce Height, this sets the Puddle Jump height for the currently selected material. Typically used for thicker materials, Puddle Jump allows the torch to have an intermediate step between Pierce Height and Cut Height. If set, the torch will proceed from Pierce Height to P-Jump Height for a period of time (P-Jump Delay) before proceeding to Cut Height to effectively ”jump” over the molten puddle. Refer to the Heights Diagrams for a visual representation.
Puddle Delay	This sets the amount of time (in seconds) the torch will stay at the P-Jump Height before proceeding to Cut Height.
Pause At End	This sets the amount of time (in seconds) the torch will stay on at the end of the cut before proceeding with the M5 command to turn off and raise the torch. For more information see Pause At End Of Cut.
Gas Pressure	This sets the gas pressure for the currently selected material. This setting is only valid if PowerMax communications are being used. 0 = Use the PowerMax’s automatic pressure mode.

Cuadro 10.31: (continued)

Name	Descripción
Cut Mode	This sets the cut mode for the currently selected material. This setting is only valid if PowerMax communications are being used. 1 = Normal 2 = CPA (Constant Pilot Arc) 3 = Gouge/Mark

nota
See the thick materials section for more information on puddle jump.

SAVE, RELOAD, NEW, & DELETE Buttons The **SAVE** button will save the current material set to the <*machine_name*>_material.cfg file.

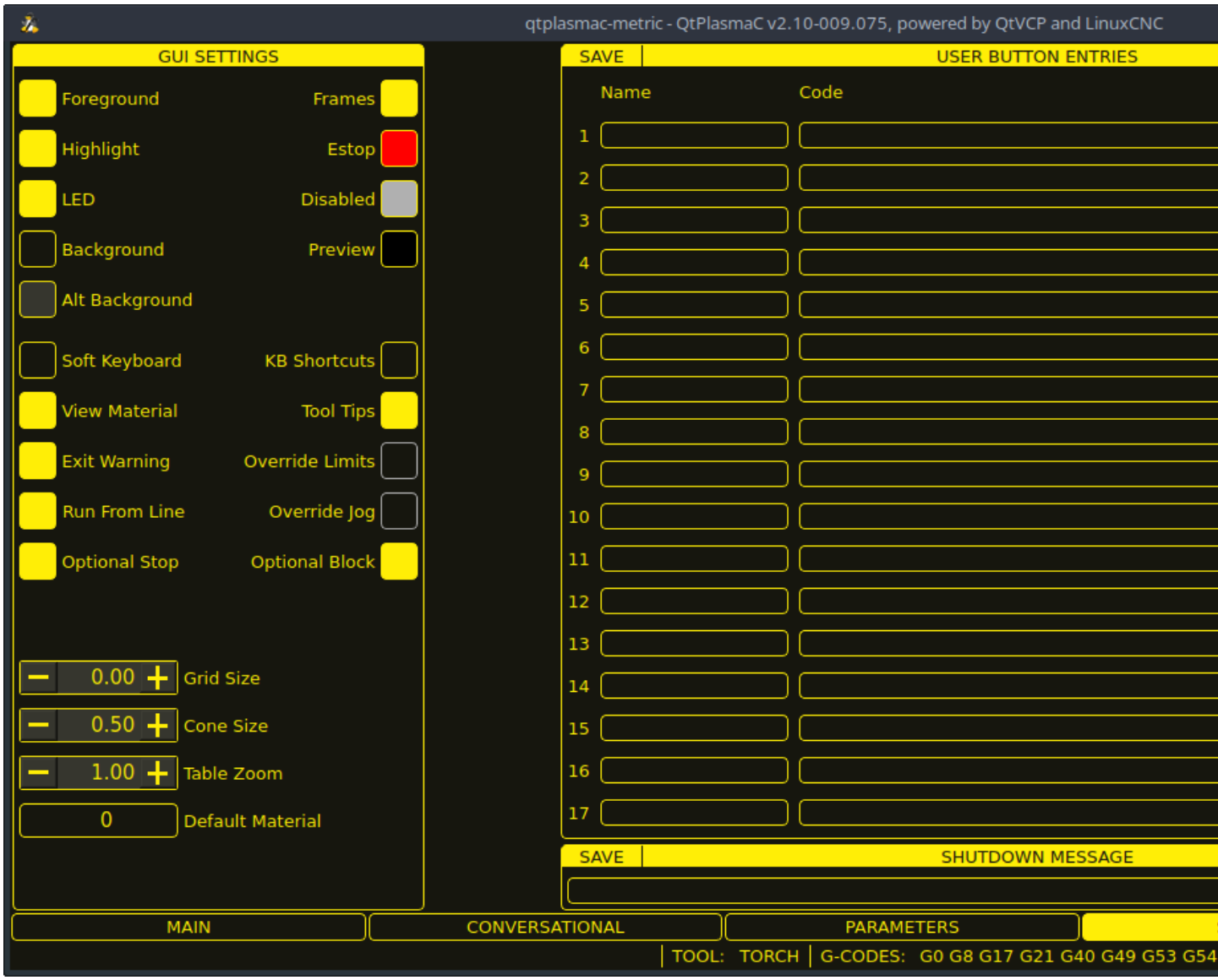
The **RELOAD** button will reload the material set from the <*machine_name*>_material.cfg file.

The **NEW** button will allow a new material to be added to the material file. The user will be prompted for a material number and a material name, all other parameters will be read from the currently selected material. Once entered, QtPlasmaC will reload the material file and display the new material. The Cut Parameters for the new material will then need to be adjusted and saved.

The **DELETE** this button is used to delete a material. After pressing it, the user will be prompted for a material number to be deleted, and prompted again to ensure the user is sure. After deletion, the material file will be reloaded and the drop-down list will display the default material.

10.8.8.6. SETTINGS Tab

Screenshot example of the QtPlasmaC SETTINGS Tab in **16:9** aspect ratio:



This tab is used to display GUI configuration parameters, button text, and shutdown text that are modified infrequently as well as some utility buttons.

It is possible to hide this tab so machine settings cannot be modified by unauthorized personnel. This may be achieved either by wiring the pin to a physical key-switch or similar or it may also be set in a HAL file using the following command:

```
setp qtplasmac.settings_disable 1
```

GUI SETTINGS This section shows parameters that effect the GUI appearance and GUI behaviors.

To return any of the color changes to their default values, see the Returning To The Default Styling section.

Cuadro 10.32: **GUI SETTINGS** Parameters that effect the GUI appearance and GUI behaviors.

Name	Descripción
Foreground	This button allows the user to change the color of the GUI Foreground.
Highlight	This button allows the user to change the color of the GUI Highlight.

Cuadro 10.32: (continued)

Name	Descripción
LED	This button allows the user to change the color of the GUI LED.
Background	This button allows the user to change the color of the GUI Background.
Alt Background	This button allows the user to change the color of the GUI Alternate Background.
Frames	This button allows the user to change the color of the GUI Frames.
Estop	This button allows the user to change the color of the GUI Estop.
Disabled	This button allows the user to change the color of the GUI's Disabled features.
Preview	This button allows the user to change the color of the GUI Preview Window Background.
Soft Keyboard	This radio button allows the user to enable or disable the soft touchscreen keyboard. If the "onboard" virtual keyboard is installed then the custom layouts will be enabled.
KB Shortcuts	This radio button allows the user to enable or disable Keyboard Shortcuts within the GUI (such as keyboard jogging). In addition to the standard jog keys, a list of the additional shortcuts is available in the keyboard shortcuts section.
View Material	This radio button allows the user to enable or disable the addition of a visual reference showing key material cut settings to the Preview Windows of the MAIN and CONVERSATIONAL tabs. Examples are: Feed Rate, Pierce Height, Pierce Delay, and Cut Height. Cut Amps will be shown if PowerMax communications are enabled.
Exit Warning	This radio button allows the user to enable or disable whether a warning will always be displayed during shutdown. It is possible to add a custom message to the warning by editing the EXIT WARNING MESSAGE option in the **[GUI_OPTIONS]** section of the <*machine_name*>.prefs file. The custom message can be made multi-line by adding a "\" between lines.
Parada opcional	This radio button allows the user to enable or disable whether or not a running program will pause at an **M1** command.
Run From Line	This radio button allows the user to enable or disable Run From Line. If enabled, the user can click on a line of G-code and have the program start from that line.
Anulación de límites	This radio button allows the user to temporarily Override the input from a Limit Switch in the event the limit switch becomes tripped during operation. This button can only be clicked when a limit switch is tripped.
Override Jog	This radio button will also allow jogging while jogging is inhibited due to a float switch, breakaway switch, or ohmic probe activation. This button can only be clicked when a jog is inhibited.
Bloqueo opcional	This radio button allows the user to enable or disable whether or not lines starting with "/" will be skipped if present in a running program.
Grid Size	This allows a user to change the size of the grid in the Preview Window on the MAIN Tab. Grid size of 0.0 will disable the grid.
Cone Size	This allows a user to change the size of the cone (which represents the current tool) in the Preview Window on the MAIN Tab.
Table Zoom	This allows a user to change the default zoom level for the top down full table view in the Preview Window on the MAIN Tab.

USER BUTTON ENTRIES USERBUTTON

This section shows the text that appears on the Custom User Buttons as well as the code associated

with the user button. User buttons may be changed, and the new settings used without restarting LinuxCNC.

The text and/or code may be edited at any time and will be loaded ready for use if the **SAVE** button is clicked.

Deleting the **Name** and **Code** text will cause that user button to be hidden if the **SAVE** button is clicked.

To return all the **Name** and **Code** text to their last saved values press the **RELOAD** button.

Name	**Código**
The text that is displayed on the button	The code that is run when the button is pressed.

nota
There are 20 user buttons available but not all may be displayed depending on the window size.

EXIT WARNING MESSAGE

This section shows the text that appears on the shutdown dialog if the **Exit Warning** is enabled.

The text may be edited at any time and will be loaded ready for use if the **SAVE** button is clicked.

To return the **EXIT WARNING MESSAGE** text to the last saved value press the **RELOAD** button.

UTILITIES Some standard LinuxCNC utilities are provided as an aid in the diagnosis of issues that may arise:

- Halshow
- Halscope
- Halmeter
- Calibration
- Status

In addition the following two QtPlasmaC specific utilities are provided:

The **SET OFFSETS** button is used if the table has a laser or camera for sheet alignment, a scribe, or uses offset probing. The required offsets for these peripherals need to be applied by following the procedure described in Peripheral Offsets.

The **BACKUP CONFIG** button will create a complete machine configuration backup for archiving or to aid in fault diagnosis. A compressed backup of the machine configuration will be saved in the user's Linux home directory. The file name will be <*machine_name*><*version*><date>_<time>.tar.gz, where <*machine_name*> is the machine name entered in the configuration wizard, <*version*> is the current QtPlasmaC version the user is on, <*date*> is the current date (YY-MM-DD), and <*time*> is the current time (HH-MM-SS).

Prior to the backup being made, the machine log will be saved to a file in the configuration directory named machine_log_<date>_<time>.txt where <*date*> and <*time*> are formatted as described above. This file along with up to five previous machine logs will also be included in the backup.

These files are not required by QtPlasmaC and are safe to delete at any time.

10.8.8.7. STATISTICS Tab

The STATISTICS Tab provides statistics to allow for the tracking of consumable wear and job run times. These statistics are shown for the current job as well as the running total. Previous job statistics are reset once the next program is run. The total values may be reset either individually by clicking the corresponding ”RESET” button, or they may all be reset together by clicking ”RESET ALL”.

The **RS485 PMX STATISTICS** panel will only be displayed if the user has Hypertherm PowerMax communications and a valid RS485 connection to the PowerMax is established. This panel will show the **ARC ON TIME** for the PowerMax in hh:mm:ss format.

The **MACHINE LOG** is also displayed on the STATISTICS Tab, this log will display any errors and/or important information that occurs during the current LinuxCNC session. If the user makes a backup of the configuration from the SETTINGS Tab then the machine log is also included in the backup.



10.8.9. Using QtPlasmaC

Once QtPlasmaC is successfully installed, no Z axis motion is required to be part of the G-code cut program. In fact, if any Z axis references are present in the cut program, the standard QtPlasmaC configuration will remove them during the program loading process.

For reliable use of QtPlasmaC the user should **NOT** use any Z axis offsets other than the coordinate system offsets (G54-G59.3). For this reason, G92 offsets have been disabled across the GUI.

QtPlasmaC will automatically add a line of G-code to move the Z axis to the correct height at the beginning of every G-code program.

---

nota
It is possible to keep Z motion for use with different tools by adding the magic comment #<keep-z-motion>=1. If using an angular A, B, or C axis for tube cutting then Z axis motion is required in the G-code file.

Version Information - QtPlasmaC will display versioning information in the title of the main window. The information will be displayed as followed "QtPlasmaC v*N.XXX.YYY* - powered by QtVCP on LinuxCNC vZ.Z.Z" where *N* is the version of QtPlasmaC, *XXX* is the version of the HAL component (PlasmaC.comp), *YYY* is the GUI version, and *Z.Z.Z* is the version of LinuxCNC.

10.8.9.1. Units Systems

All settings and parameters in QtPlasmaC are required to be in the same units as specified in the <*machine_name*>.ini file, being either metric or imperial.

If the user is attempting to run a G-code file that is in the "other" unit's system then all parameters including the material file parameters are still required to be in the native machines units. Any further conversions necessary to run the G-code file will be handled automatically by the G-code filter program.

For example: If a user had a metric machine and wished to run a G-code file that was set up to cut 1/4" thick material using imperial units (inch - G20) then the user with the metric machine would need to ensure that either the material number in the G-code file was set to the corresponding metric material to be cut, or that a new material is created with the correct metric parameters for the metric material to be cut. If the metric user wanted to cut the G-code file using imperial material, then the new material parameters would need to be converted from imperial units to metric when they are entered.

10.8.9.2. Preamble and Postamble Codes

The following stanzas are the minimum recommended codes to include in the preamble and postamble of any G-code file to be run by QtPlasmaC:

Metric:

```
G21 G40 G49 G64p0.1 G80 G90 G92.1 G94 G97
```

Imperial:

```
G20 G40 G49 G64p0.004 G80 G90 G92.1 G94 G97
```

A detailed explanation of each G-code can be found in the docs here.

Note that throughout this user guide there are several additional recommendations for codes that are prudent to add to both the preamble and postamble depending on the features the user wishes to utilize.

10.8.9.3. Mandatory Codes

Aside from the preamble code, postamble code, and X/Y motion code, the only mandatory G-code syntax for QtPlasmaC to run a G-code program using a torch for cutting is `M3 $0 S1` to begin a cut and `M5 $0` to end a cut.

For backwards compatibility it is permissible to use `M3 S1` in lieu of `M3 $0 S1` to begin a cutting job and `M5` in lieu of `M5 $0` to end a cutting job. Note, that this applies to cutting jobs only, for scribe and spotting jobs the `$n` tool identifier is mandatory.

10.8.9.4. Coordinates

See recommended Z axis settings.

Each time LinuxCNC (QtPlasmaC) is started Joint homing is required. This allows LinuxCNC (QtPlasmaC) to establish the known coordinates of each axis and set the soft limits to the values specified in the <*machine_name*>.ini file in order to prevent the machine from crashing into a hard stop during normal use.

If the machine does not have home switches, then the user needs to ensure that all axes are at the home coordinates specified in the <*machine_name*>.ini file before homing.

If the machine has home switches, then it will move to the specified home coordinates when the Joints are homed.

Depending on the machine's configuration there will either be a **Home All** button or each axis will need to be homed individually. Use the appropriate button/buttons to home the machine.

As mentioned in the Initial Setup section, it is recommended that the first time QtPlasmaC is used that the user ensure there is nothing below the torch then jog the Z axis down until it stops at the Z axis MINIMUM_LIMIT then click the 0 next to the Z axis DRO to **Touch Off** with the Z axis selected to set the Z axis at zero offset. This should not need to be done again.

If the user intends to place the material in the exact same place on the table every time, the user could jog the X and Y axes to the machine to the corresponding X0 Y0 position as established by the CAM software and then **Touch Off** both axes with a zero offset.

If the user intends to place the material randomly on the table, then the user must **Touch Off** the X and Y axes at the appropriate position before starting the program.

10.8.9.5. Cut Feed Rate

QtPlasmaC is able to read a material file to load all the required cut parameters. To enable to G-code file to use the cut feed rate setting from the cut parameters use the following code in the G-code file:

```
F#<_hal[plasmac.cut-feed-rate]>
```

It is possible to use the standard G-code **F** word to set the cut feed rate as follows:

```
F 1000
```

If the **F** word is used, and the **F** word value does not match the cut feed rate of the selected material then a warning dialog will indicate this during loading of the G-code file.

10.8.9.6. Material File

Material handling uses a material file that was created for the machine configuration when the configuration wizard was ran and allows the user to conveniently store known material settings for easy

recall either manually or automatically via G-code. The resulting material file is named ***<machine_name>*_material.cfg**.

QtPlasmaC does not require the use of a material file. Instead, the user could change the cut parameters manually from the MATERIAL section of the PARAMETERS Tab. It is also not required to use the automatic material changes. If the user does not wish to use this feature, they can simply omit the material change codes from the G-code file.

It is also possible to not use the material file and automatically load materials from within the G-code file.

Material numbers in the materials file do not need to be consecutive nor do they need to be in numerical order.

The following variables are mandatory, and an error message will appear if any are not found when the material file is loaded.

- PIERCE_HEIGHT
- PIERCE_DELAY
- CUT_HEIGHT
- CUT_SPEED

nota
If doing tube cutting using the #<tube_cut>=1 magic comment then the only mandatory variable is PIERCE_DELAY, all other variables are optional.

The following variables are optional. If they are not detected or have no value assigned, they will be assigned a value of 0 and no error message will appear.

- NAME
- KERF_WIDTH
- THC
- PUDDLE_JUMP_HEIGHT
- PUDDLE_JUMP_DELAY
- CUT_AMPS
- CUT_VOLTS
- PAUSE_AT_END
- GAS_PRESSURE
- CUT_MODE

nota
Material numbers 1000000 and above are reserved for temporary materials.



aviso
It is the responsibility of the operator to ensure that the variables are included if they are a requirement for the G-code to be run.

The material file uses the following format:

```
[MATERIAL_NUMBER_1]
NAME                = name
KERF_WIDTH          = value
THC                 = value (0 = off, 1 = on)
PIERCE_HEIGHT       = value
PIERCE_DELAY        = value
PUDDLE_JUMP_HEIGHT  = value
PUDDLE_JUMP_DELAY   = value
CUT_HEIGHT          = value
CUT_SPEED           = value
CUT_AMPS            = value (for info only unless PowerMax communications is enabled)
CUT_VOLTS           = value (modes 0 & 1 only, if not using auto voltage sampling)
PAUSE_AT_END        = value
GAS_PRESSURE        = value (only used for PowerMax communications)
CUT_MODE            = value (only used for PowerMax communications)
```

It is possible to add new material, delete material, or edit existing material from the PARAMETERS tab.. It is also possible to achieve this by using magic comments in a G-code file.

The material file may be edited with a text editor while LinuxCNC is running. After any changes have been saved, press **Reload** in the MATERIAL section of the PARAMETERS Tab to reload the material file.

10.8.9.7. Manual Material Handling

For manual material handling, the user would manually select the material from the materials list in the MATERIAL section of the PARAMETERS Tab before starting the G-code program. In addition to selecting materials with materials list in the MATERIAL section of the PARAMETERS Tab, the user could use the MDI to change materials with the following command:

```
M190 Pn
```

The following code is the minimum code necessary to have a successful cut using the manual material selection method:

```
F#<_hal[plasmac.cut-feed-rate]>
M3 $0 S1
.
.
M5 $0
```

---

nota
Manual material handling will restrict the user to only one material for the entire job.

10.8.9.8. Automatic Material Handling

For automatic material handling, the user would add commands to their G-code file which will enable QtPlasmaC to change the material automatically.

The following codes may be used to allow QtPlasmaC to automatically change materials:

- **M190 P*n*** - Changes the currently displayed material to material number *n*.
- **M66 P3 L3 Q1** - Adds a small delay (1 second in this example) to wait for QtPlasmaC to confirm that it successfully changed materials.

- **F#<_hal[plasmac.cut-feed-rate]>** - Sets the cut feed rate to the feed rate shown in the MATERIAL section of the PARAMETERS Tab.

For automatic material handling, the codes MUST be applied in the order shown. If a G-code program is loaded which contains one or more material change commands then the first material will be displayed in the top header of the PREVIEW WINDOW on the MAIN Tab as the program is loading.

Minimum code necessary to have a successful cut using the automatic material selection method:

```
M190 Pn
M66 P3 L3 Q1
F#<_hal[plasmac.cut-feed-rate]>
M3 $0 S1
.
.
M5 $0
```

nota
Returning to the default material prior to the end of the program is possible with the code **M190 P-1**.

10.8.9.9. Material Addition Via Magic Comments In G-code

By using ”magic comments” in a G-code file it is possible to do the following:

- Add new materials to the <*machine_name*>_material.cfg file.
- Edit existing materials in the <*machine_name*>_material.cfg file.
- Use one or more temporary materials.

Temporary materials are numbered automatically by QtPlasmaC and the material change will also be done by QtPlasmaC and should not be added to the G-code file by CAM software or otherwise. The material numbers begin at 1000000 and are incremented for each temporary material. It is not possible to save a temporary material, however the user could create a new material while a temporary material is displayed, and it will use the settings from the temporary material as the defaults.

sugerencia
It is possible to use temporary materials only and have an empty <*machine_name*>_material.cfg file. This negates the need to keep the QtPlasmaC materials file updated with the CAM tool file.

- The entire comment must be in parentheses.
- The beginning of the magic comment must be: **(o=**
- The equals sign must immediately follow each parameter with no space.
- The mandatory parameters must be in the magic comment (for option 0, **na** is optional and **nu** is not used).
- There can be any number and type of magic comments in a G-code file.
- If option 0 is to be used in addition to option 1 and/or option 2 then all option 0 must appear after all option 1 or all option 2 in the G-code file.

Las opciones son:

Option	Descripción
0	Creates a temporary default material. Material information added with this option will be discarded by a LinuxCNC restart or materials reload. They may also be overwritten by a new G-code file that has temporary materials.
1	Adds a new material if the number specified does not exist.
2	Overwrites an existing material if the number specified exists. Adds a new material if the number specified does not exist.

Mandatory parameters are:

Name	Descripción
o	Selects the option to be used.
nu	Sets the material number (not used for option 0).
na	Sets the material name (optional for option 0).
ph	Sets the pierce height.
pd	Sets the pierce delay.
ch	Sets the cut height.
fr	Sets the feed rate.

Optional parameters are:

Name	Descripción
mt	Sets the material thickness.
kw	Sets the kerf width.
th	Sets the THC status (0=disabled, 1=enabled).
ca	Sets the cut amps.
cv	Sets the cut voltage.
pe	Sets the pause at end delay.
gp	Sets the gas pressure (PowerMax).
cm	Sets the cut mode (PowerMax).
jh	Sets the puddle jump height.
jd	Sets the puddle jump delay.

A complete example (metric):

```
(o=0, nu=2, na=5mm Mild Steel 40A, ph=3.1, pd=0.1, ch=0.75, fr=3000, mt=5, kw=0.5, th=1, ca ↩
    =40, cv=110, pe=0.1, gp=5, cm=1, jh=0, jd=0)
```

A complete example (imperial):

```
(o=0, nu=2, na=0.197” Mild Steel 40A, ph=0.122, pd=0.1, ch=0.029, fr=118, mt=0.197, kw ↩
    =0.020, th=1, ca=40, cv=110, pe=0.1, gp=72, cm=1, jh=0, jd=0)
```

If a temporary material has been specified in a G-code file then the material change line (M190…) and wait for change line (M66…) will be added by the G-code filter and are not required in the G-code file.

10.8.9.10. Material Converter

This application is used to convert existing tool tables into QtPlasmaC material files. It can also create a material file from manual user input to entry fields.

At this stage the only conversions available are for tool tables exported from either SheetCam or Fusion 360.

SheetCam tool tables are complete, and the conversion is fully automatic. The SheetCam tool file must be in the SheetCam .tools format.

Fusion 360 tool tables do not have all of the required fields so the user will be prompted for missing parameters. The Fusion 360 tool file must be in the JSON format of Fusion 360.

If the user has a format from a different CAM software they would like converted, create a **New Topic** in the PlasmaC forum section of the LinuxCNC forum to request this addition.

El Convertidor de Material puede ejecutarse desde una terminal usando uno de los dos métodos siguientes.

For a package installation (Buildbot) enter the following command in a terminal window:

```
qtplasmac-materials
```

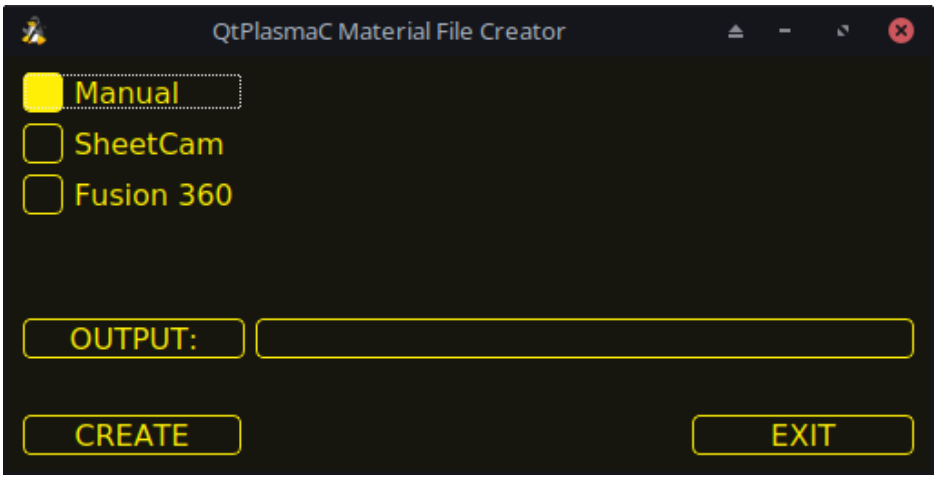
For a run in place installation enter the following two commands in a terminal window:

```
source ~/linuxcnc-dev/scripts/rip-environment
qtplasmac-materials
```

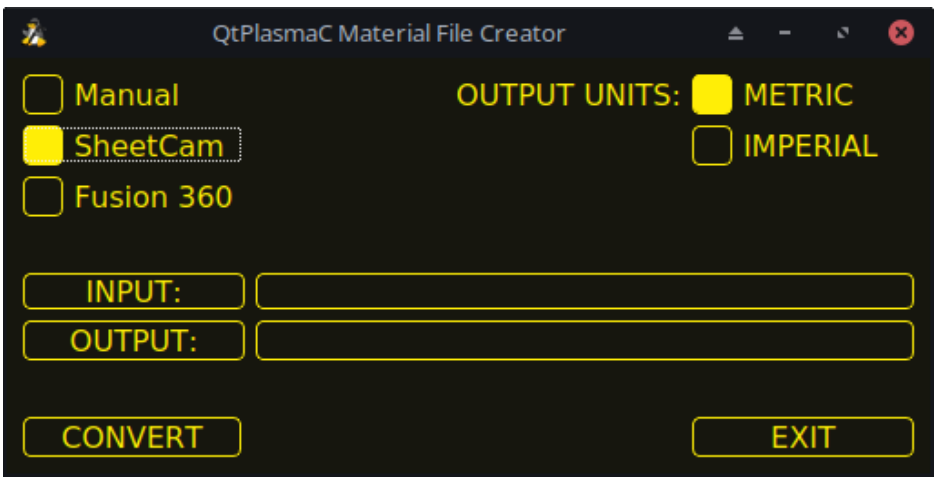
This will bring up the Material Converter Main dialog box with Manual selected as the default.

Seleccione uno de:

- **Manual** - to manually create a new material file.

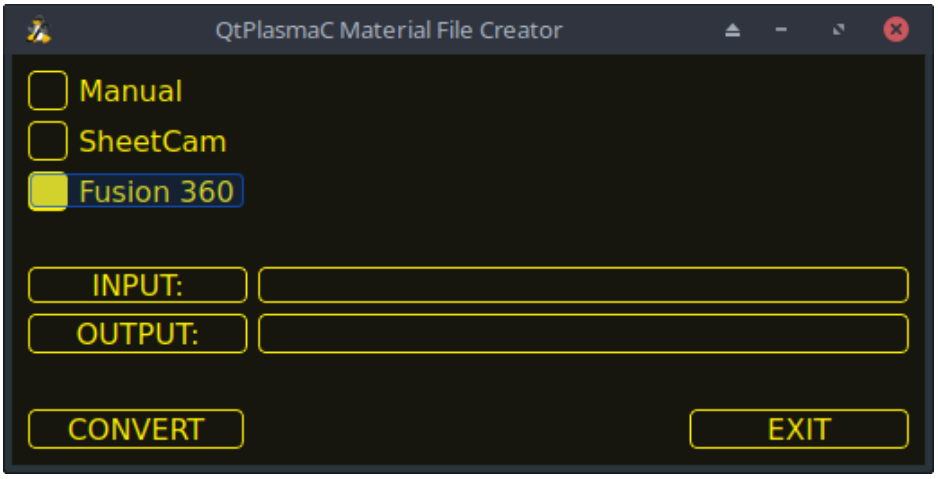


- **SheetCam** - to convert a SheetCam tool file.



For SheetCam only, select whether the user requires a metric or imperial output file.

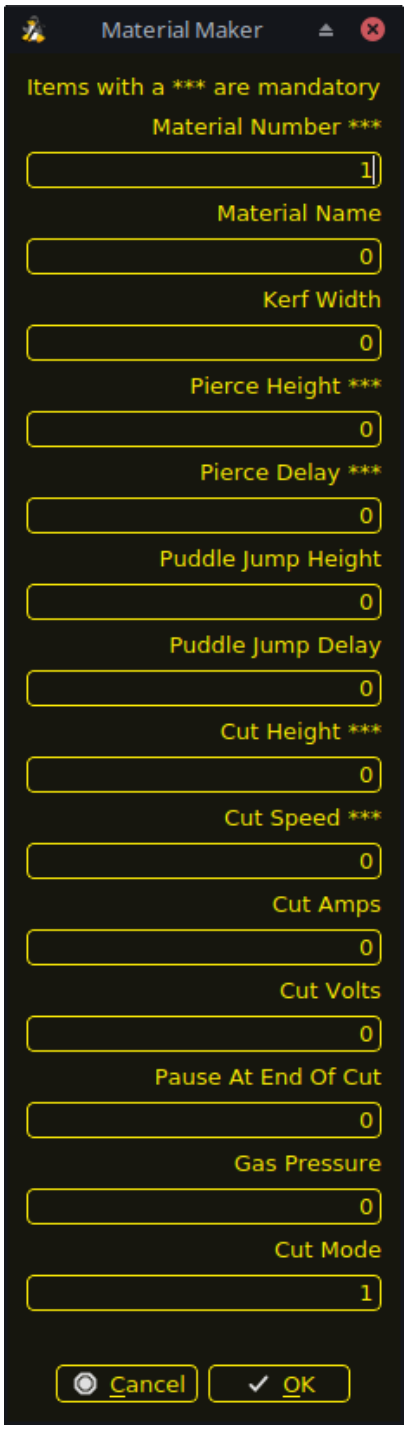
- **Fusion 360** - to convert a Fusion 360 tool file.



Para convertir:

1. Select the Input File to be converted, press **INPUT** to bring up a file selector or directly enter the file in the entry box.
2. Select the Output File to write to, press **OUTPUT** to bring up a file selector or directly enter the file in the entry box. This would normally be ~/linuxcnc/configs/<*machine_name*>_material.cfg. If necessary, the user could select a different file and hand edit the <*machine_name*>_material.cfg file.
3. Click **CREATE/CONVERT** and the new material file will be created.

For both a Manual creation or a Fusion 360 conversion, a dialog box will show with all available parameters displayed for input. Any entry marked with *** is mandatory and all other entries are optional depending on the user's configuration needs.



nota
If the user selects ~/linuxcnc/configs/<*machine_name*>_material.cfg and the file already exists, it will be overwritten.

10.8.9.11. LASER

QtPlasmaC has the ability to use a laser to set the origin with or without rotation compensation. Rotation compensation can be used to align the work offset to a sheet of material with edge(s) that

are not parallel to the machine's X/Y axes. The LASER button will be enabled after the machine is homed. This button will not be visible until a LASER offset is set in the <*machine_name*>.prefs file.

To use this feature, the user must set the laser's offset from the torch center by following the procedure described in Peripheral Offsets.

To modify the offsets manually, the user could edit either or both the following options in the **[LASER_OFFSET]** section of the <*machine_name*>.prefs file:

```
X axis = n.n
Y axis = n.n
```

where *n.n* is distance from the center line of the torch to the laser's cross hairs.

Additionally, the laser can be tied to any available output to turn the laser on and off via a HAL pin with the following name:

```
qtplasmac.laser_on
```

To set the origin with zero rotation:

1. Click the **LASER** button.
2. **LASER** button label will change to **MARK EDGE** and the HAL pin named qtplasmac.laser_on will be turned on.
3. Jog until the laser cross hairs are on top of the desired origin point.
4. Press **MARK EDGE**. The **MARK EDGE** button label will change to **SET ORIGIN**.
5. Press **SET ORIGIN**. The **SET ORIGIN** button label will change to **MARK EDGE** and the HAL pin named qtplasmac.laser_on will be turned off.
6. The torch will now move to the X0 Y0 position.
7. The offset is now successful set.

To set the origin with rotation:

1. Click the **LASER** button.
2. **LASER** button label will change to **MARK EDGE** and the HAL pin named qtplasmac.laser_on will be turned on.
3. Jog until the laser cross hairs are at the edge of the material a suitable distance away from the desired origin point.
4. Press **MARK EDGE**. The **MARK EDGE** button label will change to **SET ORIGIN**.
5. Jog until the laser cross hairs are at the origin point of the material.
6. Press **SET ORIGIN**. The **SET ORIGIN** button label will change to **MARK EDGE** and the HAL pin named qtplasmac.laser_on will be turned off.
7. The torch will now move to the X0 Y0 position.
8. The offset is now successfully set.

To turn the laser off and cancel an alignment:

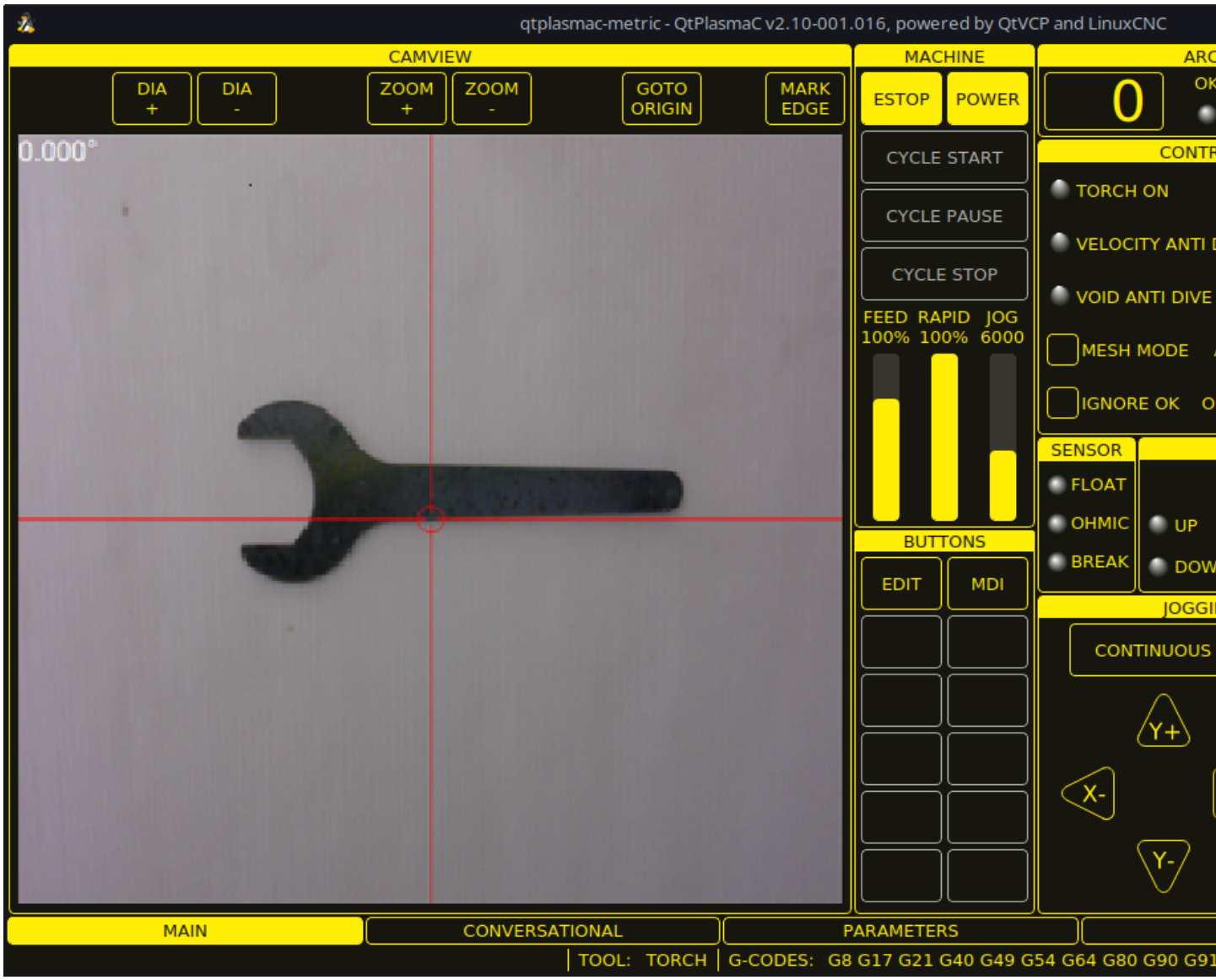
1. Press the **LASER** button and hold for longer than 750 ms.
2. **LASER** button label will change to **LASER** and the HAL pin named qtplasmac.laser_on will be turned off.

3. Release the **LASER** button.

If an alignment laser has been set up then it is possible to use the laser during CUT RECOVERY for accurate positioning of the new start coordinates.

To dry run the G-code file with the laser: . Ensure there are no bounds errors and CYCLE START is enabled. . Press the **LASER** button and hold for longer than 750 ms, the laser will turn on and the dry run will start. . Release the **LASER** button.

10.8.9.12. CAMERA



QtPlasmaC has the ability to use a USB camera to set the origin with or without rotation compensation. Rotation compensation can be used to align the work offset to a sheet of material with edge(s) that are not parallel to the machine's X/Y axes. The CAMERA button will be enabled after the machine is homed. This button will not be visible until a CAMERA offset is set in the <*machine_name*>.prefs file.

To use this feature, the user must set the camera's offset from the torch center by following the procedure described in Peripheral Offsets.

To modify the offsets manually, the user could edit either or both the following axes options in the **[CAMERA_OFFSET]** section of the <*machine_name*>.prefs file:

```
X axis = n.n
Y axis = n.n
Camera port = 0
```

where *n.n* is distance from the center line of the torch to the camera's cross hairs.

To set the origin with zero rotation:

1. Jog until the cross hairs are on top of the desired origin point.
2. Press **MARK EDGE**. The **MARK EDGE** button label will change to **SET ORIGIN** and the **GOTO ORIGIN** button will be disabled.
3. Press **SET ORIGIN**. The **SET ORIGIN** button label will change to **MARK EDGE** and the **GOTO ORIGIN** button will be enabled.
4. The torch will now move to the X0 Y0 position.
5. The offset is now successful set.

To set the origin with rotation:

1. Jog until the cross hairs are at the edge of the material a suitable distance away from the desired origin point.
2. Press **MARK EDGE**. The **MARK EDGE** button label will change to **SET ORIGIN** and the **GOTO ORIGIN** button will be disabled.
3. Jog until the cross hairs are at the origin point of the material.
4. Press **SET ORIGIN**. The **SET ORIGIN** button label will change to **MARK EDGE** and the **GOTO ORIGIN** button will be enabled.
5. The torch will now move to the X0 Y0 position.
6. The offset is now successfully set.

In the CAMVIEW panel, the mouse can affect the cross hairs, and the zoom level as follows:

- Mouse Wheel Scroll - Change cross hair diameter.
- Mouse Wheel Button Double Click - Restores cross hair diameter to default.
- Mouse Left Button Clicked + Wheel Scroll - Changes camera zoom level.
- Mouse Left Button Clicked + Wheel Button Double Click - Restores default camera zoom level.

10.8.9.13. Path Tolerance

Path tolerance is set with a G64 command and a following P value. The P value corresponds to the amount that the actual cut path followed by the machine may deviate from the programmed cut path.

The default LinuxCNC path tolerance is set for maximum speed which will severely round corners when used with normal plasma cutting speeds.

It is recommended that the path tolerance is set by placing the appropriate G64 command and P value in the header of each G-code file.

The provided G-code filter program will test for the existence of a `G64 P__n__` command prior to the first motion command. If no G64 command is found it will insert a `G64 P0.1` command which sets the path tolerance to 0.1 mm. For a imperial config the command will be `G64 P0.004`.

For Metric:

```
G64 P0.1
```

For Imperial:

```
G64 P0.004
```

10.8.9.14. Paused Motion

QtPlasmaC has the ability to allow the repositioning of the X and Y axes along the current cut path while the G-code program is paused, aiding in Cut Recovery.

In order to use this feature, LinuxCNC's Adaptive Feed Control (M52) must be turned on (P1). This is also a requirement for Hole Cutting Velocity Reduction.

To enable **Paused Motion** The preamble of the G-code must contain the following line:

```
M52 P1
```

To turn off **Paused Motion** at any point, use the following command:

```
M52 P0
```

10.8.9.15. Pause At End Of Cut

This feature can be used to allow the arc to "catch up" to the torch position to fully finish the cut. It is usually required for thicker materials and is especially useful when cutting stainless steel.

Using this feature will cause all motion to pause at the end of the cut while the torch is still on. After the dwell time (in seconds) set by the **Pause At End** parameter in the MATERIAL section of the PARAMETERS Tab has expired, QtPlasmaC will proceed with the M5 command to turn off and raise the torch.

10.8.9.16. Multiple Tools

QtPlasmaC has the ability to allow the use of more than one type of plasma tool by utilizing LinuxCNC spindles as a plasma tool when running a G-code program.

Valid plasma tools for use are:

Name	TOOL #	Descripción
Plasma Torch	0	Usado para corte por plasma normal.
Scribe	1	Used for material engraving.
Plasma Torch	2	Used for spotting (creating dimples to aid in drilling).

A LinuxCNC spindle number (designated by $\$n$) is required to be in the starting command and also the end command to be able to start and stop the correct plasma tool. Examples:

- `M3 $0 S1` will select and start the plasma cutting tool.
- `M3 $1 S1` will select and start the scribe.
- `M3 $2 S1` will select and start the plasma spotting tool.
- `M5 $0` will stop the plasma cutting tool.
- `M5 $1` will stop the scribe.

- `M5 $2` will stop the plasma spotting tool.

It is permissible to use **M5 $-1** in lieu of the M5 $*n* codes above to stop all tools.

In order to use a scribe, it is necessary for the user to add the X and Y axis offsets to the LinuxCNC tool table. Tool 0 is assigned to the Plasma Torch and Tool 1 is assigned to the scribe. Tools are selected with a **T*n* M6** command, and then a **G43 H0** command is required to apply the offsets for the selected tool. It is important to note that the LinuxCNC tool table and tool commands only come into play if the user is using a scribe in addition to a plasma torch. For more information, see scribe.

10.8.9.17. Velocity Reduction

There is a HAL pin available named **motion.analog-out-03** that can be changed in G-code with the **M67 (Synchronized with Motion)/M68 (Immediate)** commands. This pin will reduce the velocity of the original feed rate to the percentage specified in the command.

The ”VEL:” label at the top right of the preview window will update to reflect the percentage of the original feed rate being used. For example, ”VEL@20 %:” means the table is cutting at 20 % of the programmed feed rate - an 80 % reduction.

---

nota
Because of differences between the GUI and the PlasmaC component polling intervals, velocity label updates may lag (typically by up to 100 ms).

It is important to thoroughly understand the difference between **Synchronized with Motion** and **Immediate**:

- `M67` (Synchronized with Motion) - The actual change of the specified output (P2 (THC) for example) will happen at the beginning of the next motion command. If there is no subsequent motion command, the output changes will not occur. It is best practice to program a motion code (G0 or G1 for example) right after a M67.
- `M68` (Immediate) - These commands happen immediately as they are received by the motion controller. Since these are not synchronized with motion, they will break blending. This means if these codes are used in the middle of active motion codes, the motion will pause to activate these commands.

Examples:

- `M67 E3 Q0` would set the velocity to 100 % of **CutFeedRate**.
- `M67 E3 Q40` would set the velocity to 40 % of **CutFeedRate**.
- `M67 E3 Q60` would set the velocity to 60 % of **CutFeedRate**.
- `M67 E3 Q100` would set the velocity to 100 % of **CutFeedRate**.

Q values that are less than or equal to 0 or greater than or equal to 100 will be set to 100.

If the user intends to use this feature it would be prudent to add `M68 E3 Q0` to both the preamble and postamble of the G-code program so the machine starts and ends in a known state.



importante
G-CODE THC AND **VELOCITY BASED THC** ARE NOT ABLE TO BE USED IF **CUTTER COMPENSATION** IS IN EFFECT; AN ERROR MESSAGE WILL BE DISPLAYED.



aviso
If Cut Feed Rate in the MATERIAL section of the PARAMETERS Tab is set to zero, then QtPlasmaC will use **motion.requested-velocity** (as set by a standard feed rate call in the G-code) for the THC calculations. This is not recommended as it is not a reliable way of implementing velocity-based THC.

nota
All references to CutFeedRate refer to the **Cut Feed Rate** value displayed in the MATERIAL section of the PARAMETERS Tab.

10.8.9.18. THC (Torch Height Controller)

The THC can be enabled or disabled from the THC frame of the MAIN Tab.

The THC can also be enabled or disabled directly from the G-code program.

The THC does not become active until the velocity reaches 99.9 % of the **CutFeedRate** and then the THC **Delay** time if any in the THC section of the PARAMETERS Tab has timed out. This is to allow the arc voltage to stabilize.

QtPlasmaC uses a control voltage which is dependent on the state of the **AUTO VOLTS** checkbox on the MAIN Tab:

1. If **Use Auto Volts** is checked, then the actual cut voltage is sampled at the end of the THC **Delay** time and this is used as the target voltage to adjust the height of the torch.
2. If **Use Auto Volts** is not checked then the voltage displayed as Cut Volts in the MATERIAL section of the PARAMETERS Tab is used as the target voltage to adjust the height of the torch.

G-code THC THC may be disabled and enabled directly from G-code, provided the THC is not disabled in the THC Section of the MAIN Tab, by setting or resetting the **motion.digital-out-02** pin with the M-Codes M62-M65:

- `M62 P2` deshabilitará THC (Sincronizada con movimiento)
- `M63 P2` will enable THC (Synchronized with Motion)
- `M64 P2` deshabilitará THC (Inmediatamente)
- `M65 P2` will enable THC (Immediately)

It is important to thoroughly understand the difference between **Synchronized with Motion** and **Immediate**:

- `M62` and `M63` (Synchronized with Motion) - The actual change of the specified output (P2 (THC) for example) will happen at the beginning of the next motion command. If there is no subsequent motion command, the output changes will not occur. It is best practice to program a motion code (G0 or G1 for example) right after a M62 or M63.
- `M64` and `M65` (Immediate) - These commands happen immediately as they are received by the motion controller. Since these are not synchronized with motion, they will break blending. This means if these codes are used in the middle of active motion codes, the motion will pause to activate these commands.

THC basado en velocidad

If the cut velocity falls below a percentage of **CutFeedRate** (as defined by the VAD Threshold % value in the THC frame of the CONFIGURATION section of the PARAMETERS Tab) the THC will be locked until the cut velocity returns to at least 99.9 % of **CutFeedRate**. This will be made apparent by the **VELOCITY ANTI DIVE** indicator illuminating in the CONTROL Panel on the MAIN Tab.

Velocity based THC prevents the torch height being changed when velocity is reduced for a sharp corner or a small hole.

It is important to note that Velocity Reduction affects the Velocity Based THC in the following ways:

1. If Velocity Reduction is invoked in the middle of the cut, the THC will be locked.
2. The THC will remain locked until the velocity reduction is canceled by returning it to a value that is above the **VAD Threshold**, and the torch actually reaches 99.9 % of the **CutFeedRate**.

10.8.9.19. Cutter Compensation

LinuxCNC (QtPlasmaC) has the ability to automatically adjust the cut path of the current program by the amount specified in Kerf Width of the selected material's Cut Parameters. This is helpful if the G-code is programmed to the nominal cut path and the user will be running the program on different thickness materials to help ensure consistently sized parts.

To use cutter compensation the user will need to use G41.1, G42.1 and G40 with the kerf width HAL pin:

- `G41.1 D#<_hal[plasmac.kerf-width]>` : offsets torch to the left of the programmed path
- `G42.1 D#<_hal[plasmac.kerf-width]>` : offsets torch to the right of the programmed path
- `G40` desactiva la compensación del cortador



importante
IF **CUTTER COMPENSATION** IS IN EFFECT **G-CODE THC**, **VELOCITY BASED THC** AND **OVER CUT** ARE NOT ABLE TO BE USED; AN ERROR MESSAGE WILL BE DISPLAYED.

10.8.9.20. Initial Height Sense (IHS) Skip

Initial Height Sense may be skipped in one of two different ways:

1. If the THC is disabled, or the THC is enabled but not active, then the IHS skip will occur if the start of the cut is less than **Skip IHS** distance from the last successful probe.
2. If the THC is enabled and active, then the IHS skip will occur if the start of the cut is less than **Skip IHS** distance from the end of the last cut.

A value of zero for **Skip IHS** will disable all IHS skipping.

Any errors encountered during a cut will disable IHS skipping for the next cut if **Skip IHS** is enabled.

10.8.9.21. Probing

Probing may be done with either ohmic sensing or a float switch. It is also possible to combine the two methods, in which case the float switch will provide a fallback to ohmic probing. An alternative to ohmic probing is Offset Probing

If the machine's torch does not support ohmic probing, the user could have a separate probe next to the torch. In this case the user would extend the probe below the torch. The probe must NOT extend more than the minimum Cut Height below the torch and the Z axis offset distance needs to be entered as the **Ohmic Offset** in the PROBING frame of the CONFIGURATION section of the PARAMETERS Tab.

Probing setup is done in the PROBING frame of the CONFIGURATION section of the PARAMETERS Tab.

QtPlasmaC can probe at the full Z axis velocity so long as the machine has enough movement in the float switch to absorb any overrun. If the machine's float switch travel is suitable, the user could set the Probe Height to near the Z axis MINIMUM_LIMIT and do all probing at full speed.

Some float switches can exhibit a large switching hysteresis which shows up in the probing sequence as an excessive time to complete the final probe up.

- This time may be decreased by changing the speed of the final probe up.
- This speed defaults to 0.001 mm (0.000039") per servo cycle.
- It is possible to increase this speed by up to a factor of 10 by adding the following line to the custom.hal file:

```
setp plasmac.probe-final-speed n
```

where *n* is a value from 1-10. It is recommended to keep this value as low as possible.

Using this feature will change the final height slightly and will require thorough probe testing to confirm the final height.

This speed value affects ALL probing so if the user uses ohmic probing and the user changes this speed value then the user will need to probe test to set the require offset to compensate for this speed change as well as the float travel.

The reliability of this feature will only be as good as the repeatability of the float switch.

---

nota
Probe Height refers to the height above the Z axis MINIMUM_LIMIT.

10.8.9.22. Offset Probing

Offset Probing is the use of a probe that is offset from the torch. This method is an alternative to Ohmic Probing and uses the `plasmac.ohmic-enable` output pin to operate a solenoid for extending and retracting the probe. The `plasmac.ohmic-probe` input pin is used to detect the material and the **Ohmic Offset** in the PROBING frame of the CONFIGURATION section of the PARAMETERS Tab is used to set the correct measured height.

The probe could be a mechanically deployed probe, a permanently mounted proximity sensor or even simply a stiff piece of wire extending about 0.5 mm (0.2") below the torch tip. If the probe is mechanically deployed, then it needs to extend/retract rather quickly to avoid excessive probing times and would commonly be pneumatically operated.

To use this feature, the user must set the probe's offset from the torch center by following the procedure described in Peripheral Offsets.

To modify the offsets manually, the user could edit either or both the following options in the **[OFFSET_PROBING]** section of the <*machine_name*>.prefs file:

```
X axis = n.n
Y axis = n.n
Delay = t.t
```

where *n.n* is the offset of the probe from the torch center in machine units for the X and Y axes and *t.t* is the time in seconds to allow for any mechanical deployment of the probe if required.

Each of these parameters is optional and also may appear in any order. If a parameter is not detected, then the default is 0.0. There can be no space after the X or Z, lower case is permissible.

When this variable appears in the <*machine_name*>.prefs file with either X or Y not equal to zero, then QtPlasmaC will do **all** Ohmic Probing as Offset Probing. If Offset Probing is valid then the feed rate at which the X and Y axes move to the offset position may be adjusted by the use of the **Offset Speed** parameter in the PROBING frame of the PARAMETERS Tab.

When a probe sequence has begun, the `plasmac.ohmic-enable` pin will be set True causing the probe to extend. When the material is detected the `plasmac.ohmic-enable` pin will be reset to false causing the probe to retract.

The probe will begin moving to the offset position simultaneously with the Z axis moving down to the Probe Height, probing will not commence unless the deployment timer has completed. It is required that the **Probe Height** in the PROBING frame of the CONFIGURATION section of the PARAMETERS Tab is above the top of the material to ensure that the probe is fully offset to the correct X/Y position before the final vertical probe down movement.



importante
PROBE HEIGHT NEEDS TO BE SET ABOVE THE TOP OF THE MATERIAL FOR OFFSET PROBING.

10.8.9.23. Cut Types

QtPlasmaC allows two different cut modes:

1. **NORMAL CUT** - runs the loaded G-code program to pierce then cut.
2. PIERCE ONLY - only pierces the material at each cut start position, useful prior to a **NORMAL CUT** on thick materials

There are two ways of enabling this feature:

1. Utilize the default custom user button to toggle between the cut types.
2. Adding the following line to the G-code program before the first cut to enable **Pierce Only** mode for the current file:

```
#<pierce-only> = 1
```

If using a custom user button is utilized then QtPlasmaC will automatically reload the file when the cut type is toggled.

10.8.9.24. Hole Cutting - Intro

It is recommended that any holes to be cut have a diameter no less than one and a half times the thickness of the material to be cut.

It is also recommended that holes with a diameter of less than 32 mm (1.26”) are cut at 60 % of the feed rate used for profile cuts. This should also lock out THC due to velocity constraints.

QtPlasmaC can utilize G-code commands usually set by a CAM Post Processor (PP) to aid in hole cutting or if the user does not have a PP or the user’s PP does not support these methods then QtPlasmaC can automatically adapt the G-code to suit. This automatic mode is disabled by default.

There are three methods available for improving the quality of small holes:

1. **Velocity Reduction** - Reducing the velocity to approximately 60 % of the **CutFeedRate**.
2. **Arc Dwell (Pause At End)** - Keeping the torch on for a short time at the end of the hole while motion is stopped to allow the arc to catch up.
3. **Over cut** - Turning the torch off at the end of the hole then continue along the path.

---

nota
If both **Arc Dwell** and **Over cut** are active at the same time, then **Over cut** will take precedence.



importante
OVER CUT IS NOT ABLE TO BE USED IF CUTTER COMPENSATION IS IN EFFECT; AN ERROR MESSAGE WILL BE DISPLAYED.

10.8.9.25. Hole Cutting

G-code commands can be set up by either by a CAM Post Processor (PP) or by hand coding.

Hole Cutting Velocity Reduction

If cutting a hole requires a reduced velocity, then the user would use the following command to set the velocity: `M67 E3 Qnn` where `nn` is the percentage of the velocity desired. For example, `M67 E3 Q60` would set the velocity to 60 % of the current material’s **CutFeedRate**.

In order to use this feature, LinuxCNC’s Adaptive Feed Control (M52) must be turned on (P1). This is also a requirement for Paused Motion during Cut Recovery.

To enable **Hole Cutting Velocity Reduction** The preamble of the G-code must contain the following line:

```
M52 P1
```

To turn off **Hole Cutting Velocity Reduction** at any point, use the following command:

```
M52 P0
```

See the Velocity Based THC section.

Sample code for hole cutting with reduced velocity.

```
G21 (metric)
G64 P0.005
M52 P1 (enable adaptive feed)
F#<_hal[plasmac.cut-feed-rate]> (feed rate from cut parameters)
G0 X10 Y10
M3 $0 S1 (start cut)
G1 X0
M67 E3 Q60 (reduce feed rate to 60%)
G3 I10 (the hole)
M67 E3 Q0 (restore feed rate to 100%)
M5 $0 (end cut)
G0 X0 Y0
M2 (end job)
```

Arc Dwell (Pause At End) This method can be invoked by setting the Pause At End parameter in the MATERIAL frame of the PARAMETERS Tab.

Over cut

The torch can be turned off at the end of the hole by setting the `motion.digital-out-03` pin with the M-Codes `M62` (Synchronized with Motion)* or `M64` (Immediate). After turning the torch off it is necessary to allow the torch to be turned on again before beginning the next cut by resetting the `motion.digital-out-03` pin with the M-Codes `M63` or `M65`, this will be done automatically by the QtPlasmaC G-code parser if it reaches an M5 command without seeing a `M63 P3` or `M65 P3`.

After the torch is turned off the hole path will be followed for a default length of 4 mm (0.157”). This distance may be specified by adding `#<oclength>` = *n* to the G-code file.

- `M62 P3` apagará la antorcha (Sincronizado con movimiento)
- `M63 P3` permitirá encender la antorcha (Sincronizado con movimiento)
- `M64 P3` apagará la antorcha (Inmediatamente)
- `M65 P3` will allow the torch to be turned on (Immediately)

It is important to thoroughly understand the difference between **Synchronized with motion** and **Immediate**:

- `M62 and M63` (Synchronized with Motion) - The actual change of the specified output (P2 (THC) for example) will happen at the beginning of the next motion command. If there is no subsequent motion command, the output changes will not occur. It is best practice to program a motion code (G0 or G1 for example) right after a M62 or M63.
- `M64 and M65` (Immediate) - These commands happen immediately as they are received by the motion controller. Since these are not synchronized with motion, they will break blending. This means if these codes are used in the middle of active motion codes, the motion will pause to activate these commands.

Código de muestra:

```
G21 (metric)
G64 P0.005
M52 P1 (enable adaptive feed)
F#<_hal[plasmac.cut-feed-rate]> (feed rate from cut parameters)
G0 X10 Y10
M3 $0 S1 (start cut)
G1 X0
M67 E3 Q60 (reduce feed rate to 60%)
G3 I10 (the hole)
M62 P3 (turn torch off)
```

```
G3 X0.8 Y6.081 I10 (continue motion for 4 mm)
M63 P3 (allow torch to be turned on)
M67 E3 Q0 (restore feed rate to 100%)
M5 $0 (end cut)
G0 X0 Y0
M2 (end job)
```

10.8.9.26. Hole Cutting - Automatic

QtPlasmaC has the ability to automatically modify the G-code to reduce the velocity and/or apply **Over cut** which can be useful when cutting holes.

For valid hole sensing it is required that all values in the G2 or G3 G-code line are explicit, an error dialog will be displayed if any values are mathematically calculated.

QtPlasmaC Hole Sensing is disabled by default. It can be enabled/disabled by using the following G-code parameters to select the desired hole sensing mode:

- `#<holes> = 0` - Causes QtPlasmaC to disable hole sensing if it was previously enabled.
- `#<holes> = 1` - Causes QtPlasmaC to reduce the speed of holes less than 32 mm (1.26”) to 60 % of **CutFeedRate**.
- `#<holes> = 2` - Causes QtPlasmaC to Over cut the hole in addition to the velocity changes in setting 1.
- `#<holes> = 3` - Causes QtPlasmaC to reduce the speed of holes less than 32 mm (1.26”) and arcs less than 16 mm (0.63”) to 60 % of **CutFeedRate**.
- `#<holes> = 4` - Causes QtPlasmaC to Over cut the hole in addition to the velocity change in setting 3.

The default hole size for QtPlasmaC hole sensing is 32 mm (1.26”). It is possible to change this value with the following command in a G-code file:

- `#<h_diameter>` = *nn* - To set a diameter (*nn*) in the same units system as the rest of the G-code file.

The default velocity for QtPlasmaC small holes is 60 % of the current feed rate. It is possible to change this value with the following command in a G-code file:

- `#<h_velocity>` = *nn* - to set the percentage (*nn*) of the current feed rate required.

Over cut If Hole Sensing modes 2 or 4 are active, QtPlasmaC will over cut the hole in addition to the velocity changes associated with modes 1 and 3.

The default over cut length for QtPlasmaC hole sensing is 4 mm (0.157”). It is possible to change this value with the following command in a G-code file:

- `#<oclength>` = *nn* to specify an over cut length (*nn*) in the same units system as the rest of the G-code file.

Arc Dwell (Pause At End) This feature can be used in addition to setting the desired hole sensing mode via the appropriate G-code parameter by setting the Pause At End parameter in the MATERIAL frame of the PARAMETERS Tab.

Código de muestra:

```
G21 (metric)
G64 P0.005
M52 P1 (enable adaptive feed)
F#<_hal[plasmac.cut-feed-rate]> (feed rate from cut parameters)
#<holes> = 2 (over cut for holes)
#<oclength> = 6.5 (optional, 6.5 mm over cut length)
G0 X10 Y10
M3 $0 S1 (start cut)
G1 X0
G3 I10 (the hole)
M5 $0 (end cut)
G0 X0 Y0
M2 (end job)
```

nota
It is OK to have multiple and mixed hole commands in a single G-code file.

10.8.9.27. Single Cut

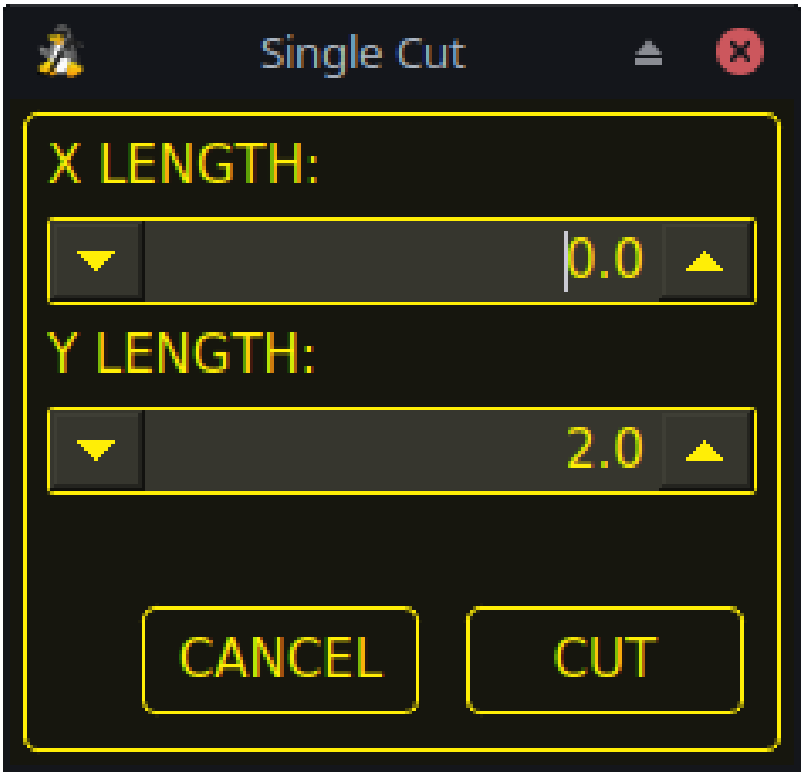
A single cut is a single unidirectional cutting move often used to cut a sheet into smaller pieces prior to running a G-code program.

The machine needs to be homed before commencing a single cut.

A single cut will commence from the machine's current X/Y position.

Automatic Single Cut

This is the preferred method. The parameters for this method are entered in the following dialog box that is displayed after pressing a user button which has been coded to run single cut:



1. Jog to the required X/Y start position.
2. Set required appropriate material, or edit the Feed Rate for the default material in the PARAMETERS Tab.
3. Press the assigned single cut user button.
4. Enter the length of the cut along the X and/or Y axes.
5. Press the **CUT** button and the cut will commence.

Pendant Single Cut If the machine is equipped with a pendant that can start and stop the spindle plus jog the X and Y axes, the user can manually perform a single cut.

1. Jog to the required X/Y start position.
2. Set the required feed rate with the Jog Speed slider.
3. Start the cut process by starting the spindle.
4. After probing the torch will fire.
5. When the Arc OK is received the machine can be jogged along the cut line using the jog buttons.
6. When the cut is complete stop the spindle.
7. The torch will turn off and the Z axis will return to the starting position.

Manual Single Cut

Manual single cut requires that either keyboard shortcuts are enabled in the GUI SETTINGS section of the SETTINGS Tab, or a custom user button is specified as a manual cut button.

If the user is using a custom user button then, substitute **F9** with **User Button** in the following description.

1. Jog to the required X/Y start position.
2. Start the procedure by pressing **F9**. The jog speed will be automatically set to the feed rate of the currently selected material. The jog label will blink to indicate that the jog speed is temporarily being overridden (jog speed manipulation will be disabled while a manual cut is active). **CYCLE START** will change to **MANUAL CUT** and blink.
3. After probing the torch will fire.
4. When the Arc OK is received the machine can be jogged along the cut line using the jog keys.
5. The Z height will remain locked at the cut height for the duration of the manual cut, regardless of the Torch Height Controller **ENABLE** status.
6. When the cut is complete press **F9** or **Esc** or the **CYCLE STOP** button.
7. The torch will turn off and the Z axis will return to the starting position.
8. The jog speed will automatically be returned to the value it was prior to initiating the manual cut process, the label will stop blinking and the jog speed manipulation will be enabled. **MANUAL CUT** will stop blinking and revert to **CYCLE START**.

---

nota
If the torch flames out during cutting, the user must still press **F9** or **Esc** or the **CYCLE STOP** button to end the cut. This clears the Z offsets and returns the torch to the starting position.

10.8.9.28. Thick Materials

Cutting thick materials can be problematic in that the large amount of molten metal caused by piercing can shorten the life of consumables and also may cause a puddle high enough that the torch may hit the puddle while moving to cut height.

There are several functions built into QtPlasmaC to help alleviate these issues, Pierce Only and Puddle Jump described in this section as well as Wiggle Pierce and Ramp Pierce described in the Moving Pierce section.

Pierce Only

Pierce Only mode converts the loaded G-code program and then runs the program to pierce the material at the start position of each cut. Scribe and Spotting commands will be ignored, and no pierce will take place in those locations.

This mode is useful for thick materials which may produce enough dross on the material surface from piercing to interfere with the torch while cutting. The entire sheet can pierced, and then cleaned off prior to cutting.

It is possible to use near-end-of-life consumables for piercing and then they can be swapped out for good consumables to be used while cutting.

The pierce location during **Pierce Only** mode may be offset in the X and/or Y axes to ensure that the arc is able to transfer correctly when piercing after returning to the **Normal Cut** mode. The parameters for the X and Y Offsets are in the PIERCE ONLY frame of the CONFIGURATION section of the PARAMETERS Tab

Pierce Only is one of two different cut types

Puddle Jump Puddle Jump is the height that the torch will move to after piercing and prior to moving to **Cut Height** and is expressed as a percentage of **Pierce Height**. This allows the torch to clear any puddle of molten material that may be caused by piercing. The maximum allowable height is 200 % of the **Pierce Height**

Setting for **Puddle Jump** are described in cut parameters

The recommended option is to use **Pierce Only** due to it being able to utilise near end-of-life consumables.



importante
PUDDLE JUMP IS DISABLED DURING CUT RECOVERY

10.8.9.29. Mesh Mode (Expanded Metal Cutting)

QtPlasmaC is capable of cutting of expand (mesh) metal provided the machine has a pilot arc torch and it is capable of Constant Pilot Arc (CPA) mode.

Mesh Mode disables the THC and also ignores a lost Arc OK signal during a cut. It can be selected by checking the **Mesh Mode** check button in the CONTROL section of the MAIN Tab.

If the machine has RS485 communications enabled with a Hypertherm PowerMax plasma cutter, selecting **Mesh Mode** will automatically override the **Cut Mode** for the currently selected material and set it to cut mode 2 (CPA). When **Mesh Mode** is disabled, the **Cut Mode** will be return to the default cut mode for the currently selected material.

It is also possible to start a **Mesh Mode** cut without receiving an Arc OK signal by checking the **Ignore Arc OK** check button in the CONTROL section of the MAIN Tab.

Both **Mesh Mode** and **Ignore Arc OK** can be enabled/disabled at any time during a job.

10.8.9.30. Ignore Arc OK

Ignore Arc OK mode disables the THC, will begin a cut without requiring an Arc OK signal, and will ignore a lost Arc OK signal during a cut.

This mode can be selected by:

1. Checking the **Ignore Arc OK** check button in the CONTROL section of the MAIN Tab.
2. Setting HAL pin **motion.digital-out-01** to 1 via G-code.
   - `M62 P1` will enable **Ignore Arc OK** (Synchronized with Motion)
   - `M63 P1` deshabilitará **Ignore Arc OK** (Sincronizado con movimiento)
   - `M64 P1` will enable **Ignore Arc OK** (Immediately)
   - `M65 P1` will disable **Ignore Arc OK** (Immediately)

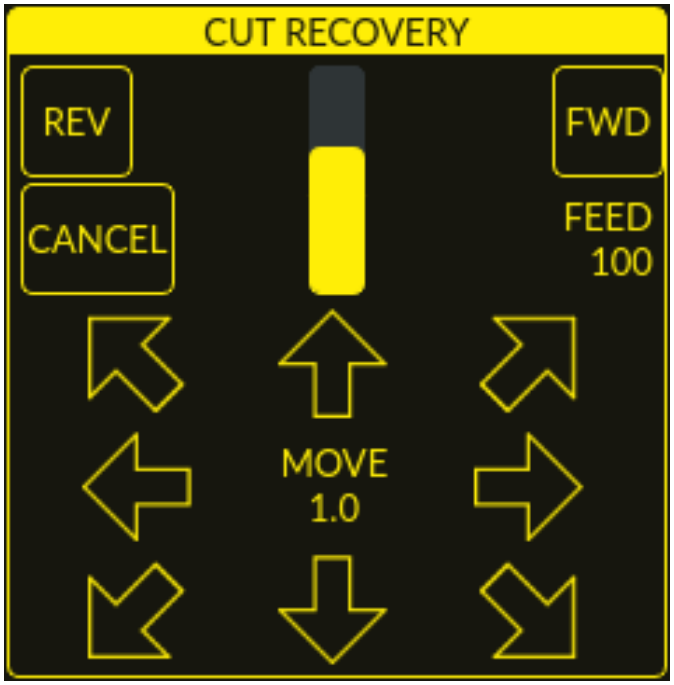
It is important to thoroughly understand the difference between **Synchronized with motion** and **Immediate**:

- `M62` and `M63` (Synchronized with Motion) - The actual change of the specified output (P2 (THC) for example) will happen at the beginning of the next motion command. If there is no subsequent motion command, the output changes will not occur. It is best practice to program a motion code (G0 or G1 for example) right after a M62 or M63.
- `M64` and `M65` (Immediate) - These commands happen immediately as they are received by the motion controller. Since these are not synchronized with motion, they will break blending. This means if these codes are used in the middle of active motion codes, the motion will pause to activate these commands.

This mode may also be used in conjunction with **Mesh Mode** if the user doesn't require an Arc OK signal to begin the cut.

Both **Mesh Mode** and **Ignore Arc OK** can be enabled/disabled at any time during a job.

10.8.9.31. Cut Recovery



This feature will produce a CUT RECOVERY panel that will allow the torch to be moved away from the cut path during a paused motion event in order to position the torch over a scrap portion of the material being cut so that the cut restarts with a minimized arc-divot. The CUT RECOVERY panel will display automatically over top of the JOGGING panel when motion is paused.

It is preferable to make torch position adjustments from the point at which paused motion occurred, however if moving along the cut path is necessary prior to setting the new start point, the user may use the paused motion controls (**REV**, **FWD**, and a **JOG-SPEED** slider) at the top of the CUT RECOVERY panel. Once the user is satisfied with the positioning of the torch along the cut path, moving off the cut path is achieved by pressing the **DIRECTION** buttons. Each press of the **DIRECTION** button will move the torch a distance equivalent to the **Kerf Width** parameter of the currently selected material.

The moment the torch has been moved off the cut path, the paused motion controls (**REV**, **FWD**, and a **JOG-SPEED** slider) at the top of the CUT RECOVERY panel will become disabled.

Once the torch position is satisfactory, press **CYCLE RESUME** and the cut will resume from the new position and travel the shortest distance to the original paused motion location. The CUT RECOVERY panel will close, and the JOGGING panel will display when the torch returns to the original paused motion location.

Pressing **CANCEL MOVE** will cause the torch to move back to where it was positioned before the direction keys were used to offset the torch. It will not reset any **REV** or **FWD** motion.

Pressing **CYCLE STOP** will cause the torch to move back to where it was positioned before the direction keys were used to offset the torch and the CUT RECOVERY panel overlay will return to the JOGGING panel. It will not reset any **REV** or **FWD** motion.

If an alignment laser has been set up then it is possible to use the laser during cut recovery for very accurate positioning of the new start coordinates. If either the X axis offset or Y axis offset for the laser would cause the machine to move out of bounds, then an error message will be displayed.

To use a laser for cut recovery when paused during a cut:

1. Click the **LASER** button.
2. **LASER** button will change to disabled, the HAL pin named qtplasmac.laser_on will be turned on and the X and Y axis will offset so that the laser cross hairs will indicate the starting coordinates of the cut when it is resumed.
3. Continue the cut recovery as described above.

If a laser offset is in effect when **CANCEL MOVE** is pressed, then this offset will also be cleared.

---

nota
Cut recovery movements will be limited to a radius of 10 mm (0.4”) from either the point the program was paused, or from the last point on the cut path if paused motion was used.



importante
PUDDLE JUMP IS DISABLED DURING CUT RECOVERY

10.8.9.32. Run From Line

If the user has the Run From Line option enabled in the GUI SETTINGS section of the SETTINGS Tab then they will have the ability to start from any line in a G-code program via the following methods:

1. Clicking any line in the Preview Window

2. Clicking any line in the G-code Window

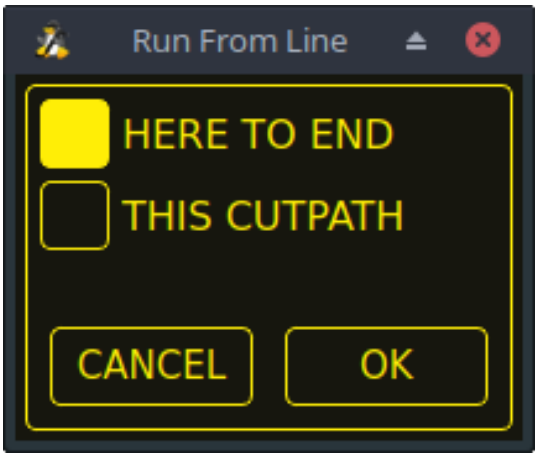
It is important to note that G-code programs can be run from any selected line using this method, however a lead-in may not be possible depending on the line selected. In this case, an error message will be displayed to let the user know the lead-in calculation was not possible.

Once the user has selected the starting place, the **CYCLE START** button will blink **”SELECTED *nn*”** where *nn* is the corresponding line number selected. Clicking this button will bring up the following Run From Line dialog box:

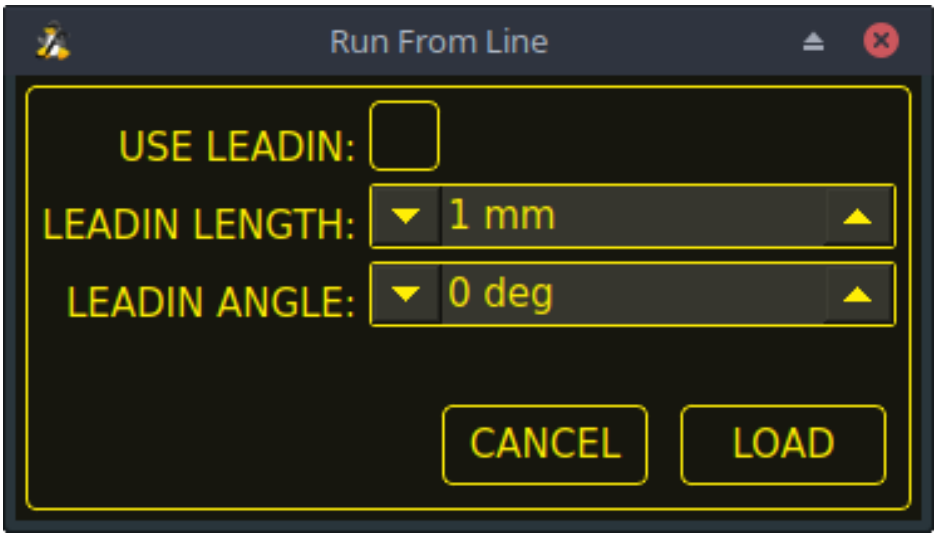
It is not possible to use Run From Line from within a subroutine. If the user selects a line within a subroutine and clicks **”SELECTED *nn*”** then an error message will be displayed that includes the O-code name of the subroutine.

It is not possible to use Run From Line if previous G-code has set cutter compensation active. If the user selects a line while cutter compensation is active and clicks **”SELECTED *nn*”** then an error message will be displayed.

It is possible to select a new line while Run From Line is active.



HERE TO END will run from the beginning of the selected line to the end of the G-code file. The user will be presented with the option of adding a lead-in if the selected line falls within an ”active” cutting operation (between an M3 and M5).



Name	Descripción
USE LEADIN	This radio button will allow the user to start the selected line with a lead-in.
LEADIN LENGTH	If USE LEADIN is selected, this will set the length of the lead in the machine units.
LEADIN ANGLE	If USE LEADIN is selected, this will set the angle of approach for the lead-in. The angle is measured such that positive increases in value move the lead-in counterclockwise: 0 Degrees = 3 o'clock position 90 Degrees = 12 o'clock position 180 Degrees = 9 o'clock position 270 Degrees = 6 o'clock position
CANCEL	This button will cancel the Run From Line dialog box and any selections.
LOAD	This button will load a temporary "rfl.ngc" program with any selected lead-in parameters applied. If the lead-in cannot be calculated for the selected line, the following error message will be displayed: "Unable to calculate a lead-in for this cut Program will run from selected line with no lead-in applied"

After pressing **LOAD**, the blinking "SELECTED *nn*" button will change to **RUN FROM LINE CYCLE START** button. Click this button to start the program from the beginning of the selected line.

THIS CUTPATH will run only the cutpath that the selected segment is a part of.

The blinking "SELECTED *nn*" button will change to **RUN FROM LINE CYCLE START** button. Click this button to run the selected cutpath.

La ejecución desde selección de línea puede cancelarse de las siguientes maneras:

1. Click the background of the preview window - this method will cancel a selection of either a cut line in the preview window, or a G-code line in the G-code window.
2. Click the text of the first line of the G-code program in the G-code display - this method will cancel a selection of either a cut line in the preview window, or a G-code line in the G-code window.
3. Clicking **RELOAD** in the G-code window header - this method will cancel the Run From Line process if LOAD was clicked on the Run From Line dialog box and "rfl.ngc" is displayed as the loaded file name in the G-code window header. This will return the user to the originally loaded file.

---

nota
Although Run From Line allows the user to begin execution at any line, not every possible scenario can be fully tested. Most testing has focused on recovering typical cutting operations. Therefore when recovering outside of cutting operations, users should select a starting line that provides the GUI with the most context about the operation. For example, when restarting a spotting operation, choose the G0 line before the M3 command rather than the M3 itself or the G1 X0.000001 move in the middle of the spotting operation.

10.8.9.33. Scribe

A scribe may be operated by QtPlasmaC in addition to the plasma torch.

Using a scribe requires the use of the LinuxCNC tool table. Tool 0 is assigned to the plasma torch and Tool 1 is assigned to the scribe. The scribe X and Y axes offsets from the plasma torch need to be

entered into the LinuxCNC tool table. This is done by editing the tool table via the main GUI, or by editing the **tool.tbl** file in the ***<machine_name>*** configuration directory. This will be done after the scribe can move to the work piece to help determine the appropriate offset.

The plasma torch offsets for X and Y will always be zero. The tools are selected by the **T*n* M6** command followed by a **G43 H0** command which is required to apply the offsets. The tool is then started with a **M3 $*n* S1** command. For *n*, use 0 for torch cutting or 1 for scribing.

To stop the scribe, use the G-code command **M5 $1**.

If the user has not yet assigned the HAL pins for the scribe in the configuration wizard then they may do so by using the appropriate configuration wizard or by manually editing the HAL file, see modifying QtPlasmaC.

There are two HAL output pins used to operate the scribe; the first pin is used to arm the scribe which moves the scribe to the surface of the material. After the Arm Delay has elapsed, the second pin is used to start the scribe. After the On Delay has elapsed, motion will begin.

Using QtPlasmaC after enabling the scribe requires the selection of either the torch or the scribe in each G-code file as a LinuxCNC tool.

The first step is to set the offsets for the scribe by following the procedure described in Peripheral Offsets.

The final step is to set the scribe delays required:

1. **Arm Delay** - allows time for the scribe to descend to the surface of the material.
2. **On Delay** - allows time for the scribe to start before motion begins.

Guarda los parámetros en la pestaña de configuración.

After the above directions are completed, the scribe may be tested manually by issuing a **M3 $1 S1** command in the MDI input. The user may find it helpful to use this method to scribe a small divot and then try to pulse the torch in the same location to align the offsets between the scribe and the torch.

To use the scribe from G-code:

```
...
M52 P1 (enable adaptive feed)
F#<_hal[plasmac.cut-feed-rate]>
T1 M6 (select scribe)
G43 H0 (apply offsets for current tool)
M3 $1 S1 (start the scribe)
.
M5 $1 (stop the scribe)
.
T0 M6 (select torch)
G43 H0 (apply offsets for current tool)
G0 X0 Y0 (parking position)
M5 $-1 (end all)
```

It is a good idea to switch back to the torch at the end of the program before the final rapid parking move, so the machine is always in the same state at idle.

The user can switch between the torch and the scribe any number of times during a program by using the appropriate G-codes.

Issuing **M3 S1** (without $*n*) will cause the machine to behave as if an **M3 $0 S1** had been issued and issuing **M5** (without $*n*) will cause the machine to behave as if an **M5 $0** had been issued. This will control the torch firing by default in order to provide backward compatibility for previous G-code files.

aviso
If there is an existing manual tool change parameter set in the <*machine_name*>.hal file then QtPlasmaC will convert it to an automatic tool change.

10.8.9.34. Spotting

To achieve spotting to mark the material prior to drilling etc., QtPlasmaC can pulse the torch for a short duration to mark the spot to drill.

Spotting can be configured by following these steps:

1. Set the arc voltage **Threshold** in the Spotting section of the PARAMETERS Tab. Setting the voltage threshold to zero will cause the delay timer to begin immediately upon starting the torch. Setting the voltage threshold above zero will cause the delay timer to begin when the arc voltage reaches the threshold voltage.
2. Set the **Time On** in the Spotting section of the PARAMETERS Tab. When the **Time On** timer has elapsed, the torch will turn off. Times are adjustable from 0 to 9999 milliseconds.

The torch is then turned on in G-code with the **M3 $2 S1** command which selects the plasma torch as a spotting tool.

To turn the torch off, use the G-code command **M5 $2**.

Para obtener mas información sobre varias herramientas vea herramientas múltiples.

LinuxCNC (QtPlasmaC) requires some motion between any **M3** and **M5** commands. For this reason, a minimal movement at a high speed is required to be programmed.

Un ejemplo de código G es:

```
G21 (metric)
F99999 (high feed rate)
.
.
G0 X10 Y10
M3 $2 S1 (spotting on)
G91 (relative distance mode)
G1 X0.000001
G90 (absolute distance mode)
M5 $2 (spotting off)
.
.
G0 X0 Y0
G90
M2
```

nota
The **high feed rate** of 99999 is to ensure that the motion is at the machine's highest feed rate.

importante
SOME PLASMA CUTTERS WILL NOT BE SUITABLE FOR THIS FEATURE.
IT IS RECOMMENDED THAT THE USER CARRY OUT SOME TEST SPOTTING TO ENSURE THAT THE PLASMA CUTTER IS CAPABLE OF UTILIZING THIS FEATURE.

10.8.9.35. Tube Cutting

Tube cutting with an angular A, B, or C axis is achieved with the following in the G-code file:

- #<tube_cut>=1 magic comment before any motion command.
- All material probing must be done using the G38 straight probe codes.
- All Z axis motion is required, PlasmaC does no internal Z axis motion during tube cutting.
- PIERCE_DELAY is the only required material parameter
- Start a cut with M3 $0 S1.
- End a cut with M5 $0

10.8.9.36. Virtual Keyboard Custom Layouts

Virtual keyboard support is available for only the ”onboard” onscreen keyboard. If it is not already on the system, it may be installed by typing the following in a terminal:

```
sudo apt install onboard
```

The following two custom layouts are used for soft key support:



Figura 10.50: Number keypad - used for the CONVERSATIONAL Tab and the PARAMETERS Tab

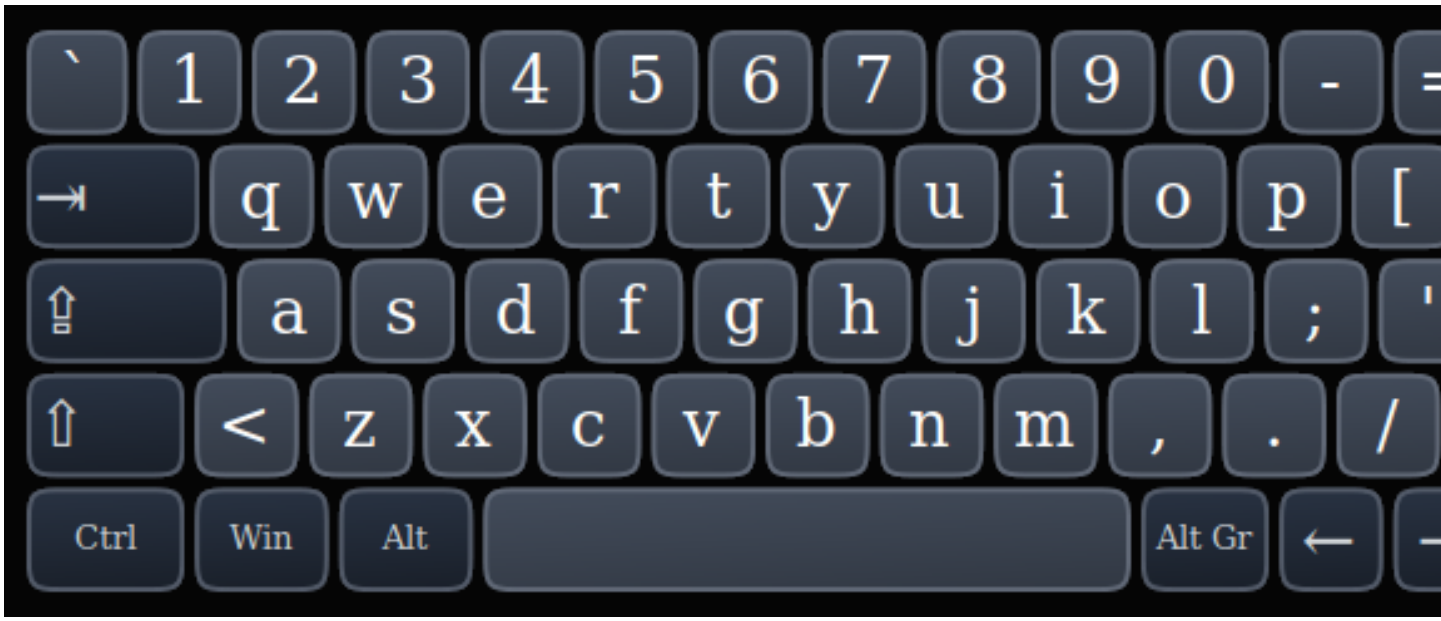


Figura 10.51: Alphanumeric keypad - used for G-code editing and file management.

If the virtual keyboard has been repositioned and on the next opening of a virtual keyboard it is not visible, then clicking twice on the onboard icon in the system tray will reposition the virtual keyboard so the move handle is visible.

10.8.9.37. Keyboard Shortcuts

Below is a list of all available keyboard shortcuts in QtPlasmaC.

nota
Todos los atajos de teclado están predeterminadamente desactivados.

In order to utilize them, **KB Shortcuts** must be enabled in the **GUI SETTINGS** section of the SETTINGS Tab.

Keyboard Shortcut	**Action**
ESC	Aborts current automated motion (example: a running program, a probe test, etc.) as well as an active torch pulse (behaves the same as clicking CYCLE STOP).
F1	Toggles the GUI E-STOP button (if the GUI E-STOP button is enabled).
F2	Toggles the GUI power button.
F9	Toggles the ”Cutting” command, used to begin or end a manual cut.
F12	Muestra el editor de hojas de estilo.
ALT+RETURN	Places QtPlasmaC into Manual Data Input (MDI) mode. Note that ALT + ENTER will achieve the same result. In addition, pressing RETURN (or ENTER) with no entry in the MDI will close the MDI window.
`, 1-9, 0	Changes jog speed to 0 %, 10 %-90 %, 100 % of the value present in the DEFAULT_LINEAR_VELOCITY variable in the **[DISPLAY]** section of the *<machine_name>*.ini file.

Keyboard Shortcut	Action
SHIFT+`, 1-9, 0	Changes rapid speed to 0 %, 10 %-90 %, 100 %.
CTRL+1-9, 0	Cambia la velocidad de alimentación a 10 %-90 %, 100 %.
CTRL+HOME	Homes all axes if they are not yet homed and have a homing sequence set in the <*machine_name*>.ini file. If they are already homed, they will no longer be homed.
CTRL+R	Cycle Start if the program is not already running. Cycle Resume if the program is paused.
END	Touches off X and Y to 0.
DEL	Allows the user to use a laser to set an origin with or without rotation. See the LASER section for detailed instructions.
SPACE BAR	Pausa el movimiento.
CTRL+SPACE BAR	Limpia notificaciones.
O	Opens a new program.
L	Loads the previously opened program if no program is loaded. Reloads the current program if there is a program loaded.
→	Jogs the X axis positive.
←	Jogs the X axis negative.
↑	Jogs the Y axis positive.
↓	Jogs the Y axis negative.
PAGE UP	Jogs the Z axis positive.
PAGE DOWN	Jogs the Z axis negative.
[	Jogs the A axis positive.
]	Jogs the A axis negative.
.	Jogs the B axis positive.
,	Jogs the B axis negative.
SHIFT (+ Jog Key)	The shift key is used with any jog key to invoke a rapid jog.
+ (+Jog Key)	The plus key can be used with any jog key to invoke a rapid jog (behaves the same as SHIFT).
- (+Jog Key)	The minus key can be used with any jog key to invoke a slow jog (10 % of the displayed jog speed) If SLOW jogging is already active, the axis will jog at the displayed jog speed.

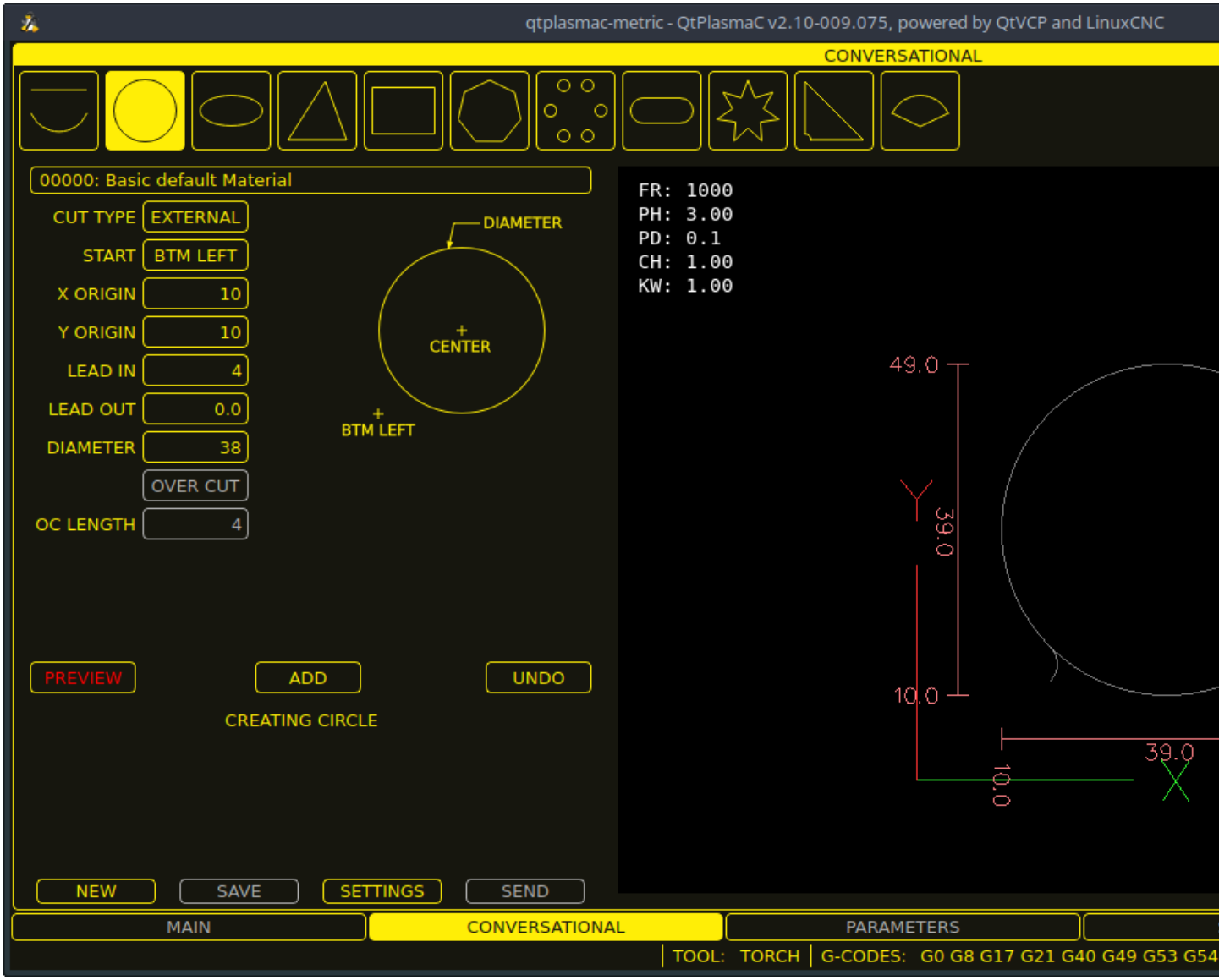
10.8.9.38. MDI

In addition to the typical G and M codes that are allowed by LinuxCNC in MDI mode, the MDI in QtPlasmaC can be used to access several other handy features. The following link outlines the features and their use: Sección 12.7.2.15[MDILine Widget]

nota
M3, M4, and M5 are not allowed in the QtPlasmaC MDI.

In addition, pressing RETURN (or ENTER) with no entry in the MDI will close the MDI window.

10.8.10. Conversational Shape Library



The **Conversational Shape Library** consists of several basic shapes and functions to assist the user with generating quick G-code at the machine to cut simple shapes quickly. This feature is found on the CONVERSATIONAL Tab.

nota
The Conversational Library is not meant to be a CAD/CAM replacement as there are limitations to what can be achieved.

Blank entries in the shape input boxes will use the current setting at the time the G-code was generated. For example, if **X start** was left blank then the current X axis position would be used.

All lead-ins and lead-outs are arcs except for **Circles** and **Stars**:

Circles:

- If the circle is external then any lead-in or lead-out will be an arc.

- If the circle is internal and a **small hole** then any lead-in will be perpendicular and there will be no lead out.
- If the circle is internal and not a **small hole** then any lead-in and let-out will be an arc. If the lead-in has a length greater than half the radius then the lead-in will revert to perpendicular and there will be no let-out. If the lead-out has a length greater than half the radius then there will be no leadout.

Stars:

- The lead-in is at the same angle as the first cut and the leadout is at the same angle as the last cut.

---

nota
A **small hole** is a circle that is smaller than the SMALL HOLE DIAMETER specified in the CONVERSATIONAL SETTINGS page.

---

nota
The holes in a BOLT CIRCLE shape will also abide by the above rules.

The cut order will occur in the same order as the shape was built.

Pressing **Return** on the keyboard while editing parameters will automatically show the preview of the shape if there are enough parameters entered to create the shape. Clicking any of the available check boxes will do the same.

The general functions are as follows:

Name	Descripción
Material Drop-Down	Allows the user to select the desired material for cutting. If "VIEW MATERIAL" is selected on the SETTINGS Tab, a visual reference showing key material cut settings will be displayed on the Conversational Preview Window. Examples are: Feed Rate, Pierce Height, Pierce Delay, Cut Height, and Kerf Width (for Conversational only). Cut Amps will be shown if PowerMax communications are enabled.
NEW	Removes the current G-code file and load a blank G-code file.
SAVE	Opens a dialog box allowing the current shape to be saved as a G-code file.
SETTINGS	Allows the changing of the global settings.
SEND	Loads the current shape into LinuxCNC (QtPlasmaC). If the last edit was not added, then it will be discarded.
PREVIEW	Displays a preview of the current shape provided the required information is present.
CONTINUE	This button is used for lines and arcs only. Allows another segment to be added to the current segment/segments.
ADD	Stores the current shape into the current job.
UNDO	Reverts to the previously stored state.
RELOAD	Reloads the original G-code file or a blank file if none was loaded.

If there is a G-code file loaded in LinuxCNC (QtPlasmaC) when the CONVERSATIONAL Tab is selected, that code will be imported into the conversational as the first shape of the job. If this code is not required, then it can be removed by pressing the **NEW** button.

If there is an added shape that is unsaved or unsent then it is not possible to switch tabs in the GUI. To re-enable switching tabs, it is necessary to either **SAVE** the shape, **SEND** the shape, or press **NEW** to remove the shape.

If **NEW** is pressed to remove an added shape that is unsaved or unsent then a warning dialog will be displayed.

nota
All distances are in machine units relative to the current User Coordinate System and all angles are in degrees.

10.8.10.1. Conversational Settings

Global settings for the shape library can be set by pressing the **SETTINGS** button in the CONVERSATIONAL Tab. This will display all of the available settings parameters that are used for G-code program creation. These include:

- **Preamble**
- **Postamble**
- **Origin (Center or Bottom Left)**
- **Lead-in length**
- **Leadout length**
- **Small hole diameter**
- **Small hole speed**
- **Preview Window Grid Size**

Any internal circle that has a diameter less than **Small hole diameter** is classified as a small hole and will have a straight lead-in with a length that is the lesser of either the radius of the hole or the specified lead-in length. It will also have its feed rate set to **Small hole speed**.

Preamble and Postamble may be entered as a string of G-Codes and M-Codes separate by spaces. If the user wishes for the generated G-code to have each code on an individual line, then this is made possible by separating the codes with **\n**.

This will place all codes on the same line:

```
G21 G40 G49 G64p0.1 G80 G90 G92.1 G94 G97
```

This will place each code on its own line:

```
G21\nG40\nG49\nM52P1\nG64p0.1\nG80\nG90\nG92.1\nG94\nG97
```

Note that LinuxCNC does not allow multiple **P** words on the same line.

Pressing the **RELOAD** button will discard any changed but unsaved settings.

Pressing the **SAVE** button will save all the settings as displayed.

Pressing the **EXIT** button will close the setting panel and return to the previous shape.

10.8.10.2. Conversational Lines And Arcs



Lines and arcs have an additional option in that they may be strung together to create a complex shape.

There are two line types and three arc types available:

1. **Line** given a start point and an end point.
2. **Line** given a start point, length, and angle.
3. **Arc** given a start point, way point, and end point.
4. **Arc** given a start point, end point, and radius.
5. **Arc** given a start point, length, angle, and radius.

To use lines and arcs:

1. Select the **Lines and Arcs** icon.
2. Select the type of line or arc to create.
3. Choose the material from the MATERIAL drop-down. If no material is chosen, the default material (00000) will be used.
4. Introduza los parámetros deseados.
5. Press **PREVIEW** to see the shape.
6. If satisfied with the shape press **CONTINUE**.
7. Change the line or arc type if needed and continue this procedure until the shape is complete.
8. Press **SEND** to send the G-code file to LinuxCNC (QtPlasmaC) for cutting.

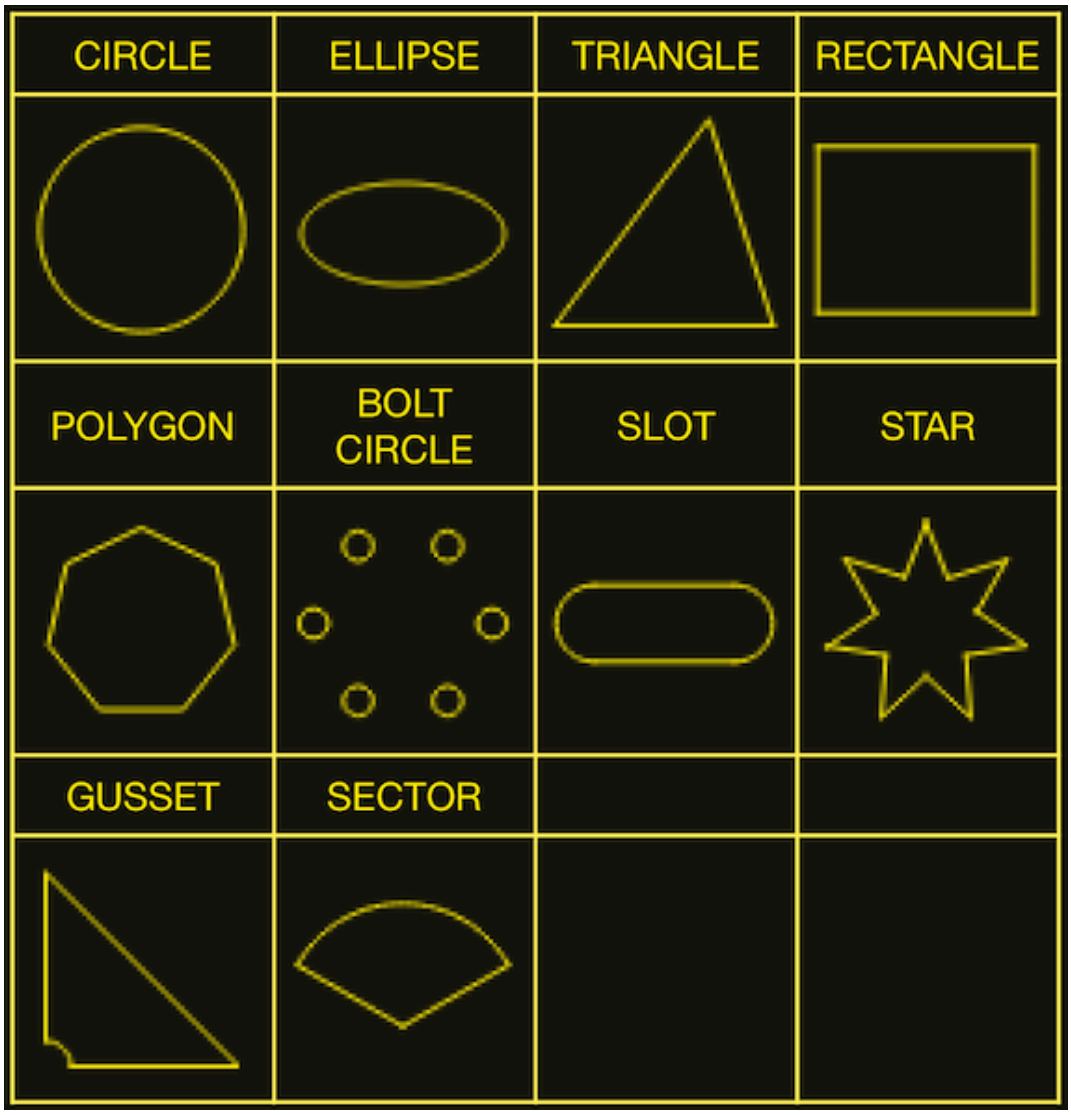
If the user wishes to create a closed shape, they will need to create any required leadin as the first segment of the shape. If a leadout is required, it will need to be the last segment of the shape.

---

nota
At this stage there is no automatic option for a lead-in/leadout creation if the shape is closed.

10.8.10.3. Conversational Single Shape

The following shapes are available for creation:



Para crear una forma:

1. Select the corresponding icon for the shape to create. The available parameters will be displayed.
2. Choose the material from the MATERIAL drop-down. If no material is chosen, the default material (00000) will be used.
3. Enter the appropriate values and press **PREVIEW** to display the shape.
4. If the shape is not correct, edit the values and press **PREVIEW** and the new shape will be displayed. Repeat until satisfied with the shape.
5. Press **ADD** to add the shape to the G-code file.
6. Press **SEND** to send the G-code file to LinuxCNC (QtPlasmaC) for cutting.

For **CIRCLE**, the **OVER CUT** button will become valid when a CUT TYPE of INTERNAL is selected and the value entered in the DIAMETER field is less than the Small Hole Diameter parameter in the Conversational SETTINGS section.

For **BOLT CIRCLE** the **OVER CUT** button will become valid if the value entered in the HOLE DIA field is less than the SMALL HOLES DIAMETER parameter in the Conversational SETTINGS section.

For the following shapes, KERF OFFSET will become active once a LEAD IN is specified:

1. TRIANGLE
2. RECTANGLE
3. POLYGON
4. SLOT
5. STAR
6. GUSSET

10.8.10.4. Conversational Group Of Shapes

Multiple shapes can be added together to create a complex group.

The cut order of the group is determined by the order in which the individual shapes are added to the group.

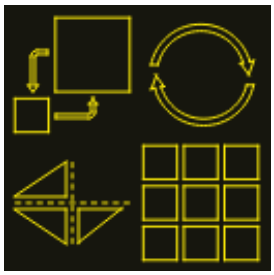
Once a shape is added to the group it cannot be edited or removed.

Groups cannot have shapes removed, only added to.

To create a group of shapes:

1. Create the first shape as in **Single Shape**.
2. Press **ADD** and the shape will be added to the group.
3. If the user wishes to add another version of the same shape, then edit the required parameters and press **ADD** when satisfied with the shape.
4. If the user wishes to add a different shape, select that shape and create it as in **Single Shape**.
5. Repeat until all the required shapes to complete the group have been added.
6. Press **SEND** to send the G-code file to LinuxCNC (QtPlasmaC) for cutting.

10.8.10.5. Conversational Block



The Conversational Block feature allows block operations to be performed on the current shape or group of shapes displayed in the CONVERSATIONAL Tab. This can include a G-code file not created using the Conversational Shape Library that has been previously loaded from the MAIN Tab.

A previously saved Block G-code file may also be loaded from the MAIN Tab and then have any of its operations edited using the Conversational Block feature.

All block operations are done in the machines native units. If a file of a different units system has been opened then the copy loaded into the CONVERSATIONAL Tab will be converted to the machines native units when the CONVERSATIONAL Tab becomes active.

Bloquear opciones:

- Rotate
- Scale
- Array
- Mirror
- Flip

To create a block:

1. Create a shape, a group, or use a previously loaded G-code file.
2. Click the **Block** icon to open the Block tab.
3. Enter the appropriate values in the Block tab and press **PREVIEW** to display the resulting changes.
4. If the result is not correct, edit the values and press **PREVIEW** and the new result will be shown. Repeat until satisfied with the result.
5. Press **ADD** to complete the procedure.
6. Press **SEND** to send the G-code file to LinuxCNC (QtPlasmaC) for cutting, or **SAVE** to save the G-code file.

COLUMNS & ROWS
: specifies the number of duplicates of the original shape arranged in columns and rows as well as the spacing between each shape's origin.

ORIGIN
: se desplaza el resultado desde las coordenadas de origen.

ANGLE
: rota el resultado.

SCALE
: scale the result.

ROTATION
: rotate the shape within the result.

MIRROR
: mirror the shape about its X coordinates within the result.

FLIP
: flip the shape about its Y coordinates within the result.

If the result is an array of shapes, then the cut order of the result is from the left column to the right column, starting at the bottom row and ending at the top row.

10.8.10.6. Conversational Saving A Job

The current job displayed in the Preview Panel may be saved at any time by using the bottom **SAVE** button. If the G-code has been sent to LinuxCNC (QtPlasmaC) and the user has left the CONVERSATIONAL Tab, the user may still save the G-code file from the GUI. Alternatively, the user could click the CONVERSATIONAL Tab which will reload the job, at which time they can press the **SAVE** button.

10.8.11. Mensajes de error

10.8.11.1. Error Logging

All errors are logged into the machine log which is able to be viewed in the STATISTICS Tab. The log file is saved into the configuration directory when QtPlasmaC is shutdown. The five last logfiles are kept, after which the oldest logfile is deleted each time a new log file is created. These saved log files may be viewed with any text editor.

10.8.11.2. Error Message Display

By default, QtPlasmaC will display error messages via a Operator Error popup window. In addition, QtPlasmaC will alert the user that an error has been sent to the machine log by displaying the message **"ERROR SENT TO MACHINE LOG"** in the lower left portion of the status bar.

The user may opt to disable the Operator Error popup window, and view the error messages by going to the STATISTICS Tab by changing the following option to **False** in the **[SCREEN_OPTIONS]** of the <*machine_name*>.prefs file in the <*machine_name*> directory:

```
desktop_notify
```

---

nota
<*machine_name*>.prefs must be edited with QtPlasmaC closed or any changes will be overwritten on exit.

Additionally, it is possible for **ERROR SENT TO MACHINE LOG** to flash to get the user's attention by adding or editing the following option in the **[GUI_OPTIONS]** section of the <*machine_name*>.prefs file:

```
Flash error = True
```

10.8.11.3. Critical Errors

There are a number of error messages printed by QtPlasmaC to inform the user of faults as they occur. The messages can be split into two groups, **Critical** and **Warning**.

Critical Errors will cause the running program to pause, and the operator will need to clear the cause of the error before proceeding.

If the error was received during cutting, then forward or reverse motion is allowed while the machine is paused to enable the user to reposition the machine prior to resuming the cut.

When the error is cleared the program may be resumed.

These errors indicate the corresponding sensor was activated during cutting:

- **breakaway switch activated, program is paused**
- **float switch activated, program is paused**
- **ohmic probe activated, program is paused**

These errors indicate the corresponding sensor was activated before probing commenced:

- **ohmic probe detected before probing program is paused**

- **float switch detected before probing program is paused**
- **breakaway switch detected before probing program is paused**

The Arc OK signal was lost during cutting motion, before the **M5** command was reached:

- **valid arc lost program is paused**

The Z axis reached the bottom limit before the work piece was detected:

- **bottom limit reached while probing down program is paused**

The work piece is too high for any safe rapid removes:

- **material too high for safe traverse, program is paused**

One of these values in MATERIAL section of the PARAMETERS Tab is invalid (For example: if they are set to zero):

- **invalid pierce height or invalid cut height or invalid cut volts, program is paused**

No arc has been detected after attempting to start the number of times indicated by **Max Starts** in the ARC frame of the CONFIGURATION section of the PARAMETERS Tab:

- **no arc detected after <*n*>d start attempts program is paused**
- **no arc detected after <*n*>d start attempts manual cut is stopped**

THC has caused the bottom limit to be reached while cutting:

- **bottom limit reached while THC moving down program is paused**

THC has caused the top limit to be reached while cutting:

- **top limit reached while THC moving up program is paused**

These errors indicate move to pierce height would exceed the Z Axis MAX_LIMIT for the corresponding probe method:

- **pierce height would exceed Z axis maximum limit condition found while moving to probe height during float switch probing**
- **pierce height would exceed Z axis maximum limit condition found while moving to probe height during ohmic probing**

These errors indicate the move to pierce height would exceed the Z axis maximum safe height for the corresponding probe method:

- **pierce height would exceed Z axis maximum safe height condition found while float switch probing**
- **pierce height would exceed Z axis maximum safe height condition found while ohmic probing**

10.8.11.4. Warning Messages

Warning messages will not pause a running program and are informational only.

These messages indicate the corresponding sensor was activated before a probe test commenced:

- **ohmic probe detected before probing probe test aborted**
- **float switch detected before probing probe test aborted**
- **breakaway switch detected before probing probe test aborted**

This indicates that the corresponding sensor was activated during a consumable change:

- **breakaway, float, or ohmic activated during consumable change, motion is paused WARNING: MOTION WILL RESUME IMMEDIATELY UPON RESOLVING THIS CONDITION!**



aviso
CONSUMABLE CHANGE MOTION WILL RESUME IMMEDIATELY UPON RESOLVING THE CORRESPONDING SENSOR ACTIVATION.

This indicates that the corresponding sensor was activated during probe testing:

- **breakaway switch detected during probe test**

This indicates that probe contact was lost before probing up to find the zero point:

- **probe trip error while probing**

This indicates that the bottom limit was reached during a probe test:

- **bottom limit reached while probe testing**

This indicates that the move to pierce height would exceed the Z Axis MAX_LIMIT during the corresponding probe method:

- **pierce height would exceed Z axis maximum limit condition found while moving to probe height during float switch probe testing**
- **pierce height would exceed Z axis maximum limit condition found while moving to probe height during ohmic probe testing**

This indicates that the safe height has been reduced due to THC raising the Z axis during cutting:

- **safe traverse height has been reduced**

This indicates that the value for the Arc Voltage was invalid (NAN or INF) when QtPlasmaC launched.

- **invalid arc-voltage-in**

10.8.12. Updating QtPlasmaC

10.8.12.1. Standard Update

QtPlasmaC update notices are posted at https://forum.linuxcnc.org/plasmac/37233-plasmac-updates .

Users are strongly encouraged to create a Username and subscribe to the above thread to receive update notices.

For a standard ISO installation, LinuxCNC will only be updated when a new minor release has been released. QtPlasmaC will then automatically update its configuration the first time it is run after a LinuxCNC update.

LinuxCNC is normally updated by entering the following commands into a terminal window (one at a time):

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

10.8.12.2. Continuous Update

Enhancements and bug fixes will not be available on a standard installation until a new minor release of LinuxCNC has been released. If the user wishes to update whenever a new QtPlasmaC version has been pushed, they could use the LinuxCNC Buildbot repository rather than the standard LinuxCNC repository by following the instructions at http://buildbot.linuxcnc.org/ .

10.8.13. Modify An Existing QtPlasmaC Configuration

Hay dos maneras para modificar una configuración de QtPlasmaC existente:

1. Running the appropriate configuration wizard and loading the .conf file saved by the wizard.
2. Manually edit the INI and/or the HAL file of the configuration.



importante
Any manual modification to the <*machine_name*>.ini and <*machine_name*>.hal files will not be registered in PnCConf or StepConf.

nota
If unsure of the HAL pin’s full name, the user may start LinuxCNC and run **HalShow** for a full listing of all HAL pins.

10.8.14. Customizing QtPlasmaC GUI

Styling of the QtPlasmaC GUI is done with Qt style sheets and some customization may be achieved by the use of a custom style sheet. This allows the user to change some GUI items such as color, border, size, etc. It cannot change the layout of the GUI.

Information on Qt style sheets is available here.

There are two methods available to apply custom styles:

1. Add A Custom Style: use this for minor style changes.
2. Create A New Style use this for a complete style change.

10.8.14.1. Add A Custom Style

Adding style changes to the default style sheet is achieved by creating a file in the <*machine_name*> configuration directory. This file MUST be named qtplasmac_custom.qss. Any required style changes are then added to this file.

For example, the user may want the arc voltage display in red, a green Torch On LED of a larger size and a larger Torch Enable button. This would be done with the following code in qtplasmac_custom.qss:

```
#arc_voltage {
    color: #ff0000 }

#led_torch_on {
    qproperty-diameter: 30;
    qproperty-color: green }

#torch_enable::indicator {
    width: 30;
    height: 30}
```

10.8.14.2. Create A New Style

Custom style sheets are enabled by setting the following option in the **[GUI_OPTIONS]** section of the <*machine_name*>.prefs file. This option must be set to the filename of the style sheet as shown below.

```
Custom style = the_cool_style.qss
```

The filename may be any valid filename. The standard extension name is .qss but this is not mandatory.

There are some constraints on the custom style sheet for QtPlasmaC, e.g., the jog buttons, cut-recovery buttons, and the conversational shape buttons are image files and are not able to be custom styled.

The custom style file requires a header in the following format:

```
/*****************************
Custom Style-sheet Header

color1 = #000000
#QtPlasmaC default = #ffee06

color2 = #e0e0e0
#QtPlasmaC default = #16160e

color3 = #c0c0c0
#QtPlasmaC default = #ffee06

color4 = #e0e0e0
#QtPlasmaC default = #26261e

color5 = #808080
#QtPlasmaC default = #b0b0b0

*****************************/
```

The colors may be expressed in any valid stylesheet format.

The above colors are used for the following widgets. So any custom styling will need to take these into account. The colors shown below are the defaults used in QtPlasmaC along with the color name from the SETTINGS Tab.

Color	Parámetro	Affects
color1 (#ffee06)	Foreground	foreground of jog buttons foreground of latching user buttons foreground of camera/laser buttons foreground of conversational shape buttons background of active conversational shape buttons
color2 (#16160e)	Background	background of latching user buttons background of camera/laser buttons background of G-code editor active line background of conversational shape buttons
color3 (#ffee06)	Highlight	background of active latching user buttons background of active camera/laser buttons foreground of G-code editor cursor
color4 (#36362e)	Alt Background	background of G-code display’s active line

10.8.14.3. Returning To The Default Styling

The user may return to the default styling at any time by following the following steps:

1. Close QtPlasmaC if open.
2. Delete qtplasmac.qss from the machine config directory.
3. Delete qtplasmac_custom.qss from the machine config directory (if it exists).
4. Open <*nombre_de_máquina*>.prefs file.
5. Delete the **[COLOR_OPTIONS]** section.
6. Delete the Custom style option from the **[GUI_OPTIONS]** section.
7. Guardar el archivo.

The next time QtPlasmaC is loaded all custom styling will be removed and the default styling will return.

Below is an example of the section and options to be deleted from <*machine_name*>.prefs:

```
[COLOR_OPTIONS]
Foreground = #ffee06
Highlight = #ffee06
LED = #ffee06
Background = #16160e
Background Alt = #36362e
Frames = #ffee06
Estop = #ff0000
Disabled = #b0b0b0
Preview = #000000
```

10.8.14.4. Custom Python Code

It is possible to add custom Python code to change some existing functions or to add new ones. Custom code can be added in two different ways: a user command file or a user periodic file.

A user command file is specified in the DISPLAY section of the <*machine_name*>.ini file and contains Python code that is processed only once during startup.

```
USER_COMMAND_FILE = mi_codigo.py
```

A user periodic file must be named user_periodic.py and must be loaded in the machine's config directory. This file is processed every cycle (usually 100 ms) and is used for functions that require regular updating.

10.8.14.5. Custom G-code Filter

All incoming G-code is parsed by a G-code filter to ensure it is suitable for QtPlasmaC. It is possible to extend this filter with custom python code executed from a file in the configuration directory to aid in converting different flavours of G-code to a format suitable for QtPlasmaC.

The name of this file is custom_filter.py and it will be automatically used if it exists.

There are three preset methods available for use:

Name	Función
custom_pre_process	This does basic processing of each line before any processing is done in the filter.
custom_pre_parse	This parses any G-code from a line **before** any parsing done in the filter.
custom_post_parse	This parses any G-code from a line **after** any parsing done in the filter.

These methods are applied by the following procedure:

- Define the method with an argument for the incoming data.
- Add any required code to manipulate the data.
- Return the resultant data.
- Attach the new method.

For example, to remove any code beginning with *G71* and change *M2* to *M5 $0* and *M2*:

```
def custom_pre_parse(data):
    if data[:3] == 'G71':
        return(None)
    if data == 'M2':
        return(f'M5 $0\n\n{data}')
    return(data)
self.custom_pre_parse = custom_pre_parse
```

In addition to these it is also possible to override any existing method in the filter the same way. This requires defining the same number of arguments as the existing method, noting that *self* in the original does not constitute an argument.

```
def new_method_name(data):
    if data[:3] == 'G71':
        return(None)
    return(data)
self.old_method_name = new_method_name
```

nota
The existing filter code may be observed in the file /bin/qtplasmac_gcode.
The file sim/qtplasmac/custom_filter.py has example skeleton code for custom filtering.

10.8.15. QtPlasmaC Advanced Topics

10.8.15.1. Custom User Buttons

The QtPlasmaC GUI offers user buttons that can be customized by adding commands in the USER BUTTON ENTRIES section of the SETTINGS Tab in the <*machine_name*>.prefs file.

The number of user buttons varies by display type and resolution as follows:

- 16:9 and 4:3 - Minimum 8, Maximum 20
- 9:16 - Minimum 15, Maximum 20

The user will need to run QtPlasmaC at the desired screen size to determine how many user buttons are available for use.

All <*machine_name*>.prefs file settings for the buttons are found in the **[BUTTONS]** section.

Button Names The text that appears on the button is set the following way:

```
n Name = HAL Show
```

Where *n* is the button number and **HAL Show** is the text.

For text on multiple lines, split the text with a \ (backslash):

```
n Name = HAL\Show
```

If an ampersand is required to be displayed as text then two consecutive ampersands are required:

```
n Name = PIERCE&&CUT
```

Button Code Buttons can run the following:

1. External commands
2. External python scripts
3. G-code commands
4. Dual code
5. Toggle a HAL pin
6. Toggle the alignment laser HAL pin
7. Pulse a HAL pin
8. Probe test
9. Ohmic Test
10. Cut Type
11. Change consumables
12. Carga un programa en código G
13. Pulse the torch on
14. Single unidirectional cut
15. Framing a job
16. Begin/End a manual cut

17. Display/Hide an offsets viewer
18. Load the latest modified NGC file found in a directory
19. Display/Hide the online HTML user manual
20. Toggle between joint and teleop modes

External Commands

To run an external command, the command is preceded by a % character.

```
n Code = %halshow
```

External Python Scripts

To run an external Python script, the script name is preceded by a % character and it also requires a .py extension. It is valid to use the ~ character as a shortcut for the user's home directory.

```
n Code = %~/user_script.py
```

G-code

To run G-code, just enter the code to be run.

```
n Code = G0 X100
```

Para ejecutar una subrutina existente.

```
n Code = o<the_subroutine> call
```

<*machine_name*>.ini file variables can be entered by using the standard LinuxCNC G-code format. If expressions are included, then they need to be surrounded by brackets.

```
n Code = G0 X#<_ini[joint_0]home> Y1
n Code = G53 G0 Z[#<_ini[axis_z]max_limit> - 1.001]
```

<*machine_name*>.prefs file variables and also <*machine_name*>.ini variables can be entered by enclosing each option in **{}** . You must put a space after each **}** if there are any following characters. If expressions are included, then they need to be surrounded by brackets.

```
BUTTON_n_CODE = G0 X{LASER_OFFSET X axis} Y{LASER_OFFSET Y axis}
BUTTON_n_CODE = G0 X{JOINT_0 HOME} Y1
BUTTON_n_CODE = G53 G0 Z[{AXIS_Z MAX_LIMIT} - 1.001]
```

Multiple codes can be run by separating the codes with a "\" (backslash) character. The exception is the special commands which are required to be a single command per button.

```
n Code = G0 X0 Y0 \ G1 X5 \ G1 Y5
```

External commands and G-code may be mixed on the same button.

```
n Code = %halshow \ g0x.5y.5 \ %halmeter
```

Código dual

Dual Code allows the running of two code snippets alternately with each button press. The button text will alternate with each button press and the indicator light may be optionally enabled.

It is mandatory to specify the button code in the following order: "dual-code", the first code, the alternate button text, and the second code separated by double semicolons. If an indicator is required then optionally add ";; true" at the end.

```
n Code = dual-code ;; code1 ;; name1 ;; code2 ;; true
```

On the first button press, code1 will be run, the button text will change to name1, and if ”true” is specified the indicator will light.

On the second button press, code2 will be run, the button text will change to n Name, and the indicator will extinguish if lit.

code1 and code2 both follow the rule of the preceding code explanations, External commands, Python code, and G-code. Multiple codes as well as mixing codes are allowed.

The following code will allow the user to use a single button to run two code snippets alternately each button press:

```
n Name = X+10
n Code = dual-code ;; G91\G0X10\G90 ;; X-10 ;; G91\G0X-10\G90
```

The original label will be X+10, when pressed the torch will move positive 10 in the X axis and the label will change to X-10. When pressed again the torch will move negative 10 in the X axis and the label will change to X+10.

Special Commands The following commands must be a single command per user button, and the button code must start with the special command. The exception is toggle-laser which may be appear anywhere in the code as demonstrated below.

Toggle HAL Pin

The following code will allow the user to use a button to invert the current state of a HAL bit pin:

```
n Code = toggle-halpin the-hal-pin-name
```

This code is required to be used as a single command and may only control one HAL bit pin per button.

The button colors will follow the state of the HAL pin.

After setting the code, upon clicking, the button will invert colors, and the HAL pin will invert pin state. The button will stay ”latched” until the button is clicked again, which will return the button to the original colors and the HAL pin to the original pin state.

It is also possible for the user to specify alternate text which will display on the button while ever it is in the latched-on condition. To specify the alternate text, use a double semicolon followed by the required text. This must be the last item in the button code.

```
n Code = toggle-halpin the-hal-pin-name ;; PIN\TOGGLED
```

There are three External HAL Pins that are available to toggle as an output, the pin names are `qtplasmac.ext_out_0`, `qtplasmac.ext_out_1`, and `qtplasmac.ext_out_2`. HAL connections to these HAL pins need to be specified in a postgui HAL file as the HAL pins are not available until the QtPlasmaC GUI has loaded.

For toggle-halpin buttons, it is possible for the user to mark the associated HAL pin as being required to be turned ”ON” before starting a cut sequence by adding ”cutcritical” after the HAL pin in the button code. If **TORCH ENABLE** is checked and **CYCLE START**, **MANUAL CUT**, or **SINGLE CUT** are initiated while the ”cutcritical” button is not ”ON” then the user will receive a dialog warning them as such and asking to CONTINUE or CANCEL. The dialog will list all untoggled buttons with a corresponding checkbox and allow the user to choose which of the buttons should be toggled automatically upon clicking CONTINUE.

```
n Code = toggle-halpin the-hal-pin-name cutcritical
```

Toggle Alignment Laser HAL Pin

The following code will allow the user to use a button to invert the current state of the alignment laser HAL bit pin:

```
n Code = toggle-laser
```

This code is also able to be used as a multiple command with G-code or external commands but may control only the alignment laser HAL bit pin.

The button colors will follow the state of the alignment laser HAL pin.

After setting the code, upon clicking, the button will invert colors, and the alignment laser HAL pin will invert pin state. The button will stay ”latched” until the button is clicked again, which will return the button to the original colors and the alignment laser HAL pin to the original pin state.

The following code would allow the user to use a button to invert the current state of the alignment laser HAL bit pin and then move the X and Y axes to the offset for the alignment laser as specified in the <*machine_name*>.prefs file:

```
n Code = G0 X{LASER_OFFSET X axis} Y{LASER_OFFSET Y axis} \ toggle-laser
```

The position of the ”toggle-laser” command is not important as it is always the first command actioned regardless of position.

Pulse HAL Pin

The following code will allow the user to use a button to pulse a HAL bit pin for a duration of 0.5 seconds:

```
n Code = pulse-halpin the-hal-pin-name 0.5
```

This code is required to be used as a single command and may only control one HAL bit pin per button.

The pulse duration is specified in seconds, if the pulse duration is not specified then it will default to one second.

The button colors will follow the state of the HAL pin.

After setting the code, upon clicking the button, the button will invert colors, the HAL pin will invert pin state, and the time remaining will be displayed on the button. The button color and the pin state will stay inverted until the pulse duration timer has completed, which will return the button to the original colors, the HAL pin to the original pin state, and the original button name.

An active pulse can be canceled by clicking the button again.

There are three External HAL Pins that are available to pulse as an output, the pin names are qtplasmac.ext_out_0, qtplasmac.ext_out_1, and qtplasmac.ext_out_2. HAL connections to these HAL pins need to be specified in a postgui HAL file as the HAL pins are not available until the QtPlasmaC GUI has loaded.

Probe Test

QtPlasmaC will begin a probe and when the material is detected, the Z axis will rise to the Pierce Height currently displayed in the MATERIAL section of the PARAMETERS Tab. If the user has ”View Material” selected in the GUI SETTINGS section of the SETTINGS Tab, this value will be displayed in the top left corner of the PREVIEW Window next to **PH:**.

QtPlasmaC will then wait in this state for the time specified (rounded to no decimal places) before returning the Z axis to the starting position. An example of a 6 second delay is below. If there is no time specified, then the probe time will default to 10 seconds.

```
n Code = probe-test 6
```

---

nota
Enabling a user button as a Probe Test button will add an external HAL pin that may be connected from a pendant etc. HAL connections to this HAL pin needs to be specified in a postgui HAL file as the HAL pin is not available until the QtPlasmaC GUI has loaded.

Ohmic Test

QtPlasmaC will enable the Ohmic Probe Enable output signal and if the Ohmic Probe input is sensed, the LED indicator in the SENSOR Panel will light. The main purpose of this is to allow a quick test for a shorted torch tip.

```
n Code = ohmic-test
```

---

nota
Enabling a user button as an Ohmic Test button will add an external HAL pin that may be connected from a pendant etc. HAL connections to this HAL pin needs to be specified in a postgui HAL file as the HAL pin is not available until the QtPlasmaC GUI has loaded.

Cut Type

This button if selected will toggle between the two cut types, Pierce and Cut (default cutting mode) or Pierce Only.

```
n Code = cut-type
```

Change Consumables

Pressing this button moves the torch to the specified coordinates when the machine is paused to allow the user easy access to change the torch consumables.

Valid entries are X*nnn* Y*nnn* F*nnn*. At least one of the X or Y coordinates are required, Feed Rate (F) is optional.

The X and Y coordinates are in absolute machine coordinates. If X or Y are missing, then the current coordinate for that axis will be used.

Feed Rate (F) is optional, if it is missing or invalid then the feed rate of the current material will be used.

There are three methods to return to the previous coordinates:

1. Press the **Change Consumables** button again - the torch will return to the original coordinates and the machine will wait in this position for the user to resume the program.
2. Press **CYCLE RESUME** - the torch will return to the original coordinates and the program will resume.
3. Press **CYCLE STOP** - the torch will return to the original coordinates and the program will abort.

```
n Code = change-consumables X10 Y10 F1000
```

---

nota
Enabling a user button as a Change Consumables button will add an external HAL pin that may be connected from a pendant etc. HAL connections to this HAL pin needs to be specified in a postgui HAL file as the HAL pin is not available until the QtPlasmaC GUI has loaded.

Load

Loading a G-code program from the directory specified by the **PROGRAM_PREFIX** variable in the <*machine_name*>.ini file (usually ~/linuxcnc/nc_files) is possible by using the following format:

```
n Code = load G-code.ngc
```

If the user's G-code file is located in a sub-directory of the **PROGRAM_PREFIX** directory, it would be accessed by adding the sub-directory name to the beginning of the G-code file name. Example for a sub-directory named **plasma**:

```
n Code = load plasma/G-code.ngc
```

Note that the first "/" is not necessary as it will be added automatically.

Torch Pulse

Pulse the torch on for a predetermined time. The time must be specified in seconds using up to one decimal place. The maximum allowable time is 3 seconds, anything specified above that value will be limited to 3 seconds. An example of a 0.5 second pulse is below. It there is no time specified then it will default to 1 second. Pulse times with more than one decimal place will be rounded to one decimal place.

Pressing the button again during the countdown will cause the torch to be turned off, as will pressing *Esc* if keyboard shortcuts are enabled in the SETTINGS Tab.

If the button is released before the countdown is complete then the torch will turn off at countdown completion, holding the button on until after the countdown has completed will cause the torch to remain on until the button has been released.

```
n Code = torch-pulse 0.5
```

---

nota
Enabling a user button as a Torch Pulse button will add an external HAL pin that may be connected from a pendant etc. HAL connections to this HAL pin needs to be specified in a postgui HAL file as the HAL pin is not available until the QtPlasmaC GUI has loaded.

Single Cut

Run a single unidirectional cut. This utilises the automatic Single Cut feature.

```
n Code = single-cut
```

Framing

Framing is the ability to move the torch around the perimeter of a rectangle that encompasses the bounds of the current job.

The laser enable HAL pin (qtplasmac.laser_on) will be turned on during the framing moves and any X/Y offsets for the laser pointer in the <*machine_name*>.prefs file will also be applied to the X/Y motion. After the framing motion is completed, the torch will move to the X0 Y0 position to clear any applied laser offsets and qtplasmac.laser_on will be turned off.

Upon starting a Framing cycle, it is important to note that by default the Z axis will be moved to a height of [AXIS_Z]MAX_LIMIT - 5 mm (0.2") before X/Y motion begins.

The velocity for the XY movements of the Framing motion can be specified so that Framing motion always occurs at a set velocity. This can be achieved by adding the feed rate (F) as the as the last portion of the button code. If the feed rate is omitted from the button code, framing motion velocity will default to the feed rate for the currently selected material.

The following GUI buttons and Keyboard Shortcuts (if enabled in the SETTINGS Tab) are valid during Framing motion:

1. Pressing **CYCLE STOP** or the ESC keyboard shortcut - Stops Framing motion.
2. Pressing **CYCLE PAUSE** or the SPACE BAR keyboard shortcut- Pauses Framing motion.

3. Pressing **CYCLE RESUME** or the CTRL+r keyboard shortcut- Resumes paused Framing motion.
4. Changing the **FEED SLIDER** or any of the CTRL+0-9 keyboard shortcuts - Slows the feed rate.

---

nota
IF THE FEED RATE IS CHANGED FOR THE FRAMING MOTION, IT WILL BE NECESSARY TO RETURN THE FEED SLIDER TO 100 % BEFORE PRESSING CYCLE START AND CUTTING THE LOADED JOB.

```
n Code = framing
```

It is possible for the user to omit the initial default Z movement and run the framing sequence at the current Z height by adding ”usecurrentzheight” after ”framing”.

```
n Code = framing usecurrentzheight
```

Para especificar una velocidad de avance:

```
n Code = framing F100
```

o:

```
n Code = framing usecurrentzheight F100
```

Enabling a user button as a framing button will add an external HAL pin that may be connected from a pendant etc. HAL connections to this HAL pin needs to be specified in a postgui HAL file as the HAL pin is not available until the QtPlasmaC GUI has loaded.

Manual Cut

Manual Cut functions identically to the **F9** button to begin or end a manual cut.

```
n Code = manual-cut
```

Offset Viewer

This allows the showing/hiding of an offset viewing screen that displays all machine offsets. All relative offsets can be edited and the G54 ~ G59.3 work system coordinates are able to be given custom names.

```
n Code = offsets-view
```

Load Latest File

This allows the loading of the last modified file in a directory. The directory name is optional and if omitted will default to the last directory a file was loaded from.

```
n Code = latest-file /home/me/linuxcnc/nc_files/qtplasmac-test
```

User Manual

This allows the showing/hiding of the online HTML user manual specific to the version of LinuxCNC currently running. Note that internet access is required for this functionality.

```
n Code = user-manual
```

Toggle Joint Mode

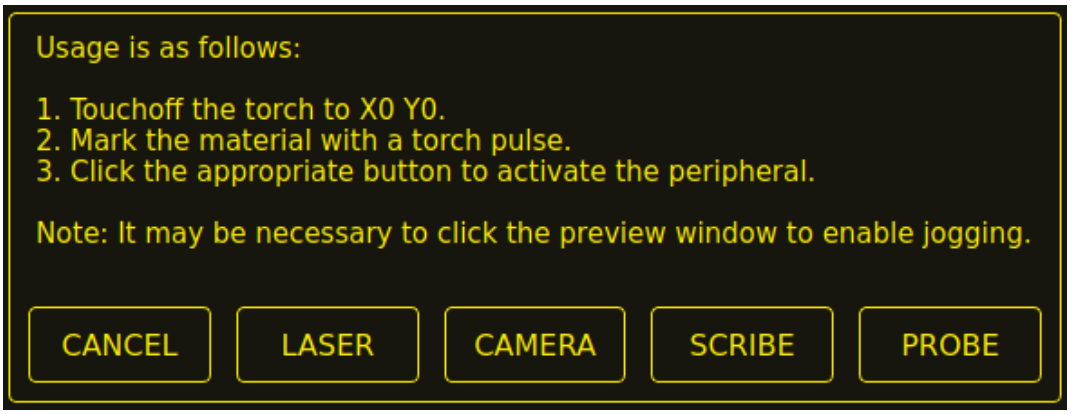
This allows the toggling between joint mode and teleop mode. The machine must be on and homed for this button to be active.

```
n Code = toggle-joint
```

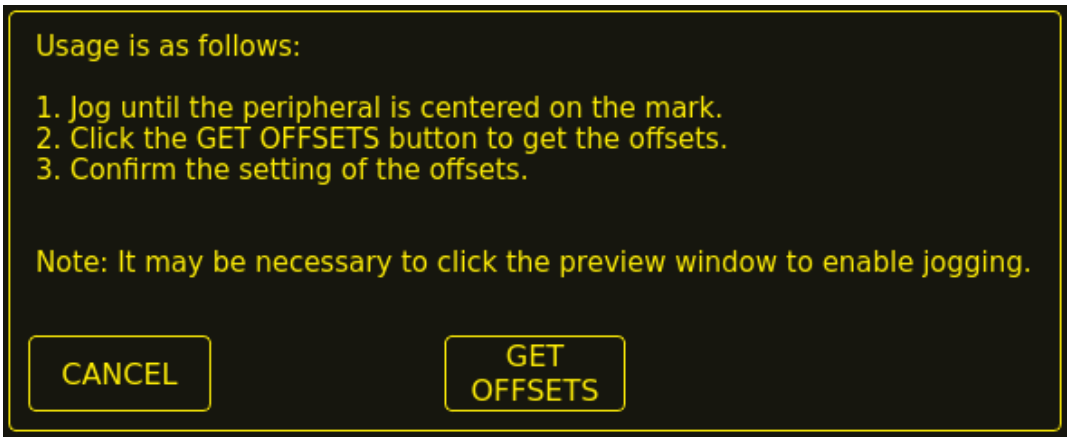
10.8.15.2. Peripheral Offsets (Laser, Camera, Scribe, Offset Probe)

Use the following sequence to set the offsets for a laser, camera, scribe, or offset probe:

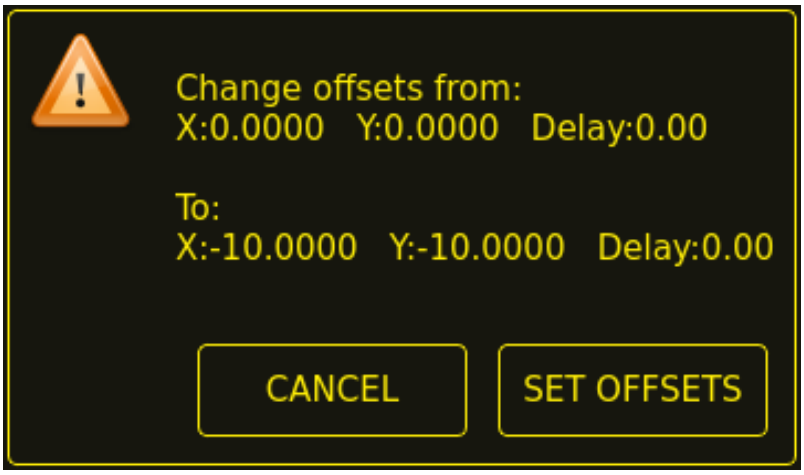
1. Place a piece of scrap material under the torch.
2. The machine must be homed and idle before proceeding.
3. Open the SETTINGS tab.
4. Click the SET OFFSETS button which opens the Set Peripheral Offsets dialog.



5. Click the X0Y0 button to set the torch position to zero.
6. Make a mark on the material by one of:
   a. Jog the torch down to pierce height then pulse the torch on to make a dimple in the material.
   b. Place marking dye on the torch shield then jog the torch down to mark the material.
7. Click the appropriate button to activate the peripheral.
8. The Get Peripheral Offsets dialog will now be showing.



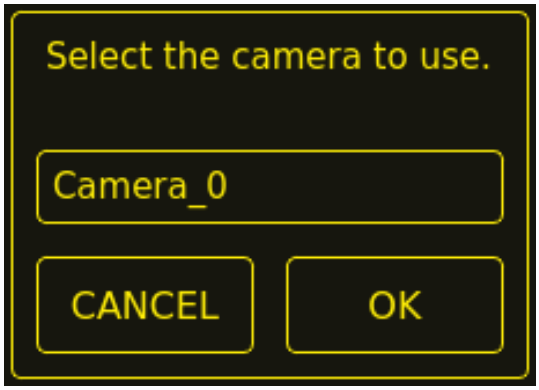
9. Raise the Z axis so the torch and peripheral are clear of the material.
10. Jog the X/Y axes so that the peripheral is centered in the mark from the torch.
11. Click the GET OFFSETS button to get the offsets and a confirmation dialog will open.



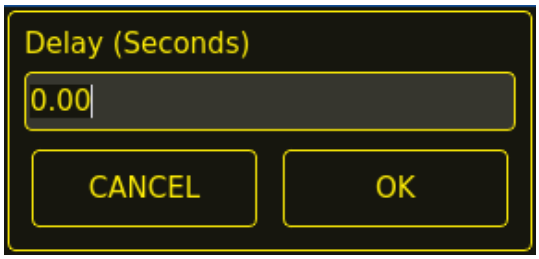
12. Click SET OFFSETS and the offsets will now be saved.

Canceling may be done at any stage by pressing the CANCEL button which will close the dialog, and no changes will be saved.

If CAMERA was selected at item 7 above and more than one camera exists, then a camera selection dialog will show. The appropriate camera needs to be selected before the Get Peripheral Offsets dialog will appear.



If PROBE was selected at item 7 above, then a delay dialog will show prior to the confirmation dialog at item 11. This is for the delay required for the probe to deploy to its working position.



nota
It may be necessary to click the preview window to enable jogging. By following the above procedure, the offsets are available for use immediately and no restart of LinuxCNC is required.

10.8.15.3. Keep Z Motion

By default, QtPlasmaC will remove all Z motion from a loaded G-code file and add an initial Z movement to bring the torch near the top of travel at the beginning of the file. If the user wishes to use their table with a marker, drag knife, diamond scribe, etc. mounted in the torch holder, QtPlasmaC has the ability to retain the Z movements when executing a program by adding the following command in a G-code file:

```
#<keep-z-motion> = 1
```

This can be used two different ways: . In a G-code file with no M3 (cutting, scribing, or spotting) commands. In this case, #<keep-z-motion> = 1 can be placed anywhere before the first Z movement and all subsequent Z motion will be retained. The G-code filter will not add any initial Z movements. . In a G-code file that also contains subsequent spotting and/or cutting operations. In this case, the marking portion of the file that contains Z movements needs to preceed the spotting and/or cutting operations, and the marking section needs to have #<keep-z-motion> = 1 at the beginning and #<keep-z-motion> = 0 at the end. In this case, the G-code filter will automatically add an initial Z movement for the first M3 command after #<keep-z-motion> = 0.

In either case, M3 commands are not supported while #<keep-z-motion> is active.

Omitting #<keep-z-motion>, or setting #<keep-z-motion> to anything but 1 will cause QtPlasmaC the default behavior of stripping all Z motion from a loaded G-code file and adding an initial Z movement to bring the torch near the top of travel before the first M3 command.

10.8.15.4. External HAL Pins

QtPlasmaC creates some HAL pins that may be used to connect a momentary external button or pendant etc.

HAL connections to these HAL pins need to be specified in a postgui HAL file as the HAL pins are not available until the QtPlasmaC GUI has loaded.

The following HAL bit pins are always created. The HAL pin has the identical behaviour of the related QtPlasmaC GUI button.

User Button Function	**Pines HAL**	**GUI Function**
Toggle machine power	`qtplasmac.ext_power`	POWER
Ejecuta el programa en código G cargado	`qtplasmac.ext_run`	CYCLE START
Pausa/reanuda el programa en código G cargado	`qtplasmac.ext_pause`	CYCLE PAUSE/CYCLE RESUME
Pausa el programa en código G cargado	`qtplasmac.ext_pause_only`	CYCLE PAUSE
Reanuda el programa en código G cargado	`qtplasmac.ext_resume`	CYCLE RESUME
Aborta el programa en código G cargado	`qtplasmac.ext_abort`	CYCLE STOP
Touchoff X & Y axes to zero	`qtplasmac.ext_touchoff`	X0Y0
Use a laser to set an origin with or without rotation	`qtplasmac.ext_laser_touch`	LASER
Toggle qtplasmac.laser_on pin	`qtplasmac.ext_laser_toggle`	N/A
Ejecuta/Pausa/Reanuda el programa en código G cargado	`qtplasmac.ext_run_pause`	CYCLE START, CYCLE PAUSE, CYCLE RESUME in sequence
Torch height override plus	`qtplasmac.ext_height_ovr_plus`	OVERRIDE +
Torch height override minus	`qtplasmac.ext_height_ovr_minus`	OVERRIDE -
Torch height override reset	`qtplasmac.ext_height_ovr_reset`	OVERRIDE RESET TO 0.00

User Button Function	Pines HAL	GUI Function
Torch height override scale	qtplasmac.ext_height_ovr_scale	N/A
Toggle jogging speed between fast and slow	qtplasmac.ext_jog_slow	JOGGING FAST/SLOW
Toggle THC enable	qtplasmac.ext_thc_enable	THC ENABLE
Toggle torch enable	qtplasmac.ext_torch_enable	TORCH ENABLE
Toggle corner Lock enable	qtplasmac.ext_cornerlock_enable	VELOCITY ANTI DIVE ENABLE
Toggle voidlock enable	qtplasmac.ext_voidlock_enable	VOID ANTI DIVE ENABLE
Toggle use auto volts	qtplasmac.ext_auto_volts_enable	AUTO VOLTS
Toggle ohmic probe enable	qtplasmac.ext_ohmic_probe_enable	OHMIC ENABLE
Toggle mesh mode	qtplasmac.ext_mesh_mode	MESH MODE
Toggle arc ignore OK	qtplasmac.ext_ignore_arc_ok	IGNORE OK
Forward along the programmed path	qtplasmac.ext_cutrec_fwd	CUT RECOVERY FWD
Reverse along the programmed path	qtplasmac.ext_cutrec_rev	CUT RECOVERY REV
Cancel any Cut Recovery movement	qtplasmac.ext_cutrec_cancel	CUT RECOVERY CANCEL MOVE
Move up	qtplasmac.ext_cutrec_n	CUT RECOVERY arrow up
Move down	qtplasmac.ext_cutrec_s	CUT RECOVERY arrow down
Move right	qtplasmac.ext_cutrec_e	CUT RECOVERY arrow right
Move left	qtplasmac.ext_cutrec_w	CUT RECOVERY arrow left
Move up-right	qtplasmac.ext_cutrec_ne	CUT RECOVERY arrow up-right
Move up-left	qtplasmac.ext_cutrec_nw	CUT RECOVERY arrow up-left
Move down-right	qtplasmac.ext_cutrec_se	CUT RECOVERY arrow down-right
Move down-left	qtplasmac.ext_cutrec_sw	CUT RECOVERY arrow down-left

The following HAL pins which allow the use of an MPG to control height override are always created.

Función	Pines HAL
Enable MPG height control	qtplasmac.ext_height_ovr_count_enable
MPG height change	qtplasmac.ext_height_ovr_counts

The following HAL bit pins are only created if the function is specified in a custom user button. The HAL pin has the identical behaviour of the related custom user button.

User Button Function	Pines HAL
Probe Test	qtplasmac.ext_probe
Torch Pulse	qtplasmac.ext_pulse
Ohmic Test	qtplasmac.ext_ohmic
Change Consumables	qtplasmac.ext_consumables
Framing	qtplasmac.ext_frame_job

The following HAL bit output pins are always created and can be used by either the Toggle HAL Pin or Pulse HAL Pin custom user buttons to change the state of an output.

Pines HAL
qtplasmac.ext_out_0
qtplasmac.ext_out_1
qtplasmac.ext_out_2

10.8.15.5. Hide Program Buttons

If the user has external buttons and/or a pendant that emulates any of the program buttons, CYCLE START, CYCLE PAUSE, or CYCLE STOP then it is possible to hide any or all of these GUI program buttons by adding the following options to the **[GUI_OPTIONS]** section of the <*machine_name*>.prefs file:

```
Hide run = True
Hide pause = True
Hide abort = True
```

For the 16:9 or 4:3 GUIs, the hiding of each of these GUI buttons will expose two more custom user buttons in the GUI.

10.8.15.6. Tuning Mode 0 Arc OK

Mode 0 Arc OK relies on the arc voltage to set the Arc OK signal. This is accomplished by sampling the arc voltage every servo thread cycle. There needs to be a specified number of consecutive samples, all within a specified threshold for the Arc OK signal to be set. These voltages are also required to be within a specified range.

There are two settings in the PARAMETERS Tab for setting the range, these are:

- **OK High Volts** which is the upper value of the voltage range. The default is 250 V.
- **OK Low Volts** which is the lower value of the voltage range. The default is 60 V.

Both of these values may be changed by direct entry or by the use of the increment/decrement buttons.

There are also two HAL pins that have been provided to allow the user to tune the set point. These HAL pins are:

- `plasmac.arc-ok-counts` which is the number of consecutive readings within the threshold that are required to set the Arc OK signal. The default is 10.
- `plasmac.arc-ok-threshold` which is the maximum voltage deviation that is allowed for a valid voltage to set the Arc OK signal. The default is 10.

The following example would set the number of valid consecutive readings required to 6:

```
setp plasmac.arc-ok-counts 6
```

These settings if used should be in the custom.hal file of the configuration.

10.8.15.7. Lost Arc Delay

Some plasma power sources/machine configurations may lose the Arc OK signal either momentarily during a cut, or permanently near the end of a cut causing QtPlasmaC to pause the program and report a ”valid arc lost” error.

There is a HAL pin named `plasmac.arc-lost-delay` that may be used to set a delay (in seconds) that will prevent a paused program/error if the lost Arc OK signal is regained, or the **M5** command is reached before the set delay period expires.

It is important to note that the THC will be disabled and locked at the cutting height at the time the Arc OK signal was lost.

El siguiente código establecerá un retardo de 0.1 segundos:

```
setp plasmac.arc-lost-delay 0.1
```

It is recommended that the user set this pin in the custom.hal file.

This setting should only be used if the user experiences the above symptoms. It should also be noted that the user could use the appropriate Ignore Arc OK G-code commands to achieve a similar result.

10.8.15.8. Zero Window

Small fluctuations in the arc voltage displayed while the machine is at idle are possible depending on many different variables (electrical noise, incorrect THCAD tuning, etc.).

After all contributing factors have been mitigated, if a small fluctuation still exists it is possible to eliminate it by widening the voltage window for which QtPlasmaC will display 0 V.

The pin for adjusting this value is named plasmac.zero-window and the default value is set to 0.1. To change this value, add the pin and the required value to the custom.hal file.

El ejemplo siguiente configura que la ventana de voltaje muestre 0 V desde -5 V hasta +5 V:

```
setp plasmac.zero-window 5
```

10.8.15.9. Tuning Void Sensing

In addition to the **Void Slope** setting in the PARAMETERS Tab there are two HAL pins to aid in the fine tuning of void anti-dive. These HAL pins are:

- **plasmac.void-on-cycles** which is the number of times the slope rate needs to be exceeded to activate void anti-dive. The default is 2.
- **plasmac.void-off-cycles** which is the number of cycles without the slope rate being exceeded to deactivate void anti-dive. The default is 10.

El ejemplo siguiente establecerá el número de ciclos requerido a 3:

```
setp plasmac.void-on-cycles 3
```

The objective is to have as low a value of Void Slope as possible without any false triggering then adjust on and off cycles to ensure clean activation and deactivation of void anti-dive. In most cases it should not be necessary to change on and off cycles from the default value.

These settings if used should be in the custom.hal file of the configuration.

10.8.15.10. Max Offset

Max Offset is the distance (in millimeters) away from the Z MAX_LIMIT that QtPlasmaC will allow the Z axis to travel while under machine control.

The pin for adjusting this value is named plasmac.max-offset and the default value (in millimeters) is set to 5. To change this value, add the pin and the required value to the custom.hal file. It is not recommended to use values less than 5 mm as offset overrun may cause unforeseen issues.

El ejemplo siguiente asigna la distancia de Z MAX_LIMIT a 10 mm:

```
setp plasmac.max-offset 10
```

10.8.15.11. Enable Tabs During Automated Motion

By default, all tabs except the MAIN Tab are disabled during automated motion. It is possible for every tab but the CONVERSATIONAL Tab to be enabled during automated motion by setting the following HAL pin True:

```
setp qtplasmac.tabs_always_enabled 1
```



aviso
It is the responsibility of the operator to ensure that the machine is equipped with a suitable, working hardware E-stop. If using only a touchscreen to navigate the QtPlasmaC GUI, there is no way to stop automated machine motion on any tab but the MAIN tab.

10.8.15.12. Override Jog Inhibit Via Z+ Jog

It is possible to override the jog inhibit by using the GUI or keyboard to jog in the Z+ direction rather than checking the Override Jog box on the SETTINGS Tab.

This is done by changing the following option to **True** in the **[GUI_OPTIONS]** of the <*machine_name*>.prefs file in the <*machine_name*> folder:
`Override jog inhibit via Z+`

10.8.15.13. QtPlasmaC State Outputs

The plasmac HAL component has a HAL pin named **plasmac.state-out** which can be used to interface with user-coded components to provide the current state of the component.

Cuadro 10.47: Different states QtPlasmaC could encounter

Estado	Name	Descripción
0	IDLE	inactivo y esperando a comando de inicio
1	PROBE_HEIGHT	move down to probe height
2	PROBE_DOWN	probe down until material sensed
3	PROBE_UP	probe up until material not sensed, this sets the zero height
4	ZERO_HEIGHT	not used at present
5	PIERCE_HEIGHT	move up to pierce height
6	TORCH_ON	turn the torch on
7	ARC_OK	wait until arc OK detected
8	PIERCE_DELAY	wait for pierce delay time
9	PUDDLE_JUMP	xy motion begins, move to puddle jump height
10	CUT_HEIGHT	move to cut height
11	CUT_MODE_01	cutting in either mode 0 or mode 1
12	CUT_MODE_2	cortando en modo 2
13	PAUSE_AT_END	pause motion at end of cut
14	SAFE_HEIGHT	move to safe height
15	MAX_HEIGHT	move to maximum height
16	END_CUT	end the current cut
17	END_JOB	end the current job
18	TORCHPULSE	un pulso de antorcha está activo
19	PAUSED_MOTION	cut recovery motion is active while paused
20	OHMIC_TEST	an ohmic test is active

Cuadro 10.47: (continued)

Estado	Name	Descripción
21	PROBE_TEST	a probe test is active
22	SCRIBING	a scribing job is active
23	CONSUMABLE_CHANGE_ON	move to consumable change coordinates
24	CONSUMABLE_CHANGE_OFF	return from consumable change coordinates
25	CUT_RECOVERY_ON	cut recovery is active
26	CUT_RECOVERY_OFF	cut recovery is deactivated

The DEBUG state is for testing purposes only and will not normally be encountered.

10.8.15.14. QtPlasmaC Debug Print

The plasmac HAL component has a HAL pin named **plasmac.debug-print** which if set to 1 (true) will print to terminal every state change as a debug aid.

10.8.15.15. Hypertherm PowerMax Communications

Communications can be established with a Hypertherm PowerMax plasma cutter that has a RS485 port. This feature enables the setting of **Cut Mode**, **Cutting Amperage** and **Gas Pressure** automatically from the **Cut Parameters** of the material file. In addition, the user will be able to view the PowerMax's **Arc On Time** in hh:mm:ss format on the STATISTICS Tab.

If **Gas Pressure** is set to zero, then the PowerMax will automatically calculate the required pressure from the **Cut Mode**, **Cut Current**, torch type, and torch length.

Changing the cutting mode will set the gas pressure to zero causing the machine to use its automatic gas pressure mode.

The maximum and minimum values of these parameters are read from the plasma cutter and the related spin-buttons in the Cut Parameters are then limited by these values. Gas pressure cannot be changed from zero until communications have been established.

This feature is enabled by setting the correct port name for the PM_PORT option in the **[POWERMAX]** section of the <*machine_name*>.prefs file. If the PM_PORT option is not set in the <*machine_name*>.prefs file then the widgets associated with this feature will not be visible.

Example showing enabling the Hypertherm PowerMax Communications on USB0:

```
[POWERMAX]
Port = /dev/ttyusb0
```

If the user is unsure of the name of the port, there is a Python script in the configuration directory that will show all available ports and can also be used to test communications with the plasma unit prior to enabling this feature in the QtPlasmaC GUI.

To use the test script, follow these instructions:

For a package installation (Buildbot) enter the following command in a terminal window:

```
pmx485-test
```

For a run in place installation enter the following two commands in a terminal window:

```
source ~/linuxcnc-dev/scripts/rip-environment
pmx485-test
```

The gas pressure units display (psi or bar) is determined by the data received during initial setup of the communication link and is then shown next to the Gas Pressure setting in the MATERIAL section of the PARAMETERS Tab.

The PowerMax machine will go into remote mode after communications have been established and may only be controlled remotely (via the QtPlasmaC GUI) at this point. The connection can be validated by observing the PowerMax display.

To switch the PowerMax back to local mode the user can either:

1. Disable PowerMax Comms from the MAIN Tab
2. Close LinuxCNC which will put the PowerMax into local mode during shutdown.
3. Turn the PowerMax off for 30 seconds and then power it back on.

---

sugerencia
If PowerMax communications is active then selecting Mesh Mode will automatically select CPA mode on the PowerMax unit.

---

nota
To use the PowerMax communications feature it is necessary to have the Python pyserial module installed.
If pyserial is not installed an error message will be displayed.

To install pyserial, enter the following command into a terminal window:

```
sudo apt install python3-serial
```

A typical connection diagram is shown in the appendix of this document as well as confirmed working interfaces.

10.8.15.16. Moving Pierce

A moving pierce allows the torch to move during the pierce delay period. This has an advantage in that it allows for thicker materials to be pierced than can be achieved with a stationary pierce. It may also support longer consumable life by allowing a style of motion that helps prevent molten material being sprayed up into the torch nozzle.

Through the use of M159 a moving pierce can be configured.

La sintaxis para el comando M159 es como sigue:

```
M159 Pn Qn
```

Action Code (P)	Action	Descripción	Value (Q)
601	Pierce Type	0=Normal, 1=Wiggle, 2=Ramp	0,1,2
602	Pierce Motion Delay	Delay before Z motion starts to Pierce End Height. Expressed as a % of Pierce Delay.	Integer 0 to 100

Action Code (P)	Action	Descripción	Value (Q)
603	Pierce End Height	Target pierce height at end of Pierce Delay. Normally lower than Pierce Height. Expressed in machine units.	Float
604	Cut Height Delay	Delay at the end of transition to Pierce End Height before transition to Cut Height. Expressed in seconds.	Float
605	Gouge Speed	Velocidad de gouge. Expresada en unidades de máquina/min.	Float
606	Gouge Distance	Longitud de gouge. Expresada en unidades de máquina.	Float
607	Creep Speed	Velocity of creep which takes effect after gouge has finished. Expressed in machine units.	Float
608	Creep Distance	Longitud de creep. Expresada en unidades de máquina.	Float
609	Reset	Resets the values for action codes 601-608 back to 0, returning to default behaviour.	Not Required

The available moving pierce models are as follows:

The model supported is the same as that created by Sheetcam’s wiggle pierce. Given a straight lead-in to the main cut the wiggle pierce is expected to move back and forth for some distance along the lead-in. Strictly speaking this straight movement is arbitrary. Technically any X/Y motion is available during the pierce delay, and it is up the CAM tooling or the user to program.

The constraint is that this motion is expected to be completed during the pierce delay value. If not then the torch will transition to normal cut height on completion of pierce delay and potentially before the wiggle motion is completed.

Therefore, the length of the wiggle and the feed rate need to be considered in calculating the pierce delay, or the size of the wiggle constrained based on feed rate and pierce delay.

For example:

- A feed rate of 1080 mm/min (18 mm/s).
- A wiggle movement of 4 mm for 3 oscillations.

This means that the length of the wiggle is 4 x 3 = 12 mm. At the 18 mm/s feed rate, the pierce delay needs to be approx 0.7 seconds to support the wiggle distance at pierce height.

The G-code needed invoke this behaviour is:

```
M159 P601 Q1
```

The G-code needed to reset to standard behaviour is:

```
M159 P609
```

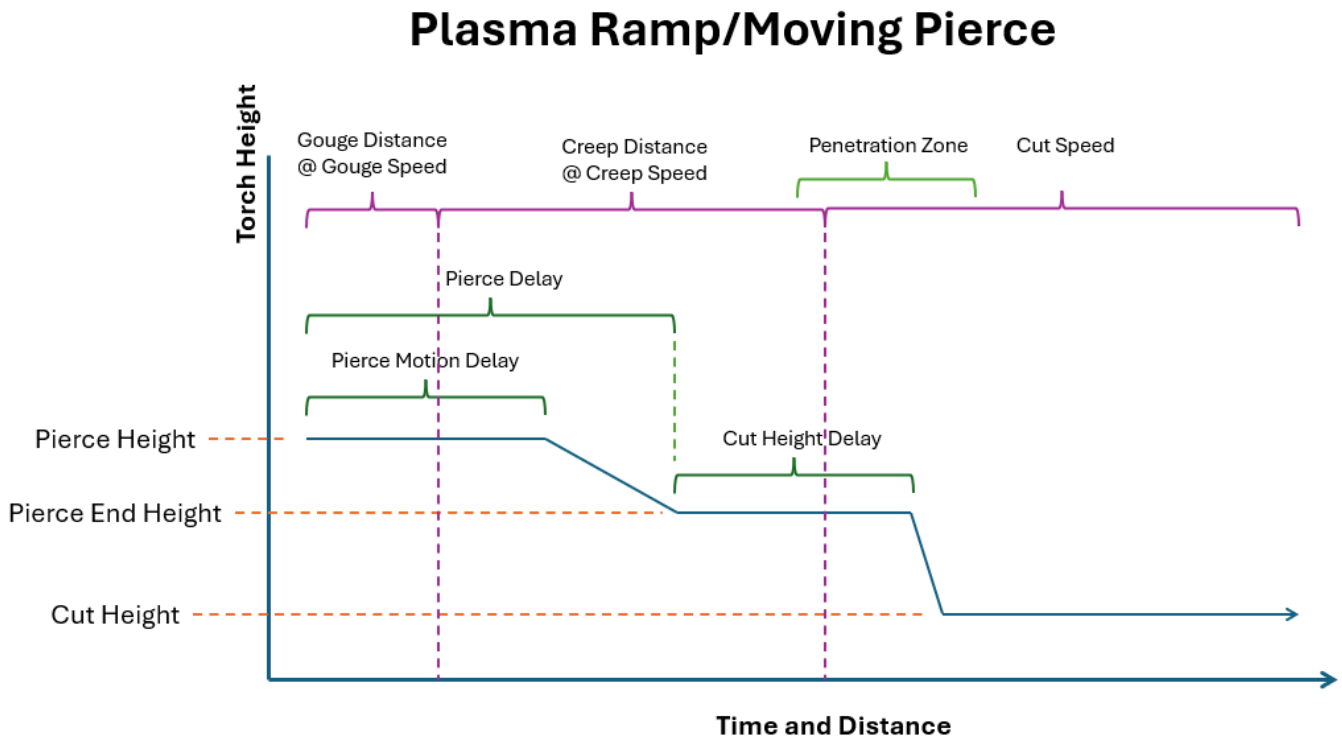
A ramping pierce combines a range of parameters so as to generate a sloped trough that causes the molten material to be evacuated. The resulting evacuation of material is sometimes likened to a ”rooster tail” as it is very directional. Careful consideration of the lead-in can allow evacuation of material in a safe direction for workers and machine components.

As the elements combine to drive the shape of the ramping pierce it is key that all of these elements are carefully considered when designing the ramp pierce and the parameters that set it. Inevitably

there will need to be experimentation to build recipes that work with the plasma power source being used in conjunction with the material to be cut.

Items to consider:

1. The gouge and creep speeds and distances in relation to the sum of Pierce Delay from the material and the Cut Height Delay action.
2. Pierce height from the material and the End Pierce Height action in relation to Pierce Delay from the material and the speeds in effect over the various distances during the Pierce Delay time.
3. The plasma source manufacturer’s cut charts for the material type and thickness.



As with wiggle pierce it is up the CAM tooling or the user to program a ramping pierce.

Below is a sample of code to setup a ramp pierce:

```
(o=0,kw=2, ph=4, pd=1, ch=1.5, fr=490, th=1, cv=99, pe=0.3, jh=0, jd=0)

M159 P601 Q2
M159 P602 Q50
M159 P603 Q2.5
M159 P604 Q1
M159 P605 Q980
M159 P606 Q5
M159 P607 Q245
M159 P608 Q3
```

This code shows us the following information in order:

1. Comentario de magia material.
   - La altura de perforación es 4 mm.

- Pierce Delay is 1 second.
- La altura de corte es 1.5 mm.
- Cut Feed Rate is 490 mm/min.

2. Mode is set to 2 which is ramp pierce.
3. Pierce Motion Delay is 50 % of Pierce Delay (0.5 seconds).
4. Pierce End Height is 2.5 mm
5. Cut Height Delay is 1 second.
6. Gouge Speed is 980 mm/min.
7. Gouge Distance is 5 mm
8. La velocidad de arrastre es 245 mm/min.
9. Distancia de arrastre es 3 mm.

With this information the following behavior can be described:

It is important to note that accelerations and decelerations are omitted from the following calculations.

The torch will start at Pierce Height (4 mm from the material) and start traveling at a Gouge Speed of 980 mm/min (16.3 mm/s) for a Gouge Distance of 5 mm which will consume 0.3 seconds (5 mm / 16.3 mm/s = 0.3 s) of the 0.5 s Pierce Motion Delay.

When the Gouge Distance is reached, the torch speed is set to a Creep Speed of 245 mm/min (4 mm/s) for a Creep Distance of 3 mm. The Creep Distance will take roughly 0.7 seconds to complete (3 mm / 4 mm/s = 0.7 s).

The torch height will remain at 4 mm for another 0.2 seconds (0.5 s (Pierce Motion Delay) - 0.3 s (Gouge Distance at Gouge Speed) = 0.2 s) after which the torch will begin descending to a Pierce End Height of 2.5 mm over the remaining 0.5 seconds of the material's Pierce Delay. Since there are 0.5 seconds of the material's Pierce Delay remaining, as well as 0.5 seconds left at Creep Speed, the Creep Distance will be covered at the same time the Pierce End Height is reached.

When the Creep Distance has been reached, the torch speed will be set to the material's Cut Feed Rate of 490 mm/min. Since there is a 1 second Cut Height Delay that started at the end of the material's Pierce Delay the transition to a Cut Height of 1.5 mm will occur after the remaining 1 second Cut Height Delay has expired.

The above text should demonstrate that there is quite a bit of configuration and subtlety can be achieved through experimentation and the careful use of different parameter combinations.

10.8.16. Internationalisation

It is possible to create translation files for QtPlasmaC to display in the language of the current locale.

To create and or edit a translation file requires that LinuxCNC has been installed as run in place.

The following assumes that the LinuxCNC git directory is ~/linuxcnc-dev.

All language files are kept in ~/linuxcnc-dev/share/screens/qtplasmac/languages.

The qtplasmac.py file is a Python version of the GUI file used for translation purposes.

The .ts files are the translation source files for the translations. These are the files that require creating/editing for each language.

The .qm files are the compiled translation files used by PyQt.

The directories qtplasmac_4x3/languages and qtplasmac_9x16/languages are only for links to the .qm files in qtplasmac/languages.

The language is determined by an underscore plus the first two letters of the locale. For example, if an Italian translation was being done then it would be `_it`. It will be referred to as `_xx` in this document, so `qtplasmac_xx.ts` in this document would actually be `qtplasmac_it.ts` for an Italian translation.

The default locale for QtPlasmaC is `_en` which means that any translation files created as `qtplasmac_en.*` will not be used for translations.

If any of the required utilities (pyuic5, pylupdate5, linguist) are not installed then the user will need to install the required development tools:

```
sudo apt install qttools5-dev-tools pyqt5-dev-tools
```

Cambiar al directorio de idiomas:

```
cd ~/linuxcnc-dev/share/qtvcp/screens/qtplasmac/languages
```

If any text changes have been made to the GUI then run the following to update the GUI Python file:

```
pyuic5 ../qtplasmac.ui > qtplasmac.py
```

The user can either create a new translation source file for a non-existing language translation or modify an existing translation source file due to changes being made to some text in a QtPlasmaC source file. If modifying an existing translation that has had no source file changes then this step is not required.

Crear o editar un archivo .ts:

```
./langfile xx
```

---

nota
this command is a script which runs the following: $ pylupdate5 **.py ../**.py ../../../../../lib/python/qtvcp/lib/qtplasmac/*.py -ts qtplasmac_xx.ts

---

The editing of the translation is done with the linguist application:

```
lingüista
```

1. Open the TS file and translate the strings

It is not necessary to provide a translation for every text string, if no translation is specified for a string, then the original string will be used in the application. The user needs to be careful with the length of strings that appear on widgets as space is limited. If possible try, to make the translation no longer than the original.

Cuando termine de editar guarde el archivo:
`Archivo -> Guardar`

Then create the .qm file:
`File -> Release`

Lingüista cercano.

Then create links to the compiled .qm file for the other QtPlasmaC GUIs.

```
$ ./langlink xx
```

---

nota
This command is a script which creates a link in both qtplasmac_4x3/languages and qtplasmac_9x16/languages to the above .qm file and then renames the link to match the GUI name.

QtPlasmaC will be translated to the language of the current locale on the next start so long as a .qm file exists in that language.

Users are welcome to submit translation files for inclusion into QtPlasmaC. An easy method is to post the up to date qtplasmac_xx.ts file on the forum and the maintainers will install the translations.

The preferred method is to submit a pull request from the users GitHub account as described in the contributing to LinuxCNC documentation. The files required to be committed are qtplasmac_xx.ts and qtplasmac_xx.qm in the qtplasmc/languages directory plus the links in both the qtplasmac_4x3/languages and qtplasmac_9x16/languages directories.

10.8.17. Appendix

10.8.17.1. Example Configurations

There are example configuration files which use the QtPlasmaC GUI to simulate plasma cutting machines.

They can be found in the LinuxCNC chooser under: Sample Configurations -> sim -> qtplasmac

Three versions are available in both metric and imperial units:

1. qtplasmac_l - 16:9 format, minimum resolution 1366x768
2. qtplasmac_p - 9:16 format, minimum resolution 786x1366
3. qtplasmac_s - 4:3 format, minimum resolution 1024x768

Each sample configuration includes a popup control panel to simulate various inputs to the GUI such as:

1. ARC VOLTAGE
2. OHMIC SENSE
3. FLOAT SWITCH
4. BREAKAWAY SWITCH
5. ESTOP

10.8.17.2. NGC Samples

There are some sample G-code files in the ~/linuxcnc/nc_files/examples/plasmac directory.

10.8.17.3. QtPlasmaC Specific G-codes

Descripción	Código
Begin cut	`M3 $0 S1`
End cut	`M5 $0`
Begin scribe	`M3 $1 S1`
End scribe	`M5 $1`
Begin center spot	`M3 $2 S1`
End center spot	`M5 $2`
End all the above.	`M5 $-1`

Descripción	Código
Select a material.	`M190 P`*n* *n* denotes the material number.
Wait for material change confirmation.	`M66 PG L3 Q`*n* + *n* is delay time (in seconds). This value may need to be increased for very large material files.
Set feed rate from material.	`F#<_hal[plasmac.cut-feed-rate]>`
Enable Ignore Arc OK	`M62 P1` (synchronized with motion) `M64 P1` (immediate)
Disable Ignore Arc OK	`M63 P1` (synchronized with motion) `M65 P1` (immediate)
Disable THC	`M62 P2` (synchronized with motion) `M64 P2` (immediate)
Enable THC	`M63 P2` (synchronized with motion) `M65 P2` (immediate)
Disable Torch	`M62 P3` (synchronized with motion) `M64 P3` (immediate)
Enable Torch	`M63 P3` (synchronized with motion) `M65 P3` (immediate)
Set velocity to a percentage of feed rate.	`M67 E3 Q`*n* (synchronized with motion) `M68 E3 Q`*n* (immediate) *n* is the percentage to set 10 is the minimum, below this will be set to 100 % 100 is the maximum, above this will be set to 100 % **It is recommended to have M68 E3 Q0 in both the preamble and postamble.**
Cutter compensation - left of path	`G41.1 D#<_hal[plasmac.kerf-width]>`
Cutter compensation - right of path	`G42.1 D#<_hal[plasmac.kerf-width]>`
Cutter compensation off	`G40`+ **Note que de M62 a M68 son inválidos mientras la compensación del cortador está activada.**
Cut holes at 60 % feed rate	`#<holes> = 1` for holes less than 32 mm (1.26”) diameter
Cut holes at 60 % feed rate, turn torch off at hole end, continue hole path for over cut.	`#<holes> = 2` for holes less than 32 mm (1.26”) diameter over cut length = 4 mm (0.157”)
Cut holes and arcs at 60 % feed rate.	`#<holes> = 3` for holes less than 32 mm (1.26”) diameter for arcs less than 16 mm (0.63”) radius
Cut holes and arcs at 60 % feed rate, turn torch off at hole end, continue hole path for over cut.	`#<holes> = 4` for holes less than 32 mm (1.26”) diameter for arcs less than 16 mm (0.63”) radius over cut length = 4 mm (0.157”)
Specify hole diameter for #<holes> = 1-4.	`#<h_diameter> =` *n* (*n* is the diameter, use the same units system as the rest of the G-code file)
Specify hole velocity for #<holes> = 1-4.	`#<h_velocity> =` *n* (*n* is the percentage, set the percentage of the current feed rate)
Specify over cut length.	`#<oclength> =` *n* (*n* is the length, use the same units system as the rest of the G-code file)
Specify pierce-only mode.	`#<pierce-only> =` *n* (*n* is the mode, 0=normal cut mode, 1=pierce only mode)

Descripción	Código
Create or edit materials. Options: 0 - Create temporary default 1 - Add if not existing 2 - Overwrite if existing else add new	mandatory parameters: (o=<*option*>, nu=<*nn*>, na=<*ll*>, ph=<*nn*>, pd=<*nn*>, ch=<*nn*>, fr=<*nn*>) optional parameters: (kw=<*nn*>, th=<*nn*>, ca=<*nn*>, cv=<*nn*>, pe=<*nn*>, gp=<*nn*>, cm=<*nn*>, jh=<*nn*>, jd=<*nn*>)
Keep Z Motion	`#<keep-z-motion> = 1`

10.8.17.4. QtPlasmaC G-code Examples

Descripción	Ejemplo

Select material and do a normal cut	`M190 P3`
	`M66 P3 L3 Q1`
	`F#<_hal[plasmac.cut-feed-rate]>`
	`M3 $0 S1`
	`.`
	`.`
	`M5 $0`

Set velocity to 100 % of CutFeedRate	`M67 E3 Q0 or M67 E3 Q100`
Set velocity to 60 % of CutFeedRate	`M67 E3 Q60`
Set velocity to 40 % of CutFeedRate	`M67 E3 Q40`

Cut a hole with 60 % reduced speed using velocity setting	`G21 (metric)`
	`G64 P0.05`
	`M52 P1 (enable adaptive feed)`
	`F#<_hal[plasmac.cut-feed-rate]>`
	`G0 X10 Y10`
	`M3 $0 S1 (start cut)`
	`G1 X0`
	`M67 E3 Q60 (reduce feed rate to 60%)`
	`G3 I10 (the hole)`
	`M67 E3 Q100 (restore feed rate to 100%)`
	`M5 $0 (end cut)`
	`G0 X0 Y0`
	`M2 (end job)`

Cut a hole with 60 % reduced speed using the #<holes> command	`G21 (metric)`
	`G64 P0.05`
	`M52 P1 (enable adaptive feed)`
	`#<holes> = 1 (velocity reduction for holes)`
	`F#<_hal[plasmac.cut-feed-rate]>`
	`G0 X10 Y10`
	`M3 $0 S1 (start cut)`
	`G1 X0`
	`G3 I10 (the hole)`
	`M5 $0 (end cut)`
	`G0 X0 Y0`
	`M2 (end job)`

Descripción	Ejemplo
Cut a hole with over cut using torch disable	G21 (metric) G64 P0.05 M52 P1 (enable adaptive feed) F#<_hal[plasmac.cut-feed-rate]> G0 X10 Y10 M3 $0 S1 (start cut) G1 X0 M67 E3 Q60 (reduce feed rate to 60%) G3 I10 (the hole) M62 P3 (turn torch off) G3 X0.8 Y6.081 I10 (continue motion for 4 mm) M63 P3 (allow torch to be turned on) M67 E3 Q0 (restore feed rate to 100%) M5 $0 (end cut) G0 X0 Y0 M2 (end job)
Cut a hole with over cut using the #<holes> command	G21 (metric) G64 P0.05 M52 P1 (enable adaptive feed) #<holes> = 2 (over cut for holes) F#<_hal[plasmac.cut-feed-rate]> G0 X10 Y10 M3 $0 S1 (start cut) G1 X0 G3 I10 (the hole) M5 $0 (end cut) G0 X0 Y0 M2 (end job)
Cut a hole with 6.5 mm over cut using the #<holes> command	G21 (metric) G64 P0.05 M52 P1 (enable adaptive feed) #<holes> = 2 (over cut for holes) <oclength> = 6.5 (6.5 mm over cut length) F<_hal[plasmac.cut-feed-rate]> G0 X10 Y10 M3 $0 S1 (start cut) G1 X0 G3 I10 (the hole) M5 $0 (end cut) G0 X0 Y0 M2 (end job)
Select scribe and select torch at end of scribing	. . M52 P1 (enable adaptive feed) F#<_hal[plasmac.cut-feed-rate]> T1 M6 (select scribe) G43 H0 (apply offsets) M3 $1 S1 (start plasmac with scribe) . . T0 M6 (select torch) G43 H0 (apply offsets) G0 X0 Y0 (parking position) M5 $1 (end)

Descripción	Ejemplo

Localización de centro de agujero.	`(Requires a small motion command or nothing happens)`
	`G21 (metric)`
	`F99999 (high feed rate)`
	`G0 X10 Y10`
	`M3 $2 S1 (spotting on)`
	`G91 (relative distance mode)`
	`G1 X0.000001`
	`G90 (absolute distance mode)`
	`M5 $2 (spotting off)`
	`G0 X0 Y0`
	`G90`
	`M2`

Create temporary default material	`(o=0, nu=2, na=5mm Mild Steel 40A, ph=3.1, pd=0.1, ch=0.75, fr=3000)`
Edit material, if not existing create a new one	`(o=2, nu=2, na=5mm Mild Steel 40A, ph=3.1, pd=0.1, ch=0.75, fr=3000, kw=1.0)`

10.8.17.5. Mesa THCAD

The Mesa THCAD is a common way of obtaining the arc voltage from a plasma cutter and is also useful for ohmic sensing of the material during probing. The THCAD may be used for parallel port configurations as well as configurations using Mesa Electronics hardware. The THCAD is available in three different models, THCAD-5, THCAD-10, and THCAD-300.

There is a mode jumper on each THCAD card which should be set to **UNIPOLAR**

There is a frequency divider jumper on each THCAD card which should be set according to the hardware type:

Input Device	Recommended Setting
Parallel Port with very low latency	F/32
Parallel Port recommended starting point	F/64
Parallel Port with higher latency, or when cutting thick material	F/128
Mesa Card	F/32

This value is required to be entered into PnCConf during installation.

nota
If using a parallel port it may be necessary for the user to adjust the jumper setting and the subsequent scaling values on the Parameters Tab to achieve optimal results. Symptoms may include random torch raises or dives during otherwise stable cutting. Halscope plots may be useful in diagnosing these issues.

Located on the rear of the THCAD is a calibration sticker showing:

```
   THCAD-nnn

0V  121.1 kHz
5V  925.3 kHz
```

or similar values, these values are required to be entered into PnCConf during installation.

PnCConf has entries for all required THCAD parameters and will calculate and configure any required settings. The calculations used are as follows:

Voltage Scale

```
vs = r / ((f - z) / d / v)
```

Voltage Offset

```
vo = z / d
```

r = divider ratio (see below).

f = full scale value from calibration sticker.

z = 0 V value from calibration sticker.

d = value from jumper above.

v = full scale voltage of THCAD

Divider Ratio *THCAD-5 or THCAD-10*

If connecting to a plasma CNC port, then the divider ratio is selected from the plasma machine. A common ratio used is 20:1.

If connecting to the plasma machines full arc voltage, then a common setup for a THCAD-10 is to use a 1 MΩ resistor from arc negative to THCAD negative and a 1 MΩ resistor from arc positive to THCAD positive. The divider ratio is obtained by:

```
r = (total_resistance + 100000) / 100000
```

THCAD-300

```
r = 1
```



importante
IF THE USER IS USING A HF START PLASMA POWER SUPPLY, THEN EACH OF THESE RESISTANCES SHOULD BE MADE UP OF SEVERAL HIGH VOLTAGE RESISTORS.



atención
IF THE USER IS USING A HF START PLASMA POWER SUPPLY, THEN OHMIC SENSING IS NOT RECOMMENDED.

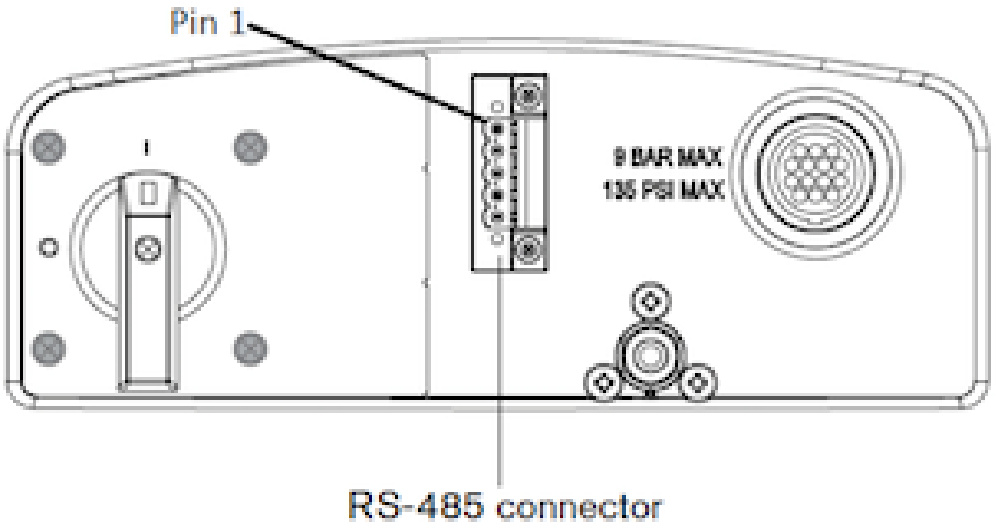
nota
These values can be calculated by using this online calculator.

nota
There is a lowpass filter available which may be useful if using a THCAD and there is a lot of noise on the returned arc voltage.

10.8.17.6. RS485 Connections

Hypertherm RS485 Wiring Diagram (wire colors inside the Hypertherm in parentheses):

Connection at Machine Pin #	Connection at Breakout Board
1 - Tx+ (Red)	->RXD+
2 - Tx- (Black)	->RXD-
3 - Rx+ (Brown)	->T/R+
4 - Rx- (White)	->T/R-
5 - GND (Green)	->GND



RS485 interfaces that are known to work:

DTECH DT-5019 USB to RS-485 converter adapter:

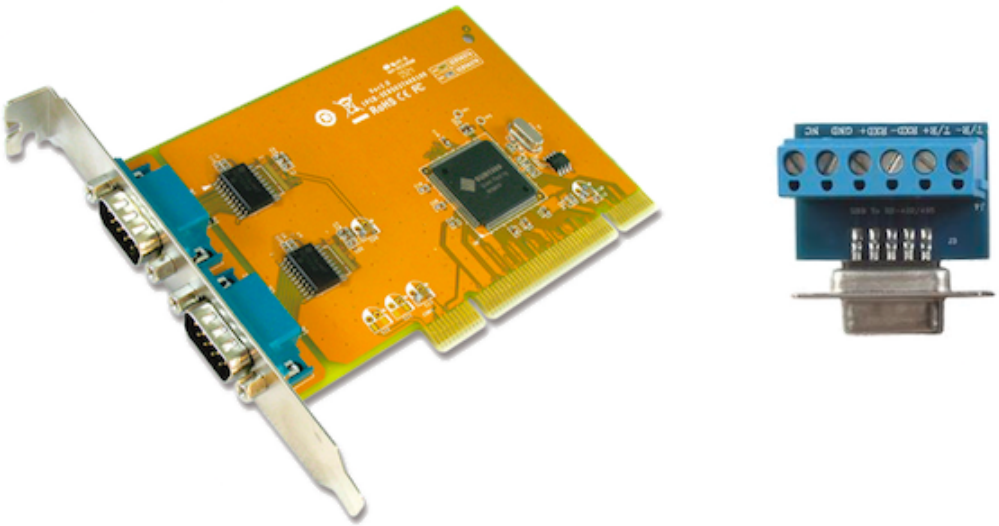


The following is necessary to convert a motherboard Serial connection or Serial card (RS232) to RS485:

DTECH RS-232 to RS-485 converter:



Serial card example (Sunnix SER5037A PCI Card shown with Breakout Board):

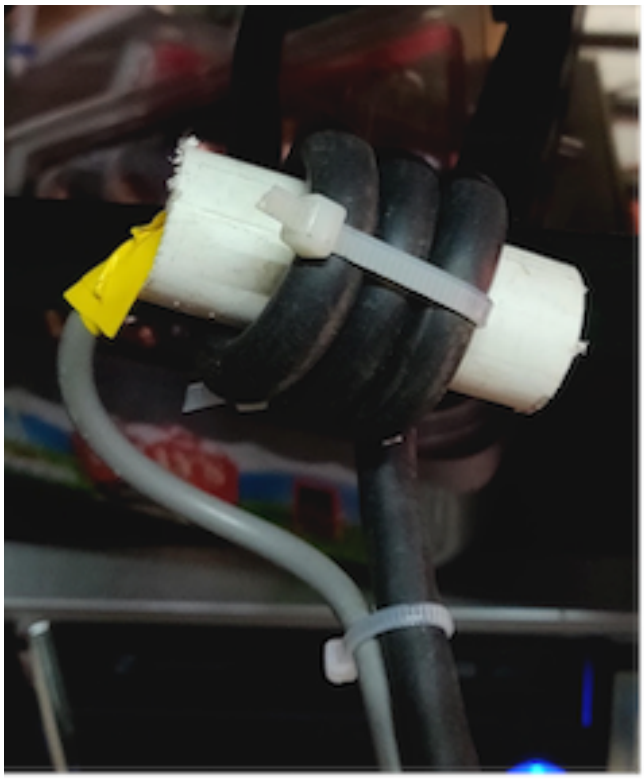


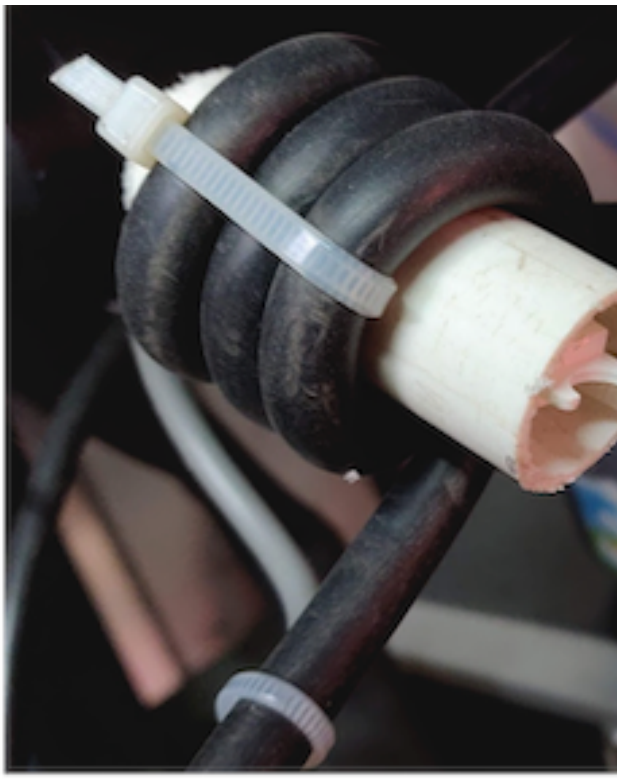
10.8.17.7. Arc OK With A Reed Relay

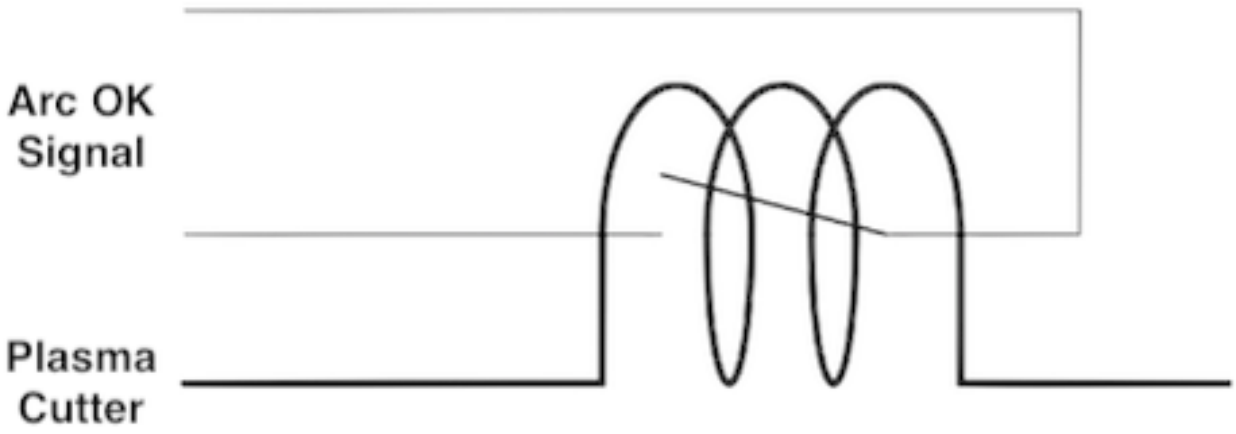
An effective and very reliable method of obtaining an Arc OK signal from a plasma power supply without a CNC port is to mount a reed relay inside a non-conductive tube and wrap and secure three turns of the work lead around the tube.

This assembly will now act as a relay that will switch on when current is flowing through the work lead which only occurs when a cutting arc has been established.

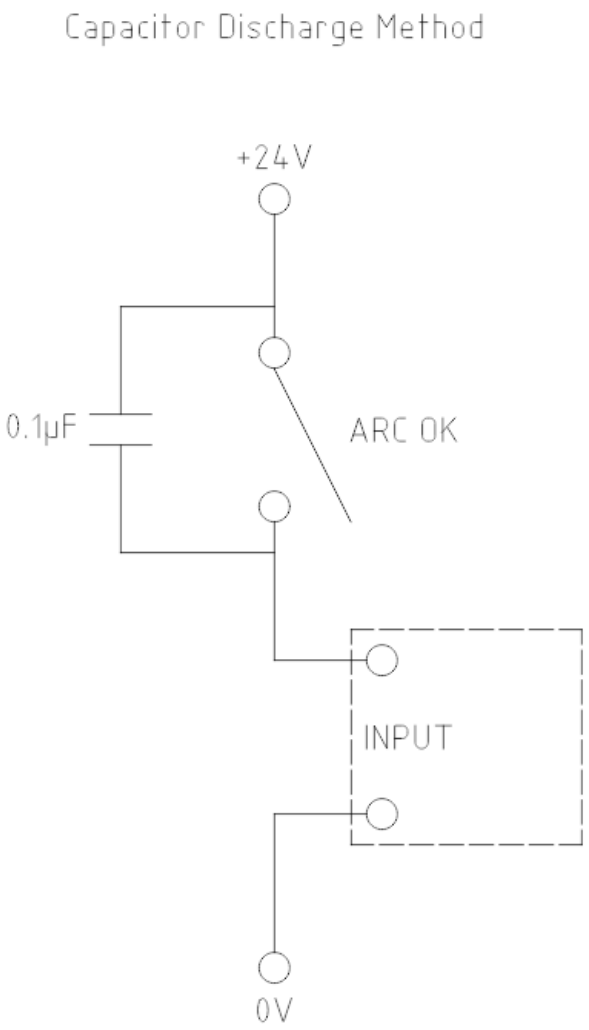
This will require that QtPlasmaC be operated in Mode 1 rather than Mode 0. See the QtPlasmaC Modes sections for more information.

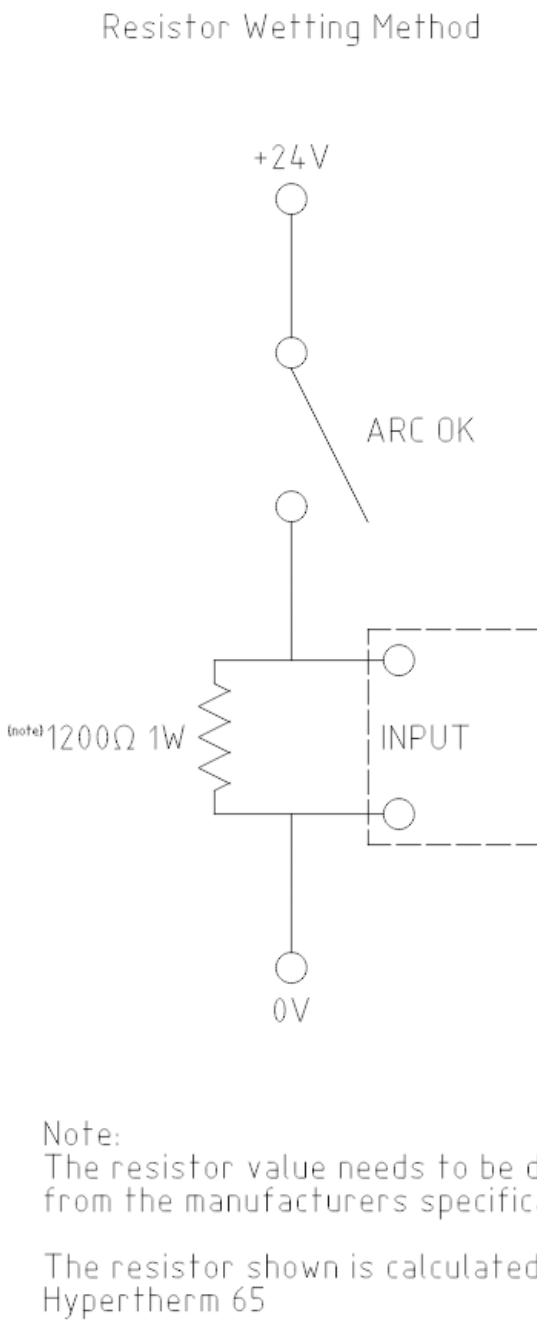





Arc OK
Signal
Plasma
Cutter

10.8.17.8. Contact Load Schematics





A full description is at Contact Load.

10.8.18. Known Issues

10.8.18.1. Keyboard Jogging

There is a known issue with some combinations of hardware and keyboards that may affect the autorepeat feature of the keyboard and will then affect keyboard jogging by intermittent stopping and starting during jogging. This issue can be prevented by disabling the Operating System's autorepeat feature for all keys. QtPlasmaC uses this disabling feature by default for all keys only when the MAIN

Tab is visible, with the following exceptions when autorepeat is allowed with the MAIN Tab visible: G-code editor is active, MDI is active. When QtPlasmaC is shut down, the Operating System's autorepeat feature will be enabled for all keys.

If the user wishes to prevent QtPlasmaC from changing the Operating System's autorepeat settings, enter the following option in the **[GUI_OPTIONS]** section of the <*machine_name*>.prefs file:

```
Autorepeat all == True
```

This issue does not affect any jogging using the GUI jog buttons.

---

nota
Disconnecting and reconnecting a keyboard during an active QtPlasmaC session will cause the auto-repeat feature to re-enable itself automatically which may cause intermittent stopping and starting during jogging. The user must restart QtPlasmaC to disable the autorepeat feature again.

10.8.18.2. NO_FORCE_HOMING

QtPlasmaC does not currently adhere to the following stanza in the <*machine_name*>.ini file:

```
NO_FORCE_HOMING = 1
```

Regardless of this setting, QtPlasmaC requires that the machine must be homed before executing MDI commands or running programs.

10.8.19. Contributing Code To QtPlasmaC

Bugfixes and enhancements to QtPlasmaC are always welcome. The preferred method to contribute code is to submit a pull request (PR) comprised of a single commit to the LinuxCNC GitHub repository. For more information on creating a PR, see the LinuxCNC documentation, the only pre-requisite is that you sign up for a GitHub account. All PR's are verified and then committed by one of the developers. If you are uncomfortable with submitting a PR then attaching the code changes on a LinuxCNC Forum thread is an acceptable method.

Bugfixes are accepted for both the latest released branch and master branch. If a bugfix applies to both branches, then it is only necessary to submit a PR for the latest released branch as it will be merged into master branch by a developer.

Enhancements are accepted for master branch only.

Every PR, except for changes to the QtPlasmaC documentation only, requires that the appropriate version number is incremented and also that the version history is updated. Version numbers are located in the following locations:

Location	**Format**	**Incremented when**
src/hal/components/plasmac.comp	nnn	component code changes
share/qtvcp/screens/qtplasmac/qtplasmac_handler.py	nnn.*nnn*	component code changes *GUI code changes*

The version history is located at share/qtvcp/screens/qtplasmac/versions.html.

10.8.20. Support

Online help and support is available from the PlasmaC section of the LinuxCNC Forum.

The user can create a compressed file containing the complete machine configuration to aid in fault diagnosis by pressing following the directions in the backup section. The resulting file is suitable for attaching to a post on the LinuxCNC Forum to help the community diagnose specific issues.

10.9. MDRO GUI

10.9.1. Introducción

MDRO is a simple graphical front-end for LinuxCNC providing a display of data from Digital Read Out (DRO) scales. It provides functionality similar to a normal machinist's DRO display, allowing the user to use the DRO scales on the machine when operating in a manual-only (hand-cranked) mode. It is most useful for manual machines such as DRO equipped Bridgeport style mills that have been converted to CNC but still have the manual controls.

MDRO is mouse and touch screen friendly.

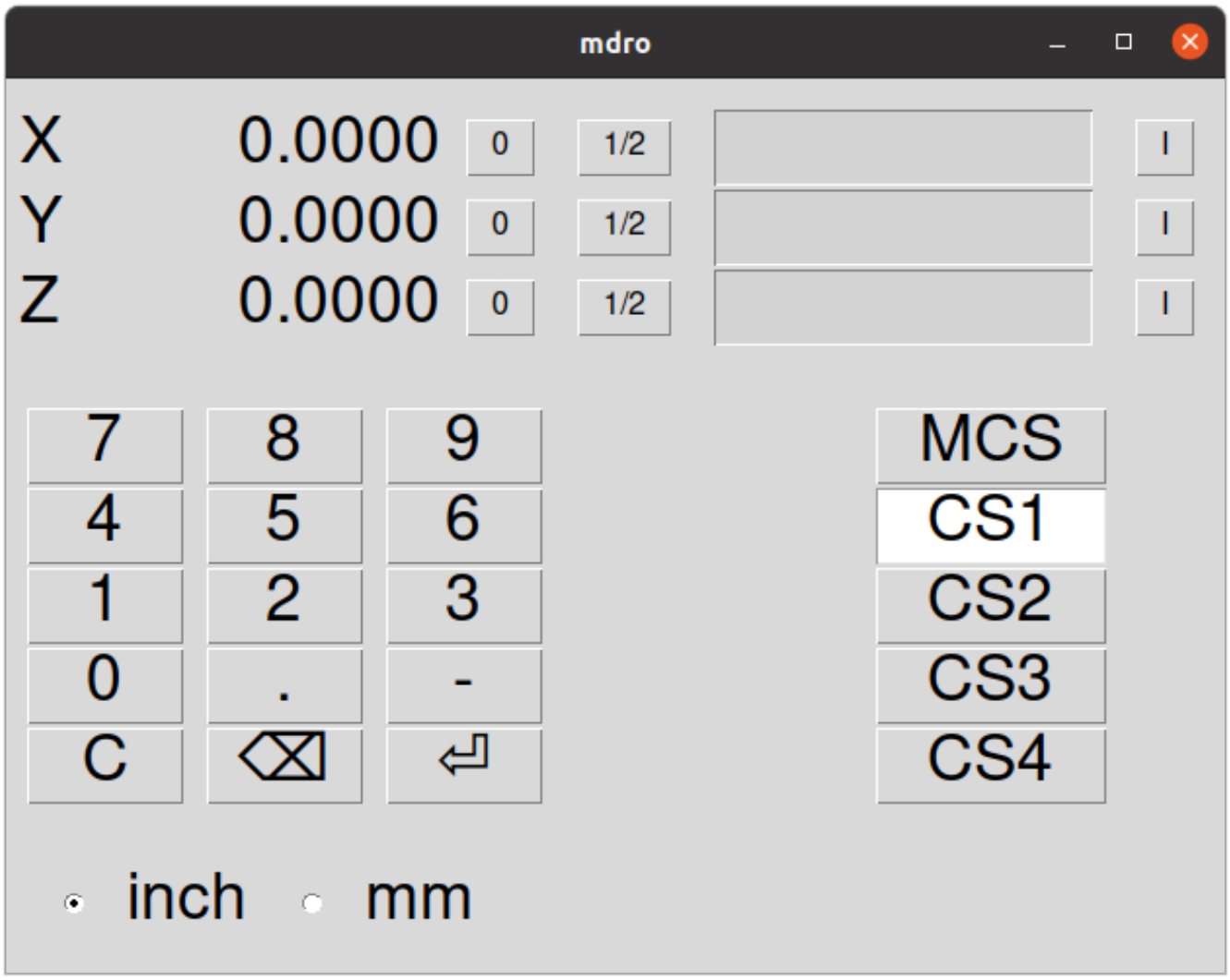


Figura 10.52: MDRO Window

10.9.2. Primeros pasos

If your configuration is not currently set up to use MDRO, you can change it by editing the INI file. In the `[DISPLAY]` section, change the `DISPLAY =` line to read `DISPLAY = mdro`. MDRO defaults to XYZ for the axes but that can be changed. Set `[DISPLAY]` section `GEOMETRY = XYZ` for a 3 axis mill. A lathe with with DRO scales on the X and Z axes might use `GEOMETRY = XZ`.

When MDRO starts, a window like the one in the figure Figura 10.52 above opens.

10.9.2.1. INI File Options

Other options that can be included on the `[DISPLAY]` section include:

- `MDRO_VAR_FILE = <file.var>` - preload G54 - G57 coordinate system data.
  - Preload a .var file. This is typically the .var file used by the operational code.
- `POINT_SIZE = <n>` - Set text point size.

  - This option sets the size of the font used which sets the overall size of the window. The default point size is 20, Typical sizes are 20 to 30.

- `MM = 1` Set this if the DRO scales provide data scaled in millimeters.

10.9.2.2. Command Line Options

MDRO can be started by a `loadusr` command in a HAL file. Options equivalent to those in the INI file can be set on the command line:

- `-l <file.var>` - preload G54 - G57 coordinate system data.
- `-p <n>` - Set text point size.
- `-m` - Set this if the DRO scales provide data scaled in millimeters.
- `<axes>` - axes to display. See `GEOMETRY` above.

10.9.2.3. Pines

Using an example of ”XYZA” for an AXES argument, these pins will be created when MDRO starts:

```
mdro.axis.0
mdro.axis.1
mdro.axis.2
mdro.axis.3
mdro.index-enable.0
mdro.index-enable.1
mdro.index-enable.2
mdro.index-enable.3
```

In this example, the first row of the display will be labeled `X` and will show the data from the DRO scale connected to pin `mdro.axis.0`. The `mdro.index-enable.n` pins should be connected to the index pins of the DRO if the DRO supports them.

The pins must be connected in the file specified in the `POSTGUI_HALFILE` entry of the INI file when the program is started from an INI file. They can be set directly after the loadusr command if the program is started in a HAL file.

10.9.3. MDRO Window

The MDRO window contains the following elements:

- A row for each axis. Each row includes:
  - el nombre del eje,
  - the current value,
  - a ”z” button that zeros the value,
  - a ”1/2” button that halves the value,
  - a entry field that can be used to set a user-defined value. This field can be set from the keyboard or from the on-screen keypad.
  - A ”I” button that starts an index operation (see below),
- a keypad used to set values in the entry field via a mouse or touchscreen,

- botones de selección del sistema de coordenadas:
  - The ”mcs” button selects the machine coordinate system. These are the raw values from the encoders connected to the `mdro.axis.n` pins.
  - The ”cs1” - ”cs4” buttons allow the user to select among one of four user-defined coordinate systems. If the program is started with the `MDRO_VAR_FILE =` option, the labels will be changed to ”g54” - ”g57” and the values from the specified .var file will be preloaded. Note that any changes to the values are not persistent: the .var file is never changed.
- Botones de selección de Pulgada/Milímetro.

10.9.4. Index operations

`MDRO` supports DRO scales with index marks. Hit the ”I” button on the axis row then crank the axis to the index position. The machine coordinate will be zeroed. This is easiest to see at startup or when the ”mcs” coordinate system has been selected.

10.9.5. Simulation

The easiest way to see how `MDRO` works is to try it in a simulation environment. Add this section to the end of your simulation HAL file, usually ”hallib/core_sim.hal”:

```
loadusr -W mdro -l sim.var XYZ
net x-pos-fb  => mdro.axis.0
net y-pos-fb  => mdro.axis.1
net z-pos-fb  => mdro.axis.2
```

Capítulo 11

Programación en G-Code

11.1. Sistemas de coordenadas

11.1.1. Introducción

En este capítulo trataremos de desmitificar a los sistemas de coordenadas. Es un concepto muy importante para comprender la operación de una máquina CNC, su configuración y su uso.

También mostraremos que es muy interesante usar un punto de referencia en el espacio o en la pieza y hacer que el programa trabaje desde este punto, sin tener en cuenta en dónde esta colocada la pieza en la mesa.

Este capítulo describe los offsets tal como los utiliza LinuxCNC. Éstos incluyen:

- Coordenadas de la máquina (G53)
- Nueve offsets del sistema de coordenadas (G54-G59.3)
- Offsets globales (G92) y locales (G52)

11.1.2. Sistema de coordenadas de la máquina

Cuando se inicia LinuxCNC, las posiciones de cada eje definen el origen de la máquina. Cuando se hace home de un eje, el origen de la máquina para ese eje se establece en esa posición home. El origen máquina es el sistema de coordenadas de la máquina en el que se basan todos los demás sistemas de coordenadas. El código G53 puede usarse para moverse en el sistema de coordenadas máquina.

11.1.2.1. Movimientos en coordenadas de máquina: G53

Independientemente de cualquier offset que pudiera estar activo, un G53 en una línea de código indica al intérprete moverse a las posiciones reales de los ejes (posiciones absolutas) especificadas. Por ejemplo:

```
G53 G0 X0 Y0 Z0
```

moverá de la posición actual a la posición donde las coordenadas de máquina de los tres ejes sean cero. Puedes usar este comando si tienes una posición fija para el cambio de herramienta o si tu máquina tiene un cambiador automático de herramienta. También puedes usar este comando para despejar el área de trabajo y acceder a la pieza de trabajo en la mordaza.

G53 es un comando no modal. Debe usarse en cada bloque en el que se desee un movimiento en el sistema coordenado de la máquina.

11.1.3. Sistemas de coordenadas

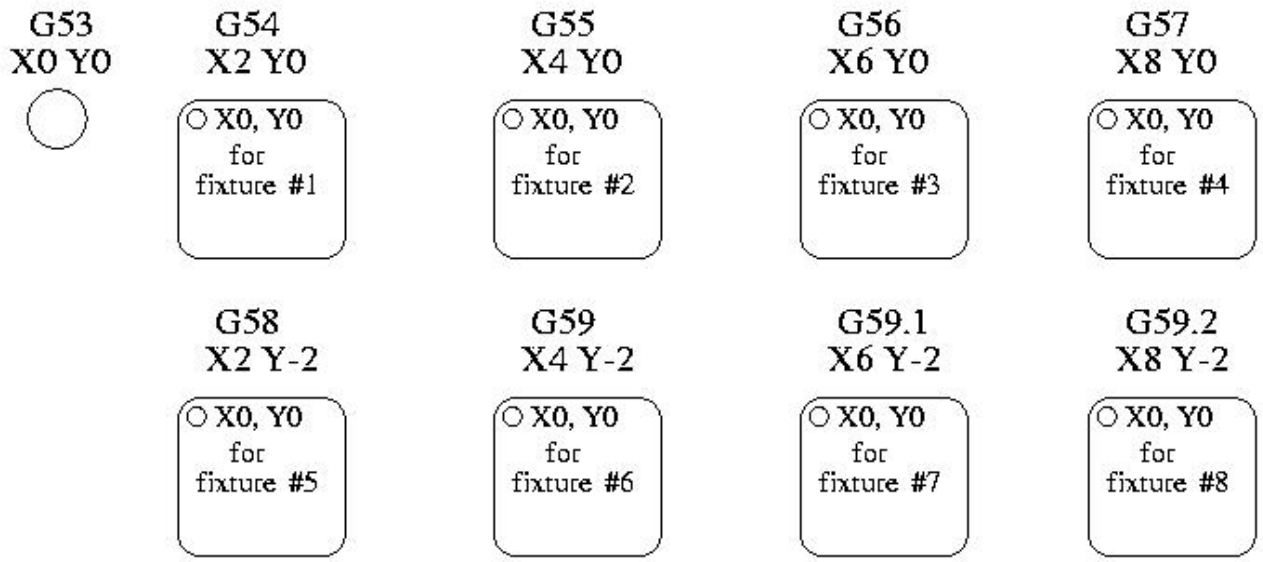


Figura 11.1: Ejemplo de sistemas de coordenadas

Offsets del sistema de coordenadas

- G54 - usa el sistema de coordenadas 1
- G55 - usa el sistema de coordenadas 2
- G56 - usa el sistema de coordenadas 3
- G57 - usa el sistema de coordenadas 4
- G58 - usa el sistema de coordenadas 5
- G59 - usa el sistema de coordenadas 6
- G59.1 - usa el sistema de coordenadas 7
- G59.2 - usa el sistema de coordenadas 8
- G59.3 - usa el sistema de coordenadas 9

Los offsets del sistema de coordenadas se utilizan para desplazar cada sistema de coordenadas desde el sistema de coordenadas de la máquina. Esto permite que el código G se programe para la pieza sin importar la ubicación de la pieza en la máquina. El uso de offsets del sistema de coordenadas te permitirá mecanizar piezas en múltiples ubicaciones con el mismo código G.

Los valores de los offsets se almacenan en el archivo VAR solicitado por el archivo INI durante el inicio de LinuxCNC. En el ejemplo de abajo, el cual usa G55, la posición de cada eje para el origen G55 se almacena en una variable numerada.

En el esquema de archivo VAR, el primer número de variable almacena el offset X, el segundo, el offset Y, y así sucesivamente para los nueve ejes. Hay conjuntos numerados como este para cada uno de los offsets de sistema de coordenadas.

Cada una de las interfaces gráficas tiene una forma de establecer valores para estos offsets. También puede establecer estos valores editando el propio archivo VAR y reinicializando luego LinuxCNC para

que lea los nuevos valores. Sin embargo, esta no es la forma recomendada. El uso de G10, G52, G92, G28.1, etc. son mejores formas de establecer las variables. Para nuestro ejemplo, editaremos directamente el archivo para que G55 tome los valores siguientes:

Cuadro 11.1: Ejemplo de parámetros G55

Eje	Variable	Valor
X	5241	2.000000
Y	5242	1.000000
Z	5243	-2.000000
A	5244	0.000000
B	5245	0.000000
C	5246	0.000000
U	5247	0.000000
V	5248	0.000000
W	5249	0.000000

Deberías leer esto como mover las posiciones cero de G55 a X = 2 unidades, Y = 1 unidad, y Z = -2 unidades desde la posición de cero absoluto.

Una vez que hay valores asignados, una llamada a G55 en un bloque de programa desplaza la referencia cero absoluta por los valores almacenados. La siguiente línea movería cada eje a la nueva posición cero. A diferencia de G53, G54 hasta G59.3 son comandos modales. Actuarán en todos los bloques de código después de que uno de ellos se ha establecido. El programa que podría ejecutarse usando offsets de fijaciones requeriría solo una coordenada de referencia para cada una de las ubicaciones y todo el trabajo se realizaria allí. El siguiente código se ofrece como un ejemplo de cómo hacer un cuadrado usando los offsets G55 que configuramos arriba.

```
G55; utilizar el sistema de coordenadas 2
G0 X0 Y0 Z0
G1 F2 Z-0.2000
X1
Y1
X0
Y0
G0 Z0
G54; utilizar el sistema de coordenadas 1
G0 X0 Y0 Z0
M2
```

En este ejemplo, el G54 cerca del final deja el sistema de coordenadas G54 con todos los offsets a cero para que haya un código modal basado en ejes absolutos de máquina. Este programa asume que lo hemos hecho y usamos el comando final como un comando para cero máquina. Hubiera sido posible usar G53 y llegar al mismo lugar pero ese comando no habría sido modal y cualquier comando emitido después hubiera vuelto a usar los offsets G55 porque ese sistema de coordenadas todavía estaría en vigor.

[source,”]

```
G54     usa parámetros del sistema de coordenadas 1
G55     usa parámetros del sistema de coordenadas 2
G56     usa parámetros del sistema de coordenadas 3
G57     usa parámetros del sistema de coordenadas 4
G58     usa parámetros del sistema de coordenadas 5
G59     usa parámetros del sistema de coordenadas 6
G59.1   usa parámetros del sistema de coordenadas 7
G59.2   usa parámetros del sistema de coordenadas 8
G59.3   usa parámetros del sistema de coordenadas 9
```

11.1.3.1. Sistema de coordenadas predeterminado

Otra variable en el archivo VAR se vuelve importante cuando pensamos en sistemas de offsets. Esta variable es la 5220. En los archivos predeterminados, su valor se establece en 1.00000. Esto significa que cuando LinuxCNC se inicia, debería usar el primer sistema de coordenadas como el predeterminado. Si se configura a 9.00000 usaría el noveno sistema de offsets como predeterminado al iniciar o reiniciar. Cualquier valor que no sea un entero (decimal realmente) entre 1 y 9, o si falta la variable 5220 hará que LinuxCNC vuelva al valor predeterminado de 1.00000 al inicio.

11.1.3.2. Configuración de offsets de sistema de coordenadas

El comando G10 L2x se puede usar para establecer los offsets de sistema de coordenadas:

- *G10 L2 P (1-9)* - Establece el(los) offset(s) a un valor. La posición actual es irrelevante (ver G10 L2 para más detalles).
- *G10 L20 P (1-9)* - Establece el(los) offset(s) de modo que la posición actual se convierte en un valor (ver G10 L20 para más detalles).

nota
Aquí solo damos un vistazo, ve a las secciones de código G para una descripción completa.

11.1.4. Offsets locales y globales

11.1.4.1. El comando G52

G52 se usa en un programa de pieza como un ”offset del sistema de coordenadas local” temporal dentro del sistema de coordenadas de la pieza de trabajo. Un ejemplo de caso de uso es cuando se mecanizan varias características idénticas en diferentes ubicaciones de una pieza. Para cada una, *G52* programa un punto de referencia local dentro de las coordenadas de pieza, y se llama a un subprograma para maquinar la característica relativa a ese punto.

Los offsets de ejes *G52* se programan relativos a las coordenadas de offset de la pieza de trabajo *G54* a *G59.3*. Como un offset local, *G52* se aplica después del offset de la pieza de trabajo, incluida la rotación. Por lo tanto, una característica parcial será mecanizada de forma idéntica en cada parte, independientemente de la orientación de la parte en el palet.



atención
En otros intérpretes de código G, *G52* no persiste después de reinicio de la máquina *M02* o *M30*, al asignar y desasignar un offset temporal dentro de un ámbito localizado de un programa de parte. En LinuxCNC, *G52* comparte parámetros con *G92*, el cual por razones históricas **persiste** esos parámetros. Ver G92 Precauciones con persistencia a continuación.



atención
G52 y *G92* comparten los mismos registros de offset. Por lo tanto, usar *G52* anulará cualquier configuración *G92* previa, y *G52* persistirá después de un reinicio de máquina cuando la persistencia de *G92* este activada. Estas interacciones pueden resultar en offsets inesperados. Ver Precauciones de interacción G92 y G52 abajo.

La programación de *G52 X1 Y2* da offsets al sistema de coordenadas actual de la pieza de trabajo, 1 para eje X y 2 para eje Y. Por consiguiente, en el DRO las coordenadas X e Y de la posición actual de la herramienta se reducirán en 1 y 2, respectivamente. Los ejes sin establecer en el comando, como Z en el ejemplo anterior, no se verán afectados: cualquier offset Z *G52* previo permanecerá vigente, y en caso de haberlo, el offset Z será cero.

El offset local temporal puede cancelarse con *G52 X0 Y0*. Cualquier eje no puesto a cero explícitamente retendrá el offset anterior.

G52 comparte los mismos registros que *G92*, por lo que *G52* es visible en el DRO y vista previa, etiquetado como *G92*.

11.1.5. G92 Offsets de ejes

G92 es el comando programable más incomprendido e inteligente en LinuxCNC. La forma en que trabaja ha cambiado un poco desde las primeras versiones a la actual y esos cambios han desconcertado a muchos usuarios. Debería ser visto como un comando que produce un offset temporal que aplica para todos los demás offsets.

11.1.5.1. Los comandos G92

G92 se usa típicamente de dos maneras conceptualmente diferentes: como un ”offset del sistema de coordenadas global” o como un ”offset del sistema de coordenadas local”.

El conjunto de comandos *G92* incluye:

- *G92* - Este comando, cuando se usa con nombres de eje, establece valores para las variables de offset.
- *G92.1* - Este comando establece valores cero para las variables G92.
- *G92.2* - Este comando suspende G92, pero no pone a cero las variables.
- *G92.3* - Este comando aplica los valores de offset que se suspendieron.

Como offset global, *G92* se usa para cambiar todas los sistemas de coordenadas de la pieza de trabajo, *G54* a *G59.3*. Un ejemplo de caso de uso es cuando se mecanizan varias piezas idénticas en fijaciones con ubicaciones conocidas en un palet, pero la ubicación del palet puede cambiar entre lotes o entre máquinas. Cada offset de ubicación de la fijación, relativo a un punto de referencia en el palet, está preestablecido en uno de los sistemas de coordenadas de pieza, de *G54* a *G59.3*, y se usa *G92* para hacer ”touch off” en el punto de referencia en el palet. Luego, para cada parte, se selecciona el sistema de coordenadas de la pieza de trabajo correspondiente y se ejecuta el programa de pieza.

---

nota
La rotación del sistema de coordenadas de la pieza *G10 R-* es específica del intérprete *rs274ngc*, y el offset *G92* se aplica *después* de la rotación. Cuando se usa *G92* como offset global, las rotaciones del sistema de coordenadas de pieza pueden tener resultados inesperados.

Como sistema de coordenadas local, *G92* se usa como offset temporal dentro del sistema de coordenadas de la pieza de trabajo. Un ejemplo de caso de uso es al mecanizar una pieza con varias características idénticas en diferentes ubicaciones. Para cada característica, se usa *G92* para establecer un punto de referencia local, y se llama a un subprograma para mecanizar la característica a partir de ese punto.

nota
Se desaconseja el uso de *G92* para programar con sistemas de coordenadas locales en un programa de pieza. En su lugar, vea G52, un offset local del sistema de coordenadas es más intuitivo cuando se conoce el offset deseado relativo a la pieza de trabajo, pero es posible que no se conozca la ubicación actual de la herramienta.

La programación *G92 X0 Y0 Z0* establece la ubicación actual de la herramienta en las coordenadas X0, Y0 y Z0, sin movimiento. G92 **no** funciona desde coordenadas absolutas de la máquina. Funciona desde **ubicación actual**.

G92 también funciona desde la ubicación actual modificada por cualquier otro offset que esté vigente cuando se invoca *G92*. Al probar las diferencias entre los offsets de trabajo y los reales se encontró que un offset *G54* podría cancelar un *G92* y, por lo tanto, parecía que no había offsets en vigor. Sin embargo, *G92* estaba todavía vigente para todas las coordenadas y produjo los offsets de trabajo esperados para los otros sistemas de coordenadas.

Predeterminadamente, los offsets *G92* se restauran después de que se inicia la máquina. Los programadores que deseen un comportamiento tipo Fanuc, donde los offsets *G92* se borran al inicio de la máquina y después de un reinicio o fin de programa, puede deshabilitar la persistencia *G92* configurando *DISABLE_G92_PERSISTENCE = 1* en el sección *[RS274NGC]* del archivo INI.

nota
Es una buena práctica eliminar los offsets *G92* al final de su uso con *G92.1* o *G92.2*. Al iniciar LinuxCNC con persistencia *G92* habilitada (el valor predeterminado), se aplicará cualquier offset en las variables *G92* cuando un eje se lleve en casa. Ver G92 Precauciones con persistencia a continuación.

11.1.5.2. Configuración de valores G92

Hay por lo menos dos maneras de asignar valores G92:

- Con un click derecho en la posición mostrada en tklinuxcnc, se abre una ventana donde es posible introducir un valor.
- Con el comando G92

Ambas funcionan desde la posición actual del eje que debería moverse.

La programación *G92 X Y Z A B C U V W* establece los valores de las variables G92 para que cada eje tome el valor asociado con su nombre. Esos valores se asignan a la posición actual de los ejes. Estos resultados satisfacen los párrafos uno y dos del documento de NIST.

Los comandos G92 funcionan desde la ubicación actual del eje y suman y restan correctamente para dar a la posición actual del eje el valor asignado por el comando G92. Los efectos funcionan a pesar de que haya offsets anteriores.

Por tanto, si el eje X muestra actualmente 2.0000 como su posición, un *G92 X0* establecerá un offset de -2.0000 para que la ubicación actual de X sea cero. Un *G92 X2* establecerá un offset de 0.0000 y la posición mostrada no cambiará. Un *G92 X5.0000* establecerá un offset de 3.0000 para que la posición actual mostrada sea 5.0000.

11.1.5.3. Precauciones de persistencia G92

De forma predeterminada, los valores de un offset *G92* se guardarán en el archivo VAR y se restaurarán después de un inicio o reinicio de la máquina.

Los parámetros G92 son:

- 5210 - Activar/desactivar bandera (1.0 / 0.0)
- 5211 - Offset eje X
- 5212 - Offset eje Y
- 5213 - Offset eje Z
- 5214 - Offset eje A
- 5215 - Offset eje B
- 5216 - Offset eje C
- 5217 - Offset eje U
- 5218 - Offset eje V
- 5219 - Offset eje W

donde 5210 es la bandera de habilitación *G92* (1 para habilitado, 0 para deshabilitado) y 5211 a 5219 son los offsets de eje. Si se ven posiciones inesperadas como resultado de un movimiento ordenado, resultado de almacenar un offset en un programa anterior y no borrarlos al final, entonces hay que emitir un G92.1 en la ventana MDI para borrar los offsets almacenados.

Si existen valores G92 en el archivo VAR cuando se inicia LinuxCNC, los valores en el archivo var se aplicarán a los valores de la ubicación actual de cada eje. Si ésta es la posición casa y está establecida como cero máquina, todo estará correcto. Una vez que se ha sido establecido casa usando interruptores de máquina reales, o moviendo cada eje a una posición inicial conocida y emitiendo un comando casa de eje, cualquier offset G92 será aplicado. Si se tiene un G92 X1 vigente al llevar a casa el eje X, el DRO leerá *X: 1.000* en lugar del esperado *X: 0.000* porque el G92 se aplicó al origen de máquina. Si se emite un G92.1 y el DRO ahora lee solo ceros, entonces hubo un desplazamiento G92 vigente la última vez que corrió LinuxCNC.

A menos que la intención sea usar los mismos offsets G92 en el programa siguiente, la mejor práctica es emitir un G92.1 al final de cualquier archivo de código G donde se utilice offsets G92.

Cuando se aborta, durante el procesamiento, un programa que tiene offsets *G92* vigentes, un inicio hará que se activen nuevamente. Como salvaguarda, tenga siempre su preámbulo estableciendo el entorno como usted lo espera. Además, la persistencia *G92* puede deshabilitarse configurando *DISABLE_G92_PERSISTENCE = 1* en la sección *[RS274NGC]* del archivo INI.

11.1.5.4. Precauciones de interacción G92 y G52

G52 y *G92* comparten los mismos registros de offset. A menos que la persistencia *G92* está deshabilitada en el archivo INI (ver Comandos G92), los offsets *G52* también persistirán después del reinicio de la máquina, *M02* o *M30*. Hay que tener cuidado que un offset *G52* vigente durante un aborto de programa, puede dar lugar a offsets no deseados cuando se ejecute el programa siguiente. Ver G92 Precauciones con persistencia más arriba.

11.1.6. Programas de muestra usando offsets

11.1.6.1. Programa de muestra utilizando offsets de coordenadas de pieza de trabajo

Esta muestra de proyecto de grabado, fresa un grupo de cuatro círculos con radio .1 en una burda disposición de estrella alrededor de un círculo central. Podemos configurar el patrón de círculo individual como este.

```
G10 L2 P1 X0 Y0 Z0 (asegúrese de que G54 esté configurado en la máquina cero)
G0 X-0.1 Y0 Z0
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0
M2
```

Podemos emitir un conjunto de comandos para crear offsets para los otros cuatro círculos, como esto.

```
G10 L2 P2 X0.5 (compensa el valor de G55 X en 0,5 pulgadas)
G10 L2 P3 X-0.5 (compensa el valor de G56 X en -0.5 pulgadas)
G10 L2 P4 Y0.5 (compensa el valor G57 Y en 0.5 pulgadas)
G10 L2 P5 Y-0.5 (compensa el valor G58 Y en -0.5 pulgadas)
```

Los ponemos juntos en el programa siguiente:

```
(un programa para fresar cinco círculos pequeños en forma de diamante)

G10 L2 P1 X0 Y0 Z0 (asegúrese de que G54 sea la máquina cero)
G10 L2 P2 X0.5 (compensa el valor de G55 X en 0,5 pulgadas)
G10 L2 P3 X-0.5 (compensa el valor de G56 X en -0.5 pulgadas)
G10 L2 P4 Y0.5 (compensa el valor G57 Y en 0.5 pulgadas)
G10 L2 P5 Y-0.5 (compensa el valor G58 Y en -0.5 pulgadas)

G54 G0 X-0.1 Y0 Z0 (círculo central)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0

G55 G0 X-0.1 Y0 Z0 (offset primer círculo)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0

G56 G0 X-0.1 Y0 Z0 (offset segundo círculo)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0

G57 G0 X-0.1 Y0 Z0 (offset tercer círculo)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0

G58 G0 X-0.1 Y0 Z0 (offset cuarto círculo)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G54 G0 X0 Y0 Z0

M2
```

Ahora llega el momento en que podríamos aplicar un conjunto de offsets G92 a este programa. Verás que se está ejecutando en cada caso en Z0. Si la fresa estaba en la posición cero, un G92 Z1.0000 emitido al inicio del programa desplazaría todo por una pulgada. También podrías cambiar todo el patrón en el plano XY agregando algunos offsets X e Y con G92. Si haces esto, debes agregar un comando G92.1 justo antes del M2 que finaliza el programa. Si no lo haces, otros programas que pudieras ejecutar después de éste también usarán ese offset G92. Además, guardarán los valores de G92 cuando cierres LinuxCNC y serán recargados cuando lo inicies de nuevo.

11.1.6.2. Programa de muestra usando offsets G52

(Por escribirse)

11.2. Compensación de herramienta

11.2.1. Touch off

Se puede actualizar la tabla de herramientas de forma automática usando la pantalla Touch off en la interfaz AXIS.

Los pasos típicos para actualizar la tabla de herramientas son:

- Después de homing, cargar una herramienta con *T*n *M6*; donde *n* es el número de herramienta.
- Mover la herramienta a un punto establecido utilizando un medidor o hacer un corte de prueba y medir.
- Hacer clic en el botón *Touch Off* en la pestaña Control manual (o presione el botón *End* en su teclado).
- Seleccione *Tabla de herramientas* en el cuadro desplegable Sistema de Coordenadas.
- Ingrese el valor calibrado o la dimensión medida y seleccione Aceptar.

La tabla de herramientas se cambiará con la longitud Z correcta para que el DRO muestre la posición Z correcta y se emitirá un comando G43 para que la nueva longitud Z de la herramienta esté vigente. Touch off de la Tabla de herramientas solo está disponible cuando una herramienta se carga con *T*n*M6*.

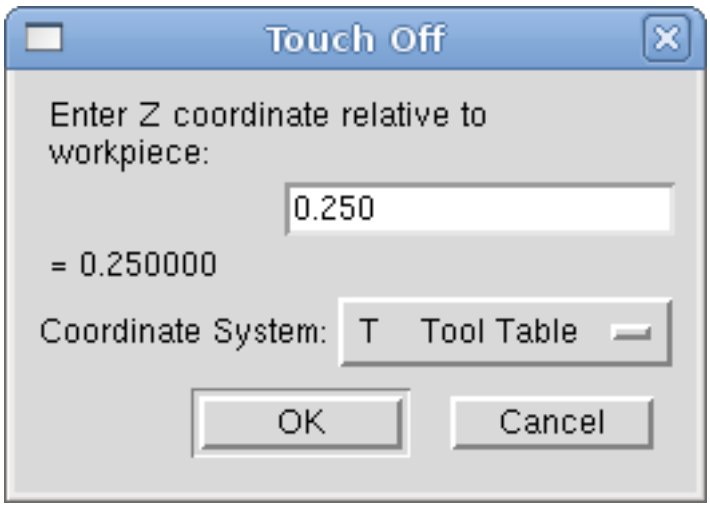


Figura 11.2: Touch off de tabla de herramientas

11.2.1.1. Usando G10 L1/L10/L11

Los comandos G10 L1/L10/L11 se pueden usar para establecer las compensaciones de la tabla de herramientas:

- `G10 L1 P__n__` - Establece los offsets en un valor. Posición actual irrelevante (ver G10 L1 para más detalles).

- `G10 L10 P__n__` - Establece el(los) offset(s) de modo que la posición actual c/amarre 1-8 se convierta en un valor (ver G10 L10 para más detalles).
- `G10 L11 P__n__` - Establece el(los) offset(s) de modo que la posición actual c/amarre 9 se convierta en un valor (ver G10 L11 para más detalles).

---

nota
Esta es solo una presentación breve, la guía de referencia de código G tiene explicaciones más detalladas.

11.2.2. Tabla de herramientas

La *Tabla de herramientas* es un archivo de texto que contiene información sobre cada herramienta. El archivo está ubicado en el mismo directorio de tu configuración y se llama predeterminadamente *tool.tbl*. Se puede especificar un nombre de archivo con la configuración [EMCIO]TOOL_TABLE del archivo INI. Las herramientas pueden estar en un cambiador de herramientas o simplemente ser cambiadas manualmente. El archivo puede editarse con un editor de texto o ser actualizado con G10 L1. Ver la sección Tabla de herramientas de torno para un ejemplo del formato de tabla de herramientas de torno. El número máximo de ranuras es 1000.

Se puede usar el Editor de herramientas o un editor de texto para editar la tabla de herramientas. Si utilizas un editor de texto, asegúrate de volver a cargar la tabla en la GUI.

11.2.2.1. Formato de tabla de herramientas

.Formato de tabla de herramientas

T#	**P#**	**X**	**Y**	**Z**	**A**	**B**	**C**	**U**	**V**	**W**	**Dia.**	**A. front.**	**A. tras.**	**Ori.**	**Coment.**
;	(sin datos después de un punto y coma)														
T1	P17	X0	Y0	Z0	A0	B0	C0	U0	V0	W0	D0	I0	J0	Q0	;rem
T2	P5	X0	Y0	Z0	A0	B0	C0	U0	V0	W0	D0	I0	J0	Q0	;rem
T3	P12	X0	Y0	Z0	A0	B0	C0	U0	V0	W0	D0	I0	J0	Q0	;rem

En general, el formato de línea de la tabla de herramientas es:

- T - número de herramienta (los números de herramienta deben ser únicos)
- P - número de ranura, 1-1000 (los números de ranura deben ser únicos, ranura 0 representa al husillo)
- X..W - offset de herramienta en el eje especificado - punto flotante
- D - diámetro de la herramienta - coma flotante, valor absoluto
- I - ángulo frontal (solo torno) - punto flotante
- J - ángulo posterior (solo torno) - punto flotante
- Q - orientación de la herramienta (solo torno) - entero, 0-9
- ; - comienzo del comentario o anotacion - texto

Los números de herramienta deben ser únicos. Las líneas que comienzan con punto y coma son ignoradas.

Las unidades utilizadas para la longitud, diámetro, etc., están en unidades de máquina.

Probablemente querrás mantener las entradas de la herramienta en orden ascendente, especialmente si vas a usar un cambiador de herramientas aleatorio, aunque la tabla de herramientas permite números de herramienta en cualquier orden.

Cada línea puede tener hasta 16 entradas, pero es probable que tengan mucho menos. Las entradas para T (número de herramienta) y P (número de ranura) son obligatorias. La última entrada (un comentario o anotación, precedido por un punto y coma) es opcional. La lectura se facilita si las entradas están organizadas en columnas, como se muestra en la tabla, pero el único requisito de formato es que haya al menos un espacio o tabulador después de cada una de las entradas en una línea y un carácter de nueva línea al final de cada entrada.

Los significados de las entradas y el tipo de datos que se colocarán en cada uno son como sigue.

Número de herramienta (requerido)
: La columna *T* contiene el número (entero sin signo) que representa un número de código para la herramienta. El usuario puede usar cualquier código para cualquier herramienta, siempre que los códigos sean enteros sin signo.

Número de ranura (requerido)
: La columna *P* contiene el número (entero sin signo) que representa el número de ranura del cambiador de herramientas donde se puede encontrar la herramienta. Las entradas en esta columna deben ser todas diferentes.

Los números de ranura generalmente comenzarán en 1 y subirán hasta la ranura más alta disponible en el cambiador de herramientas. Pero no todos los cambiadores de herramientas siguen este patrón. Tus números de ranura serán determinados por los números de tu cambiador de herramientas utiliza para referirse a las ranuras. Por tanto, los números de ranura que uses estarán determinados por el esquema de numeración utilizado en tu cambiador de herramientas, y los números de ranura que uses deben tener sentido en tu máquina.

Números de offset de datos (opcional)
: Las columnas *Datos de Offset* (XYZABCUVW) contienen números reales que representar los offsets de herramientas en cada eje. Este número se usará si la herramienta utiliza offset de longitud y está seleccionada. Estos números pueden ser positivos, cero o negativos, y de hecho son completamente opcionales. Aunque probablemente querrás hacer al menos una entrada aquí, de lo contrario no tendría mucho sentido hacer un entrada en la tabla de herramientas para empezar.

En una fresadora típica, probablemente se querrá una entrada para Z (offset de longitud de la herramienta). En un torno típico, probablemente se querrá una entrada para X (offset de herramienta X) y Z (offset de herramienta Z). En una fresadora típica usando compensación del diámetro del cortador (comp. del cortador), probablemente también se querrá agregar una entrada para D (diámetro del cortador). En un torno típico usando compensación del diámetro de la punta de la herramienta (comp. herramienta), probablemente también se querrá agregar una entrada para D (diámetro de la punta de la herramienta).

Un torno también requiere información adicional para describir la forma y orientación de la herramienta. Probablemente se querrá tener entradas para I (ángulo frontal) y J (ángulo posterior) de la herramienta. Probablemente también se querrá una entrada para Q (orientación de la herramienta).

Ver el capítulo Información para el usuario de torno para más detalles.

La columna *Diámetro* contiene un número real. Este número solo se usa si la compensación del cortador está activada con esta herramienta. Si la ruta programada durante la compensación es el borde del material que se está cortando, este debe ser un número real positivo que represente la medida

del diámetro de la herramienta. Si la ruta programada durante la compensación es la trayectoria de una herramienta cuyo diámetro es nominal, este debe ser un número pequeño (positivo o negativo, pero cercano a cero) que represente solo la diferencia entre el diámetro medido de la herramienta y el diámetro nominal. Si la compensación del cortador no se usa con una herramienta, no importa qué número hay en esta columna.

La columna *Comentario* puede usarse opcionalmente para describir la herramienta. Cualquier tipo de descripción estará bien. Esta columna es solo para beneficio de lectores humanos. El comentario debe ir precedido de un punto y coma.

---

nota
Las versiones anteriores de LinuxCNC tenían dos formatos de tabla de herramientas diferentes para fresadoras y tornos, pero desde la versión 2.4.x, se utiliza el mismo formato de tabla de herramientas para todas las máquinas.

11.2.2.2. E/S de herramientas

El programa en tiempo no-real **iocontrol** se usa convencionalmente para administrar el cambiador de herramienta (y otras funciones de E/S para habilitar LinuxCNC y el hardware de control de enfriamiento). Los pines HAL usados para administración de herramienta tienen el prefijo **iocontrol.0.**.

Un comando **T** en código G evalúa el pin HAL de salida `iocontrol.0.tool-prepare`. El pin de entrada HAL `iocontrol.0.tool-prepared` debe estar asignado por una lógica HAL externa para completar la preparación de la herramienta, llevando a un subsecuente reinicio del pin tool-prepare.

Un comando **M6** en código G evalúa el pin HAL de salida `iocontrol.0.tool-change`. El pin HAL de entrada relacionado `iocontrol.0.tool-prepared` debe ser asignado por una lógica HAL externa para indicar la terminación del cambio de herramienta, llevando a un subsecuente reinicio del pin tool-change.

Los datos de herramienta se accesan por un índice ordenado (idx) que depende del tipo de cambiador de herramienta especificado en `[EMCIO]RANDOM_TOOLCHANGER=`*tipo*.

1. Para `RANDOM_TOOLCHANGER = 0`, (el predeterminado es 0 y especifica un cambiador de herramienta no aleatorio) idx es un número que indica la secuencia en la que se cargaron -los datos de- las herramientas.
2. Para `RANDOM_TOOLCHANGER = 1`, idx es el número de ranura **actual** para el número de herramienta especificado por el comando de código G de selección de herramienta **T*n***.

El programa io proporciona pines HAL de salida para facilitar la gestión de cambio de herramienta:

1. **iocontrol.0.tool-prep-number**
2. **iocontrol.0.tool-prep-index**
3. **iocontrol.0.tool-prep-pocket**
4. **iocontrol.0.tool-from-pocket**

1. Número de herramienta *n*==0 indica sin herramienta.
2. El número de ranura de una herramienta se asigna cuando se cargan/recargan los datos de herramienta desde su fuente de datos ([EMCIO]TOOL_TABLE o [EMCIO]DB_PROGRAM).
3. En el comando de código G **T*n*** (*n* != 0):
    a. **iocontrol.0.tool-prep-index** = *idx* (índice basado en la secuencia de carga de datos de la herramienta)

   b. **iocontrol.0.tool-prep-number** = *n*
   c. **iocontrol.0.tool-prep-pocket** = el número de ranura para *n*

4. En el comando de código G **T0** (*n* == 0 quitar):

   a. **iocontrol.0.tool-prep-index** = 0
   b. **iocontrol.0.tool-prep-number** = 0
   c. **iocontrol.0.tool-prep-pocket** = 0

5. En el código M **M6** (siguiendo el pin iocontrol.0.tool-changed 0-->1):

   a. **iocontrol.0.tool-prep-pocket** = número de ranura usada para tomar la herramienta

1. Número de herramienta *n*==0 es **no especial**.
2. Número de ranura 0 es **especial** como lo indica el **husillo**.
3. El número de ranura **actual** para la herramienta *n* es el índice de datos de herramienta (*idx*) para la herramienta *n*.
4. En el comando de código G **T*n***:

   a. **iocontrol.0.tool-prep-index** = índice de datos de herramienta (idx) para la herramienta *n*
   b. **iocontrol.0.tool-prep-number** = *n*
   c. **iocontrol.0.tool-prep-pocket** = número de ranura para la herramienta *n*

5. En el código M **M6** (siguiendo el pin iocontrol.0.tool-changed 0-->1):

   a. **iocontrol.0.tool-prep-pocket** = número de ranura usada para tomar la herramienta

---

nota
En el arranque **iocontrol.0.tool-from-pocket** = 0. Un comando M61Q*n* (*n*!=0) no cambia **iocontrol.0.tool-from-pocket**. Un comando M61Q0 (*n*==0) asigna **iocontrol.0.tool-from-pocket** a 0.

11.2.2.3. Cambiadores de herramienta

LinuxCNC admite tres tipos de cambiadores de herramientas: *manual*, *ubicación aleatoria* y *ubicación no aleatorio o fija*. Información sobre la configuración de un cambiador de herramienta LinuxCNC está en la Sección EMCIO del capítulo INI.

Cambiador de herramientas manual El cambiador manual de herramientas (cambiar la herramienta a mano) se trata como un cambiador de herramienta de ubicación fija. Los cambios manuales de herramienta pueden ser ayudados por una configuración HAL que emplee el programa en tiempo no-real **hal_manualtoolchange** y se especifica típicamente en un archivo INI con sentencias INI:

```
[HAL]
HALFILE = axis_manualtoolchange.hal
```

Cambiadores de herramientas de ubicación fija Los cambiadores de herramientas de ubicación fija siempre devuelven las herramientas a una posición fija en el cambiador de herramientas. Esto también incluiría a diseños como las torretas de torno. Cuando LinuxCNC está configurado para un cambiador de herramientas de ubicación fija, el número *P* no se usa internamente en LinuxCNC (pero se lee, conserva y reescribe), por lo que se puede usar P para cualquier número que se quiera.

nota
Cuando se usa `[EMCIO]RANDOM_TOOLCHANGER = 0` (el predeterminado), el número de ranura *P* es un parámetro de los datos de herramienta que se obtiene de la fuente de datos de herramienta (`[EMCIO]TOOL_TABLE` o `[EMCIO]DB_PROGRAM`). En muchas aplicaciones es fijo, pero puede cambiarse editando `[EMCIO]TOOL_TABLE` o programáticamente cuando se usa `[EMCIO]DB_PROGRAM`. LinuxCNC manda actualizaciones a la fuente de datos (`[EMCIO]TOOL_TABLE` o `[EMCIO]DB_PROGRAM`) con códigos G10L1, G10L10, G10L11, M61. LinuxCNC puede jalar actualizaciones a los datos de herramienta desde la fuente de datos por medio de comandos de interfaz de usuario (UI) (ejemplo de Python: `linuxcnc.command().load_tool_table()`) o con el código G: `G10L0`.

Cambiadores de herramientas de ubicación aleatoria Los cambiadores de herramientas de ubicación aleatoria (`[EMCIO]RANDOM_TOOLCHANGER = 1`) intercambian la herramienta en el husillo con una del cambiador. Con este tipo de cambiador de herramientas, la herramienta siempre esta en un ranura diferente después de un cambio de herramienta. Cuando una herramienta se cambia, LinuxCNC reescribe el número de ranura para realizar un seguimiento de dónde están las herramientas. T puede ser cualquier número pero P debe ser un número que tenga sentido para la máquina.

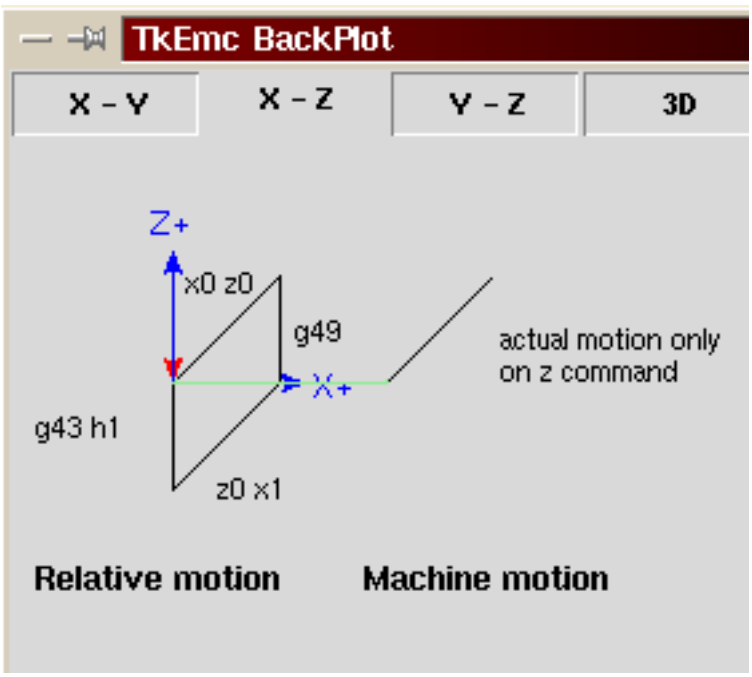
11.2.3. Compensación de longitud de herramienta

A las compensaciones de longitud de herramienta se les dan números positivos en la tabla de herramientas. Una compensación de herramienta es programada usando G43 H_n_, donde *n* es el número de índice de la herramienta deseada en la tabla de herramientas. Es intencionado que todas las entradas en la tabla de herramientas sean positivas. El valor de H es verificado, debe ser un entero no negativo cuando se lee. El intérprete se comporta como a continuación:

1. Si se programa G43 H*n*, se hace una llamada a la función `USE_TOOL_LENGTH_OFFSET( __longitud__)` (donde *longitud* es la diferencia de longitud leída de la tabla de herramientas para el índice de la herramienta *n*), se reposiciona tool_length_offset en el modelo de configuraciones de la máquina y el se ajusta del valor de current_z en el modelo. Nótese que *n* no tiene que ser el mismo que el número de ranura de la herramienta que está actualmente en el husillo.
2. Si se programa G49, se llama a `USE_TOOL_LENGTH_OFFSET(0.0)`, se restablece tool_length_offset a 0.0 en la plantilla de configuraciones de la máquina y se ajusta el valor actual de current_z en el modelo. El efecto de la compensación de longitud de herramienta se ilustra en la captura siguiente. Nótese que se resta la longitud de herramienta a Z para que el punto de control programado corresponda con la punta de la herramienta. Nótese también que el efecto de compensación de longitud de herramienta es inmediato cuando la posición de Z es vista como una coordenada relativa, pero no tiene efecto en la posición de máquina real hasta que se programe un movimiento Z.

Programa de prueba de longitud de herramienta. La herramienta #1 tiene una pulgada de largo.

```
N01 G1 F15 X0 Y0 Z0
N02 G43 H1 Z0 X1
N03 G49 X0 Z0
N04 G0 X2
N05 G1 G43 H1 G4 P10 Z0 X3
N06 G49 X2 Z0
N07 G0 X0
```



Con este programa, en la mayoría de los casos, la máquina aplicará el offset en forma rampa durante del movimiento en xyz siguiendo la palabra G43.

11.2.4. Compensación de radio de cortador

La compensación de cortador permite al programador programar la trayectoria de la herramienta sin conocer el diámetro exacto de la misma. La única advertencia es que el programador debe programar el movimiento de entrada para que sea al menos tan largo como el radio de herramienta más grande que podría usarse.

Hay dos caminos posibles que el cortador puede tomar según la compensación esté en el lado izquierdo o derecho de una línea en la dirección del movimiento vista desde detrás del cortador. Para visualizar esto, imagina que estás parado sobre la pieza caminando detrás de la herramienta mientras progresa su corte. G41 es tu lado izquierdo de la línea y G42 es el lado derecho de la línea.

El punto final de cada movimiento depende del siguiente movimiento. Si el siguiente movimiento crea una esquina exterior, el movimiento será hasta el punto final de la línea de corte compensada. Si el siguiente movimiento crea una esquina interior el movimiento se detendrá brevemente para no dañar la pieza. La figura siguiente muestra cómo el movimiento compensado se detendrá en diferentes puntos dependiendo del movimiento siguiente.

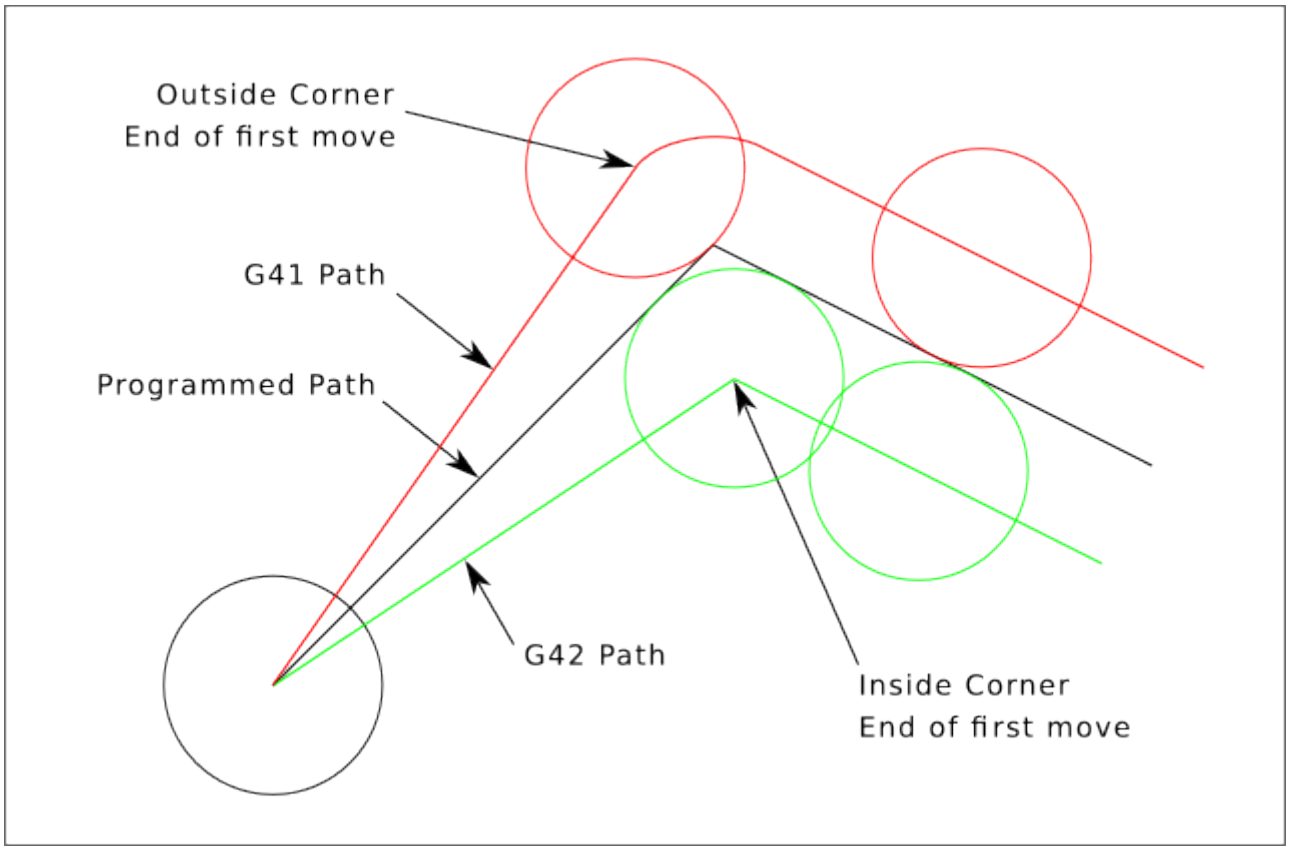


Figura 11.3: Punto final de compensación

11.2.4.1. Descripción general

La compensación del cortador utiliza los datos de la tabla de herramientas para determinar el offset necesario. Los datos se pueden configurar en tiempo de ejecución con G10 L1.

Cualquier movimiento que sea lo suficientemente largo como para realizar la compensación funcionará como movimiento de entrada. La longitud mínima es el radio de corte. Esto puede ser un movimiento rápido sobre la pieza de trabajo. Si se emiten varios movimientos rápidos después de un G41/42 solo el último moverá la herramienta a la posición compensada.

En la figura siguiente se puede ver que el movimiento de entrada se compensa a la derecha de la línea. Esto coloca el centro de la herramienta a la derecha de X0 en este caso. Si tuviera que programar un perfil y el final está en X0, el perfil resultante dejaría una protuberancia debido al offset del movimiento de entrada.

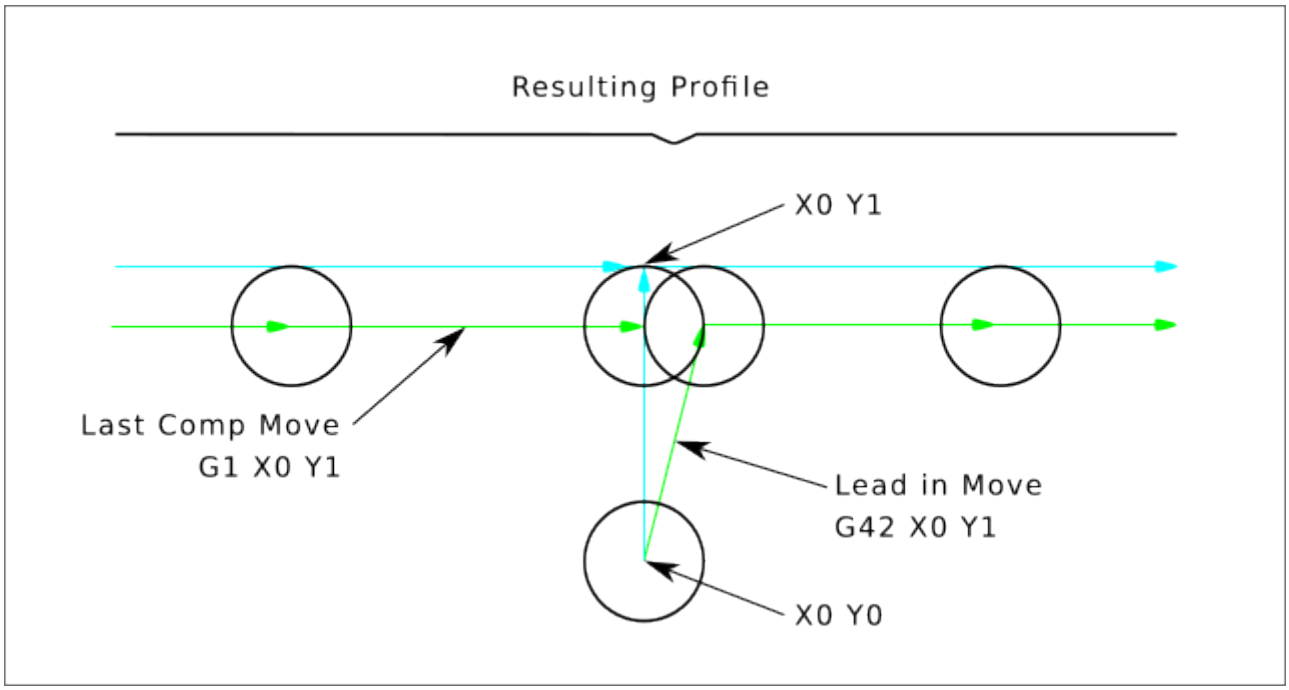


Figura 11.4: Movimiento de entrada

El movimiento del eje Z puede tener lugar mientras se sigue el contorno en el plano XY. Se pueden omitir partes del contorno retrayendo el eje Z sobre la pieza y extendiendo el eje Z en el siguiente punto de inicio.

Se pueden programar movimientos rápidos mientras la compensación está activada.

Inicie un programa con G40 para asegurar que la compensación esté desactivada.

11.2.4.2. Ejemplos

```
G-Code
F25 ( Set Feed Rate )
G40 ( Cancel Comp )
G10 L1 P1 R0.25 Z1 ( Set Tool Table )
T1 M6 ( Load Tool )
G42 ( Start Comp Right )
G1 X1 Y1 (Lead In Move)
X5 ( Cut Path )
Y5
X1
Y1
G40 ( Cancel Comp )
G0 X0 Y0 ( Exit Move )
M2 ( End Program )
```

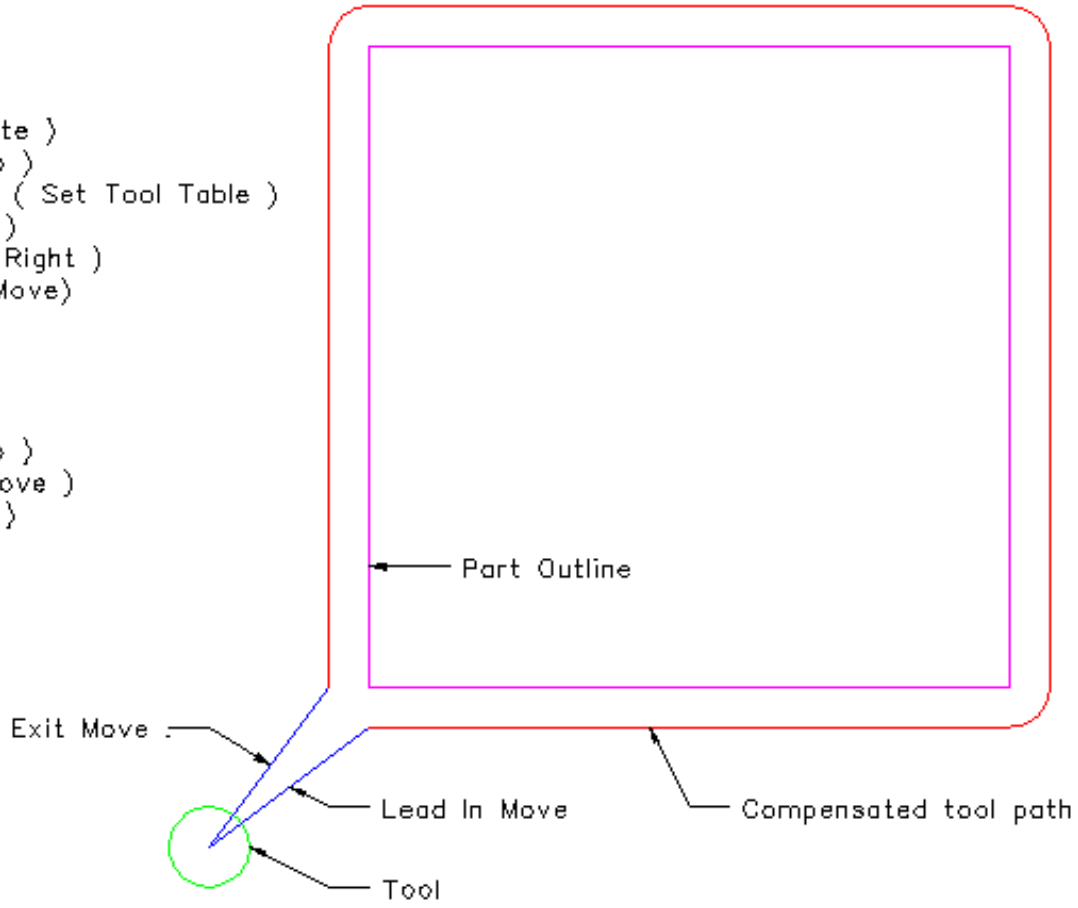


Figura 11.5: Perfil externo

```
G20 ( Inch Mode )
F30 ( Set Feed Rate )
G10 L1 P1 R.25 Z1 ( Set Tool Table )
T1 M6 ( Load the Tool )
G0 Z0 ( Move to safe Z height )
G41 ( Start Cutter Comp Left )
X4 Y3 ( Rapid to start point )
G1 X5 Z-1 ( Move to cut height )
G3 X6 Y4 J1 ( Arc into cut path )
G1 Y6 ( Cut Profile )
X2
Y2
X6
Y4
G3 X5 Y5 I-1 ( Arc out of cut path )
G0 Z0 ( Move cutter to safe Z height )
G40 ( Stop Cutter Comp )
G0 X1 Y1 ( Move to safe position )
T0 M6 ( Remove Tool )
M2 ( End Program )
```

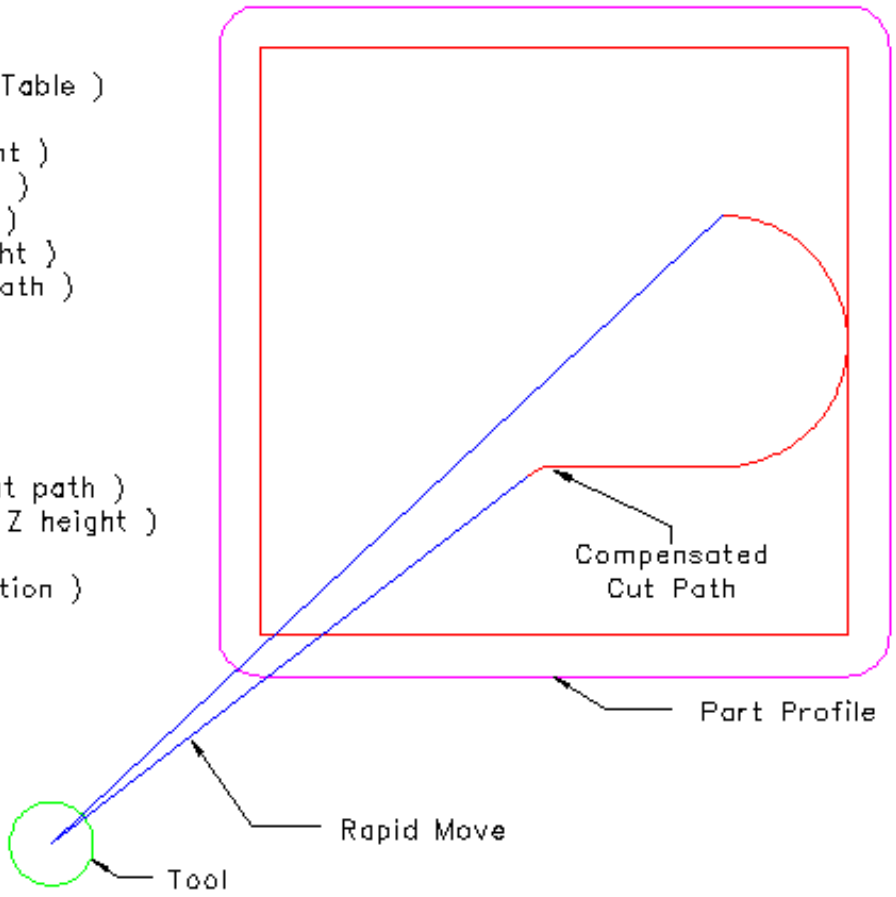


Figura 11.6: Perfil interno

11.3. Tool Edit GUI

11.3.1. Descripción general

nota
The tooledit elements described here are available since version 2.5.1 and later. In version 2.5.0, the graphical interface interface does not allow these adjustments.

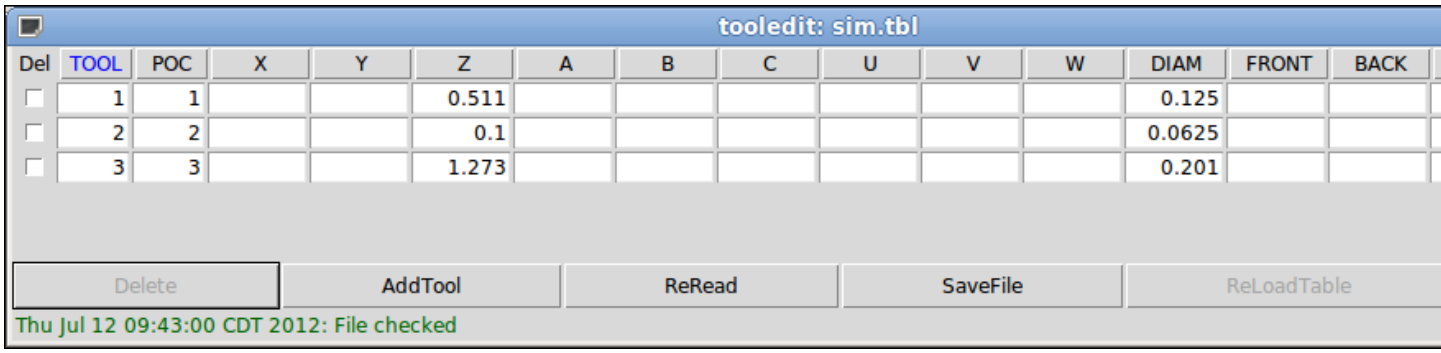


Figura 11.7: Tool Edit GUI - Overview

The *tooledit* program can update the tool table file with edited changes by using the SaveFile button. The SaveFile button updates the system file but a separate action is required to update the tool table data used by a running LinuxCNC instance. With the AXIS GUI, both the file and the current tool table data used by LinuxCNC can be updated with the ReloadTable button. This button is enabled only when the machine is ON and IDLE.

11.3.2. Column Sorting

The tool table display can be sorted on any column in ascending order by clicking on the column header. A second click sorts in descending order. Column sorting requires that the machine is configured with the default Tcl version >= 8.5.

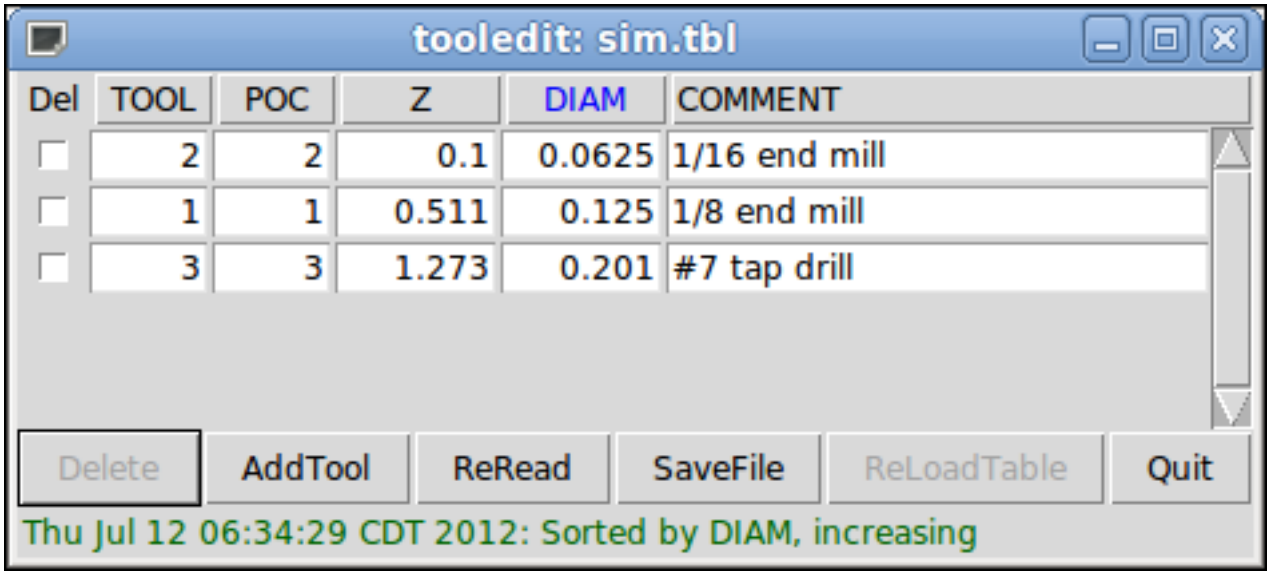


Figura 11.8: Tool Edit GUI - Column Sorting

In Ubuntu Lucid 10.04 Tcl/Tk8.4 it is installed by default. The installation is performed as follows:

```
sudo apt-get install tcl8.5 tk8.5
```

Depending upon other applications installed on the system, it may be necessary to enable Tcl/Tk8.5 with the commands:

```
sudo update-alternatives --config tclsh  ;# select the option for tclsh8.5
sudo update-alternatives --config wish   ;# select the option for wish8.5
```

11.3.3. Columns Selection

By default, the *tooledit* program will display all possible tool table parameter columns. Since few machines use all parameters, the columns displayed can be limited with the following INI file setting:

Syntax of INI file

```
[DISPLAY]
TOOL_EDITOR = tooledit column_name column_name ...
```

Example for Z and DIAM columns

```
[DISPLAY]
TOOL_EDITOR = tooledit Z DIAM
```

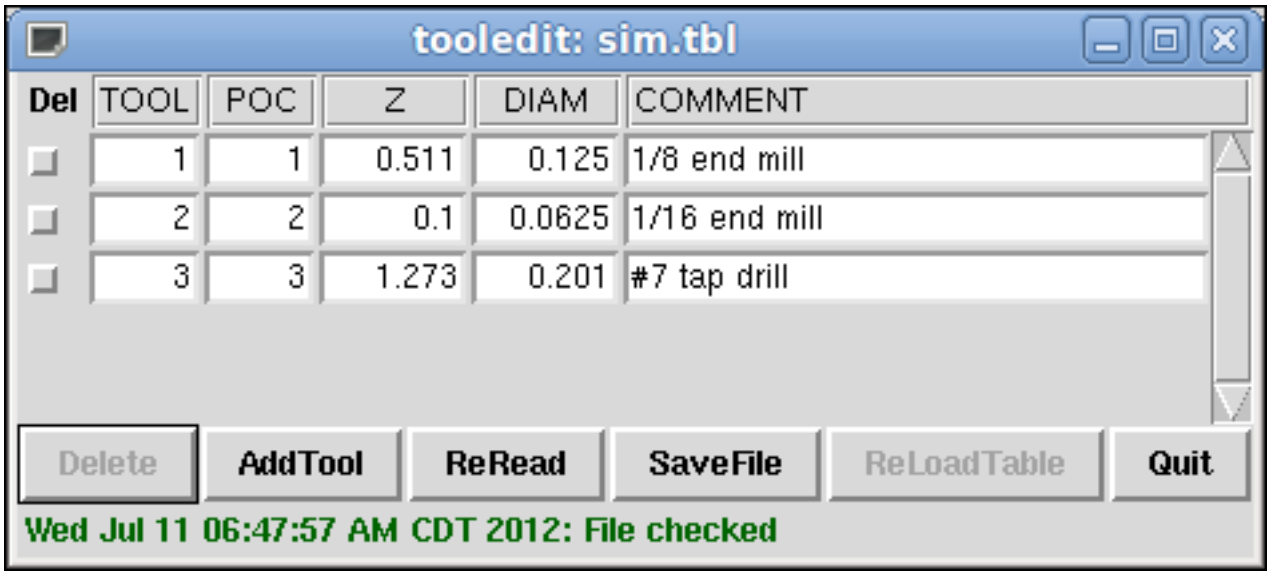


Figura 11.9: Tool Edit GUI - Columns Selection Example

11.3.4. Stand Alone Use

El programa *tooledit* también se puede invocar como un programa independiente. Por ejemplo, si el programa está en la ruta de acceso del usuario, escribiendo *tooledit* se mostrará la sintaxis de uso:

Stand Alone

```
tooledit
Usage:
       tooledit filename
       tooledit [column_1 ... column_n] filename

Valid column names are: x y z a b c u v w diam front back orien
```

To synchronize a standalone *tooledit* with a running LinuxCNC application, the filename must resolve to the same [EMCIO]TOOL_TABLE filename specified in the LinuxCNC INI file.

When using the program *tooledit* while LinuxCNC is running, G-code command execution or other programs may alter tool table data and the tool table file. File changes are detected by *tooledit* and a message is displayed:

```
Warning: File changed by another process
```

The *tooledit* tool table display can be updated to read the modified file with the ReRead button.

The tool table is specified in the INI file with an entry:

```
[EMCIO]TOOL_TABLE = tool_table_filename
```

The tool table file can be edited with any simple text editor (not a word processor).

La IGU AXIS puede usar opcionalmente una configuración de archivo INI para especificar el programa editor de herramientas:

```
[DISPLAY]TOOL_EDITOR = path_to_editor_program
```

By default, the program named *tooledit* is used. This editor supports all tool table parameters, allows addition and deletion of tool entries, and performs a number of validity checks on parameter values.

11.4. Overview of G-Code Programming

11.4.1. Descripción general

The LinuxCNC G-code language is based on the RS274/NGC language. The G-code language is based on lines of code. Each line (also called a *block*) may include commands to do several different things. Lines of code may be collected in a file to make a program.

A typical line of code consists of an optional line number at the beginning followed by one or more *words*. A word consists of a letter followed by a number (or something that evaluates to a number). A word may either give a command or provide an argument to a command. For example, *G1 X3* is a valid line of code with two words. *G1* is a command meaning *move in a straight line at the programmed feed rate to the programmed end point*, and *X3* provides an argument value (the value of X should be 3 at the end of the move). Most LinuxCNC G-code commands start with either G or M (for General and Miscellaneous). The words for these commands are called *G-codes* and *M-codes*. Also common are subroutine codes that begin with *o-* which are called *o-codes*.

El lenguaje LinuxCNC no tiene indicador para el inicio de un programa. El intérprete, sin embargo, se ocupa de los archivos. Un solo programa puede estar en un archivo único, o un programa puede extenderse a través de varios archivos. Un archivo se puede demarcar con porcentajes de la siguiente manera. La primera linea no en blanco de un archivo puede contener solo un signo de porcentaje, *%*, posiblemente rodeado de espacios en blanco y luego (normalmente al final del archivo) puede haber una línea similar. Demarcar un archivo con porcentajes es opcional si el archivo tiene un *M2* o *M30*, pero es requerido si no los tiene. Se notificará un error si un archivo tiene una línea de porcentaje al principio pero no al final. El contenido util de un archivo delimitado por porcentajes termina después de la segunda línea porcentaje. Cualquier cosa después de eso, se ignora.

The LinuxCNC G-code language has two commands (*M2* or *M30*), either of which ends a program. A program may end before the end of a file. Lines of a file that occur after the end of a program are not to be executed. The interpreter does not even read them.

11.4.2. Format of a line

Una línea permitida de código de entrada consiste en lo siguiente, en orden, con la restricción de que hay un máximo (actualmente 256) al número de caracteres permitidos en una línea.

1. an optional block delete character, which is a slash /.
2. an optional line number.
3. Any number of:
    1. words,
    2. parameter settings,
    3. subroutine codes, and
    4. comentarios.
4. an end of line marker (carriage return or line feed or both).

Any input not explicitly allowed is illegal and will cause the Interpreter to signal an error.

Spaces and tabs are allowed anywhere on a line of code and do not change the meaning of the line, except inside comments. This makes some strange-looking input legal. The line *G0X +0. 12 34Y 7* is equivalent to *G0 x+0.1234 Y7*, for example.

Blank lines are allowed in the input. They are to be ignored.

La entrada no distingue entre mayúsculas y minúsculas, excepto en los comentarios, es decir, cualquier letra fuera de un comentario puede estar en mayúsculas o minúsculas sin cambiar el significado de una línea.

11.4.2.1. /: Block Delete

El carácter opcional de eliminación de bloque es la barra / cuando se coloca al principio de una línea. Algunas interfaces de usuario pueden utilizarlo para omitir líneas de código cuando sea necesario. En Axis, la combinación de teclas Alt-m-/ activa o desactiva la eliminación de bloque. Cuando eliminar bloque está activado, cualquier línea que comience con la barra / se omite.

In AXIS, it is also possible to enable block delete with the following icon:

AXIS Block Delete Icon 

11.4.2.2. Número de línea opcional

Un número de línea es la letra N seguida de un entero sin signo, seguida opcionalmente por un punto y otro entero sin signo. Por ejemplo, *N1234* y *N56.78* son números de línea válidos. Pueden ser repetidos o usados fuera de orden, aunque la práctica normal es evitar tal uso. Los números de línea también se pueden omitir, y eso es la práctica normal. No es necesario utilizar un número de línea, pero debe estar en el lugar apropiado si se usa.

---

nota
Line numbers are not recommended. See Best Practices.

11.4.2.3. Words, Parameters, Subroutines, Comments

A word is a letter other than N or O (”o”) followed by a real value.

Words may begin with any of the letters shown in the following Table. The table includes N and O for completeness, even though, as defined above, line numbers and program flow parameters are not words. Several letters (I, J, K, L, P, R) may have different meanings in different contexts. Letters which refer to axis names are not valid on a machine which does not have the corresponding axis.

Cuadro 11.3: Palabras y sus significados

Letter	Meaning
A	A axis of machine
B	B axis of machine
C	C axis of machine
D	Tool radius compensation number
F	Velocidad de alimentación
G	General function (See table G-code Modal Groups)
H	Índice de desplazamiento de la longitud de la herramienta
I	Offset X para arcos y ciclos fijos G87
J	Offset Y para arcos y ciclos fijos G87
K	Z offset for arcs and G87 canned cycles.
	Spindle-Motion Ratio for G33 synchronized movements.
L	generic parameter word for G10, M66 and others
M	Miscellaneous function (See table M-code Modal Groups)
N	Line number (not recommended, see Best Practices)
O	o-codes for program flow control (See o-Codes)
P	Dwell time in canned cycles and with G4.
	Key used with G10.
Q	Feed increment in G73, G83 canned cycles
R	Radio del arco o plano del ciclo fijo
S	Spindle speed
T	Tool selection
U	U axis of machine
V	V axis of machine
W	W axis of machine
X	X axis of machine
Y	Y axis of machine
Z	Z axis of machine

Parameters are identified with a ”#” symbol in front of them. See Parameters Section below.

Also called *o-codes* these provide program flow control (such as if-else logic and callable subroutines) and are covered fully at the page on o-Codes and also below in Subroutine Codes and Parameters.

nota
o-codes are sometimes also called o-words.

Comments can be embedded in a line using parentheses () or for the remainder of a line using a semi-colon. There are also *active* comments like MSG, DEBUG, etc. See the section on comments.

11.4.2.4. End of Line Marker

This is any combination of carriage return or line feed.

11.4.3. Numbers

The following rules are used for (explicit) numbers. In these rules a digit is a single character between 0 and 9.

- A number consists of:
  - an optional plus or minus sign, followed by
  - zero to many digits, followed, possibly, by
  - one decimal point, followed by
  - zero to many digits - provided that there is at least one digit somewhere in the number.
- There are two kinds of numbers:
  - enteros, que no tienen un punto decimal en él,
  - Decimals, that do have a decimal point.
- Numbers may have any number of digits, subject to the limitation on line length. Only about seventeen significant figures will be retained, however (enough for all known applications).
- A non-zero number with no sign but the first character is assumed to be positive.

Observe que los ceros inicial (antes del punto decimal y el primer dígito distinto de cero) y final (después del punto decimal y el último dígito distinto de cero) están permitidos pero no son obligatorios. Un número escrito con los ceros iniciales o finales tendrán el mismo valor cuando se lea como si los ceros adicionales no estuvieran allí.

Numbers used for specific purposes in RS274/NGC are often restricted to some finite set of values or some to some range of values. In many uses, decimal numbers must be close to integers; this includes the values of indices (for parameters and carousel slot numbers, for example), M-codes, and G-codes multiplied by ten. A decimal number which is intended to represent an integer is considered close enough if it is within 0.0001 of an integer value.

11.4.4. Parámetros

The RS274/NGC language supports *parameters* - what in other programming languages would be called *variables*. There are several types of parameter of different purpose and appearance, each described in the following sections. The only value type supported by parameters is floating-point; there are no string, boolean or integer types in G-code like in other programming languages. However, logic expressions can be formulated with boolean operators (*AND*, *OR*, *XOR*, and the comparison operators *EQ*,*NE*,*GT*,*GE*,*LT*,*LE*), and the *MOD*, *ROUND*, *FUP* and *FIX* operators support integer arithmetic.

Parameters differ in syntax, scope, behavior when not yet initialized, mode, persistence and intended use.

Sintaxis

There are three kinds of syntactic appearance:

- *numbered* - #4711
- *named local* - #<localvalue>

- *named global* - #<_globalvalue>

Scope
The scope of a parameter is either global, or local within a subroutine. Subroutine parameters and local named variables have local scope. Global named parameters and numbered parameters starting from number 31 are global in scope. RS274/NGC uses *lexical scoping* - in a subroutine only the local variables defined therein, and any global variables are visible. The local variables of a calling procedure are not visible in a called procedure.

Behavior of uninitialized parameters

- Uninitialized global parameters, and unused subroutine parameters return the value zero when used in an expression.
- Uninitialized named parameters signal an error when used in an expression.

Modo
Most parameters are read/write and may be assigned to within an assignment statement. However, for many predefined parameters this does not make sense, so they are are read-only - they may appear in expressions, but not on the left-hand side of an assignment statement.

Persistence
Cuando LinuxCNC se cierra, los parámetros volátiles pierden su valores. Todos los parámetros excepto los parámetros numerados en el actual rango persistente [1] son volátiles. Los parámetros persistentes se guardan en el archivo .var y son restaurados a sus valores anteriores cuando LinuxCNC se inicia de nuevo. Los parámetros numerados volatiles se restablecen a cero.

Intended Use

- parámetros de usuario - parámetros numerados en el rango 31..5000, y parámetros nombrados globales y locales, excepto parámetros predefinidos. Estan disponible para almacenamiento de uso general de valores de punto flotante, como resultados intermedios, banderas, etc., a lo largo de la ejecución del programa. Son de lectura/escritura (se le puede asignar un valor).
- parámetros de subrutina - se utilizan para mantener los parámetros reales pasados a una subrutina.
- numbered parameters - most of these are used to access offsets of coordinate systems.
- system parameters - used to determine the current running version. They are read-only.

11.4.4.1. Numbered Parameters

A numbered parameter is the pound character # followed by an integer between 1 and (currently) 5602 [2]. The parameter is referred to by this integer, and its value is whatever number is stored in the parameter.

A value is stored in a parameter with the = operator; for example:

```
#3 = 15 (set parameter 3 to 15)
```

A parameter setting does not take effect until after all parameter values on the same line have been found. For example, if parameter 3 has been previously set to 15 and the line *#3=6 G1 X#3* is interpreted, a straight move to a point where X equals 15 will occur and the value of parameter 3 will be 6.

The # character takes precedence over other operations, so that, for example, *#1+2* means the number found by adding 2 to the value of parameter 1, not the value found in parameter 3. Of course,

[1] persistent_range,El rango de parámetros persistentes puede cambiar a medida que avanza el desarrollo. Este rango es actualmente 5161-5390. Se define en *required_parameters array* en el archivo src/emc/rs274ngc/interp_array.cc.

[2] The RS274/NGC interpreter maintains an array of numbered parameters. Its size is defined by the symbol *RS274NGC_MAX_PARAMETERS* in the file src/emc/rs274ngc/interp_internal.hh). This number of numerical parameters may also increase as development adds support for new parameters.

#[1+2] does mean the value found in parameter 3. The # character may be repeated; for example *##2* means the value of the parameter whose index is the (integer) value of parameter 2.

- *31-5000* - G-code user parameters. These parameters are global in the G code file, and available for general use. Volatile.
- *5061-5069* - Coordinates of a G38 probe result (X, Y, Z, A, B, C, U, V & W). Coordinates are in the coordinate system in which the G38 took place. Volatile.
- *5070* - G38 probe result: 1 if success, 0 if probe failed to close. Used with G38.3 and G38.5. Volatile.
- *5161-5169* - ”G28” Home for X, Y, Z, A, B, C, U, V & W. Persistent.
- *5181-5189* - ”G30” Home for X, Y, Z, A, B, C, U, V & W. Persistent.
- *5210* - 1 if ”G52” or ”G92” offset is currently applied, 0 otherwise. Persistent by default; volatile if *DISABLE_G92_PERSISTENCE = 1* in the *[RS274NGC]* section of the INI file.
- *5211-5219* - Shared ”G52” and ”G92” offset for X, Y, Z, A, B, C, U, V & W. Volatile by default; persistent if *DISABLE_G92_PERSISTENCE = 1* in the *[RS274NGC]* section of the INI file.
- *5220* - Coordinate System number 1 - 9 for G54 - G59.3. Persistent.
- *5221-5230* - Coordinate System 1, G54 for X, Y, Z, A, B, C, U, V, W & R. R denotes the XY rotation angle around the Z axis. Persistent.
- *5241-5250* - Coordinate System 2, G55 for X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5261-5270* - Coordinate System 3, G56 for X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5281-5290* - Coordinate System 4, G57 for X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5301-5310* - Coordinate System 5, G58 for X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5321-5330* - Coordinate System 6, G59 for X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5341-5350* - Coordinate System 7, G59.1 for X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5361-5370* - Coordinate System 8, G59.2 for X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5381-5390* - Coordinate System 9, G59.3 for X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5399* - Result of M66 - Check or wait for input. Volatile.
- *5400* - Tool Number. Volatile.
- *5401-5409* - Tool Offsets for X, Y, Z, A, B, C, U, V & W. Set by G43. Volatile.
- *5410* - Tool Diameter. Volatile.
- *5411* - Tool Front Angle. Volatile.
- *5412* - Tool Back Angle. Volatile.
- *5413* - Tool Orientation. Volatile.
- *5420-5428* - Current relative position in the active coordinate system including all offsets and in the current program units for X, Y, Z, A, B, C, U, V & W, volatile.
- *5599* - Flag for controlling the output of (DEBUG,) statements. 1=output, 0=no output; default=1. Volatile.

Numbered Parameters Persistence The values of parameters in the persistent range are retained over time, even if the machining center is powered down. LinuxCNC uses a parameter file to ensure persistence. It is managed by the Interpreter. The Interpreter reads the file when it starts up, and writes the file when it exits.

The format of a parameter file is shown in Table Parameter File Format.

El intérprete espera que el archivo tenga dos columnas. Se salta cualquier línea que no contenga exactamente dos valores numéricos. Se espera que la primera columna contenga un valor entero (el número de parámetro). La segunda columna contiene un número de punto flotante (último valor del parámetro). El valor se representa como un número de punto flotante de doble precisión dentro del intérprete, pero el punto decimal no se requiere en el archivo.

Pueden agregarse al archivo parámetros definidos por el usuario en el rango (31-5000). Dichos parámetros serán leídos por el intérprete y escritos en el archivo al salir.

Missing Parameters in the persistent range will be initialized to zero and written with their current values on the next save operation.

Los números de los parámetros deben organizarse en orden ascendente. Se señalará el error *Archivo de parámetros desordenado* si no están en orden ascendente.

The original file is saved as a backup file when the new file is written.

Cuadro 11.4: Formato de archivo de parámetros

Numeración de parámetros	Valor del parámetro
5161	0.0
5162	0.0

11.4.4.2. Subroutine Codes and Parameters

Subroutine codes, or o-codes (sometimes also called o-words), provide for logic and flow control in NGC programs (as in if-else logic). They are called Subroutine codes because they can also form called subroutines (as in sub-endsub).

See the chapter on o-Codes.

---

nota
If o-codes are used to form subroutines, then o-codes can also call those subroutines and pass up to 30 parameters, which are local to the subroutine and volatile. (Again, see o-Codes for fuller treatment and examples.)

---

nota
While both lower and upper case o- are valid, best practice is using lower case ”o-” because it disambiguates 0 (zero) and O (capital o).

11.4.4.3. Named Parameters

Named parameters work like numbered parameters but are easier to read. All parameter names are converted to lower case and have spaces and tabs removed, so <*param*> and <*P a R am* > refer to the same parameter. Named parameters must be enclosed with < > marks.

#<named parameter> is a local named parameter. By default, a named parameter is local to the scope in which it is assigned. You can't access a local parameter outside of its subroutine. This means that two subroutines can use the same parameter names without fear of one subroutine overwriting the values in another.

#<_parámetro con nombre global> es un parámetro con nombre global. Son accesibles desde las subrutinas llamadas y pueden establecer valores dentro de subrutinas que serán accesibles para el llamante. En lo que respecta al alcance, actúan como parámetros numéricos regulares. No se almacenan en archivos.

Examples:

Declaración de la variable global nombrada

```
#<_endmill_dia> = 0.049
```

Reference to previously declared global variable

```
#<_endmill_rad> = [#<_endmill_dia>/2.0]
```

Mezcla de parámetros literales y con nombre

```
o100 call [0.0] [0.0] [#<_inside_cutout>-#<_endmill_dia>] [#<_Zcut>] [#<_feedrate>]
```

Los parámetros con nombre aparecen cuando se les asigna un valor por primera vez. Los parámetros locales nombrados desaparecen cuando se abandona su alcance, esto es, cuando se vuelve de una subrutina se eliminan todos sus parámetros locales y ya no se puede hacer referencia a ellos.

It is an error to use a non-existent named parameter within an expression, or at the right-hand side of an assignment. Printing the value of a non-existent named parameter with a DEBUG statement - like *(DEBUG, <no_such_parameter>)* will display the string #.

Global parameters, as well as local parameters assigned to at the global level, retain their value once assigned even when the program ends, and have these values when the program is run again.

The EXISTS function tests whether a given named parameter exists.

11.4.4.4. Predefined Named Parameters

The following global read only named parameters are available to access internal state of the interpreter and machine state. They can be used in arbitrary expressions, for instance to control flow of the program with if-then-else statements. Note that new predefined named parameters can be added easily without changes to the source code.

- *#<_vmajor>* - Major package version. If current version was 2.5.2 would return 2.5.
- *#<_vminor>* - Minor package version. If current version was 2.6.2 it would return 0.2.
- *#<_line>* - Sequence number. If running a G-code file, this returns the current line number.
- *#<_motion_mode>* - Return the interpreter's current motion mode:

Motion mode	valor de retorno
G1	10
G2	20
G3	30
G33	330
G38.2	382

Motion mode	valor de retorno
G38.3	383
G38.4	384
G38.5	385
G5.2	52
G73	730
G76	760
G80	800
G81	810
G82	820
G83	830
G84	840
G85	850
G86	860
G87	870
G88	880
G89	890

- *#<_plane>* - returns the value designating the current plane:

Plane	valor de retorno
G17	170
G18	180
G19	190
G17.1	171
G18.1	181
G19.1	191

- *#<_ccomp>* - Status of cutter compensation. Return values:

Modo	valor de retorno
G40	400
G41	410
G41.1	411
G41	410
G42	420
G42.1	421

- *#<_metric>* - Return 1 if G21 is on, else 0.
- *#<_imperial>* - Return 1 if G20 is on, else 0.
- *#<_absolute>* - Return 1 if G90 is on, else 0.
- *#<_incremental>* - Return 1 if G91 is on, else 0.
- *#<_inverse_time>* - Return 1 if inverse feed mode (G93) is on, else 0.
- *#<_units_per_minute>* - Return 1 if Units/minute feed mode (G94) is on, else 0.

- #<_units_per_rev> - Return 1 if Units/revolution mode (G95) is on, else 0.
- #<_coord_system> - Return a float of the current coordinate system name (G54..G59.3). For example if your in G55 coordinate system the return value is 550.000000 and if your in G59.1 the return value is 591.000000.

Modo	valor de retorno
G54	540
G55	550
G56	560
G57	570
G58	580
G59	590
G59.1	591
G59.2	592
G59.3	593

- #<_tool_offset> - Return 1 if tool offset (G43) is on, else 0.
- #<_retract_r_plane> - Return 1 if G98 is set, else 0.
- #<_retract_old_z> - Return 1 if G99 is on, else 0.

11.4.4.5. System Parameters

* <_spindle_rpm_mode> - Return 1 if spindle rpm mode (G97) is on, else 0. * <_spindle_css_mode> - Return 1 if constant surface speed mode (G96) is on, else 0. * <_ijk_absolute_mode> - Return 1 if Absolute Arc distance mode (G90.1) is on, else 0. * <_lathe_diameter_mode> - Return 1 if this is a lathe configuration and diameter (G7) mode is on, else 0. * <_lathe_radius_mode> - Return 1 if this is a lathe configuration and radius (G8) mode is on, else 0. * <_spindle_on> - Return 1 if spindle currently running (M3 or M4) else 0. * <_spindle_cw> - Return 1 if spindle direction is clockwise (M3) else 0. * <_mist> - Return 1 if mist (M7) is on. * <_flood> - Return 1 if flood (M8) is on. * <_speed_override> - Return 1 if feed override (M48 or M50 P1) is on, else 0. * <_feed_override> - Return 1 if feed override (M48 or M51 P1) is on, else 0. * <_adaptive_feed> - Return 1 if adaptive feed (M52 or M52 P1) is on, else 0. * <_feed_hold> - Return 1 if feed hold switch is enabled (M53 P1), else 0. * <_feed> - Return the current value of F, not the actual feed rate. * <_rpm> - Return the current value of S, not the actual spindle speed. * <_x> - Return current relative X coordinate including all offsets. Same as 5420. In a lathe configuration, it always returns radius. * <_y> - Return current relative Y coordinate including all offsets. Same as 5421. * <_z> - Return current relative Z coordinate including all offsets. Same as 5422. * <_a> - Return current relative A coordinate including all offsets. Same as 5423. * <_b> - Return current relative B coordinate including all offsets. Same as 5424. * <_c> - Return current relative C coordinate including all offsets. Same as 5425. * <_u> - Return current relative U coordinate including all offsets. Same as 5426. * <_v> - Return current relative V coordinate including all offsets. Same as 5427. * <_w> - Return current relative W coordinate including all offsets. Same as 5428. * <_abs_x> - Return current absolute X coordinate (G53) including no offsets. * <_abs_y> - Return current absolute Y coordinate (G53) including no offsets. * <_abs_z> - Return current absolute Z coordinate (G53) including no offsets. * <_abs_a> - Return current absolute A coordinate (G53) including no offsets. * <_abs_b> - Return current absolute B coordinate (G53) including no offsets. * <_abs_c> - Return current absolute C coordinate (G53) including no offsets. * <_current_tool> - Return number of the current tool in spindle. Same as 5400. * <_current_pocket> - Return the tooldata index for the current tool. * <_selected_tool> - Return number of the selected tool post a T code. Default -1. * <_selected_pocket> - Return the tooldata index of the selected pocket post a T code. Default -1 (no pocket selected). * <_value> - Return value from the last O-code *return* or *endsub*. Default value 0 if no expression after *return* or *endsub*. Initialized to 0 on program start. *

<_value_returned> - 1.0 if the last O-code *return* or *endsub* returned a value, 0 otherwise. Cleared by the next O-code call. * *<_task>* - 1.0 if the executing interpreter instance is part of milltask, 0.0 otherwise. Sometimes it is necessary to treat this case specially to retain proper preview, for instance when testing the success of a probe (G38.n) by inspecting #5070, which will always fail in the preview interpreter (e.g. Axis). * *<_call_level>* - current nesting level of O-code procedures. For debugging. * *#<_remap_level>* - current level of the remap stack. Each remap in a block adds one to the remap level. For debugging.

11.4.5. HAL pins and INI values

If enabled in the INI file G-code has access to the values of INI file entries and HAL pins.

- *#<_ini[section]name>* Devuelve el valor del elemento correspondiente en el archivo INI.

For example, if the INI file looks like so:

```
[SETUP]
XPOS = 3.145
YPOS = 2.718
```

you may refer to the named parameters `#<_ini[setup]xpos>` and `#<_ini[setup]ypos>` within G-code.

`EXISTS` can be used to test for presence of a given INI file variable:

```
o100 if [EXISTS[#<_ini[setup]xpos>]]
  (debug, [setup]xpos exists: #<_ini[setup]xpos>)
o100 else
  (debug, [setup]xpos does not exist)
o100 endif
```

El valor se lee desde el archivo INI una vez y se almacena en caché del interprete. Estos parámetros son de solo lectura -asignar un valor causa un error de tiempo de ejecución. En código G los nombres no distinguen entre mayúsculas y minúsculas - son convertidos a mayúsculas antes de consultar el archivo INI. Por lo tanto, las entradas INI que contengan caracteres en minúsculas no son accesibles desde código G.

- *#<_hal[HAL item]>* Allows G-code programs to read the values of HAL pins Variable access is read-only, the only way to *set* HAL pins from G-code remains M62-M65, M67, M68 and custom M100-M199 codes. Note that the value read will not update in real-time, typically the value that was on the pin when the G-code program was started will be returned. It is possible to work round this by forcing a state synch. One way to do this is with a dummy M66 command: M66E0L0

Ejemplo:

```
(debug, #<_hal[motion-controller.time]>)
```

Access of HAL items is read-only. Currently, only all-lowercase HAL names can be accessed this way.

`EXISTS` can be used to test for the presence of a given HAL item:

```
o100 if [EXISTS[#<_hal[motion-controller.time]>]]
  (debug, [motion-controller.time] exists: #<_hal[motion-controller.time]>)
o100 else
  (debug, [motion-controller.time] does not exist)
o100 endif
```

Esta característica fue motivada por el deseo de una conexion más fuerte entre componentes de la interfaz de usuario como `GladeVCP` y` PyVCP` para actuar como fuente de parámetros para manejar el comportamiento del archivo NGC. La alternativa - pasar por los pines M6x y conectarlos - tiene un espacio de nombres limitado y no mnemotécnico y es innecesariamente engorroso como mecanismo de comunicación UI/Intérprete.

11.4.6. Expressions

An expression is a set of characters starting with a left bracket *[* and ending with a balancing right bracket *]* . In between the brackets are numbers, parameter values, mathematical operations, and other expressions. An expression is evaluated to produce a number. The expressions on a line are evaluated when the line is read, before anything on the line is executed. An example of an expression is *[1 + acos[0] - [#3 ** [4.0/2]]]*.

11.4.7. Binary Operators

Los operadores binarios solo aparecen dentro de las expresiones. Hay cuatro operaciones matemáticas básicas: suma (+), resta (-), multiplicación (*) y división (/). Hay tres operaciones lógicas: o no exclusivo (*OR*), o exclusivo (*XOR*), y lógico *AND*). La octava operación es la operación de módulo (*MOD*). La novena operación es la operación *potencia* (**), elevar el número a la izquierda de la operación al exponente de la derecha. Los operadores relacionales son igualdad (*EQ*), desigualdad (*NE*), estrictamente mayor que (*GT*), mayor o igual que (*GE*), estrictamente menor que (*LT*), y menor o igual que (*LE*).

The binary operations are divided into several groups according to their precedence. If operations in different precedence groups are strung together (for example in the expression *[2.0 / 3 * 1.5 - 5.5 / 11.0]*), operations in a higher group are to be performed before operations in a lower group. If an expression contains more than one operation from the same group (such as the first / and * in the example), the operation on the left is performed first. Thus, the example is equivalent to: *[[[2.0 / 3] * 1.5] - [5.5 / 11.0]]* , which is equivalent to to *[1.0 - 0.5]* , which is *0.5*.

The logical operations and modulus are to be performed on any real numbers, not just on integers. The number zero is equivalent to logical false, and any non-zero number is equivalent to logical true.

Cuadro 11.9: Operators Precedence

Operators	**Precedence**
**	*highest*
* / MOD	
+ -	
EQ NE GT GE LT LE	
AND OR XOR	*lowest*

11.4.8. Equality and floating-point values

Testing for equality or inequality of two double-precision floating-point values is inherently problematic. The interpreter solves this problem by considering values equal if their absolute difference is less than 1e-6 (this value is defined as *TOLERANCE_EQUAL* in src/emc/rs274ngc/interp_internal.hh).

11.4.9. Funciones

The available functions are shown in following table. Arguments to unary operations which take angle measures (*COS*, *SIN*, and *TAN*) are in degrees. Values returned by unary operations which return angle measures (*ACOS*, *ASIN*, and *ATAN*) are also in degrees.

Cuadro 11.10: G-code Functions

Function Name	Function result
ATAN[arg]/[arg]	Four quadrant inverse tangent
ABS[arg]	Absolute value
ACOS[arg]	Inverse cosine
ASIN[arg]	Inverse sine
COS[arg]	Cosine
EXP[arg]	e raised to the given power
FIX[arg]	Round down to integer
FUP[arg]	Round up to integer
ROUND[arg]	Round to nearest integer
LN[arg]	Base-e logarithm
SIN[arg]	Sine
SQRT[arg]	Square Root
TAN[arg]	Tangent
EXISTS[arg]	Check named Parameter

The *FIX* function rounds towards the left (less positive or more negative) on a number line, so that *FIX[2.8] =2* and *FIX[-2.8] = -3*.

The *FUP* operation rounds towards the right (more positive or less negative) on a number line; *FUP[2.8] = 3* and *FUP[-2.8] = -2*.

The *EXISTS* function checks for the existence of a single named parameter. It takes only one named parameter and returns 1 if it exists and 0 if it does not exist. It is an error if you use a numbered parameter or an expression. Here is an example for the usage of the EXISTS function:

```
o<test> sub
o10 if [EXISTS[#<_global>]]
    (debug, _global exists and has the value #<_global>)
o10 else
    (debug, _global does not exist)
o10 endif
o<test> endsub

o<test> call
#<_global> = 4711
o<test> call
m2
```

11.4.10. Repeated Items

A line may have any number of G words, but two G words from the same modal group may not appear on the same line. See the Modal Groups section for more information.

A line may have zero to four M words. Two M words from the same modal group may not appear on the same line.

For all other legal letters, a line may have only one word beginning with that letter.

If a parameter setting of the same parameter is repeated on a line, *#3=15 #3=6*, for example, only the last setting will take effect. It is silly, but not illegal, to set the same parameter twice on the same line.

Si aparece más de un comentario en una línea, solo el último será usado; cada uno de los otros comentarios será leído y su formato será verificado, pero se ignorará a partir de entonces. Es de esperarse que poner mas de un comentario en una línea sea muy raro.

11.4.11. Item order

The three types of item whose order may vary on a line (as given at the beginning of this section) are word, parameter setting, and comment. Imagine that these three types of item are divided into three groups by type.

The first group (the words) may be reordered in any way without changing the meaning of the line.

If the second group (the parameter settings) is reordered, there will be no change in the meaning of the line unless the same parameter is set more than once. In this case, only the last setting of the parameter will take effect. For example, after the line *#3=15 #3=6* has been interpreted, the value of parameter 3 will be 6. If the order is reversed to *#3=6 #3=15* and the line is interpreted, the value of parameter 3 will be 15.

If the third group (the comments) contains more than one comment and is reordered, only the last comment will be used.

If each group is kept in order or reordered without changing the meaning of the line, then the three groups may be interleaved in any way without changing the meaning of the line. For example, the line *g40 g1 #3=15 (foo) #4=-7.0* has five items and means exactly the same thing in any of the 120 possible orders (such as *#4=-7.0 g1 #3=15 g40 (foo)*) for the five items.

11.4.12. Commands and Machine Modes

Muchos comandos hacen que el controlador cambia de un modo a otro, y el modo permanece activo hasta que algún otro comando lo cambie implícita o explícitamente. Tales comandos se denominan *modales*. Por ejemplo, si el refrigerante está encendido, permanece encendido hasta que sea explícitamente apagado. Los códigos G para movimiento también son modales. Si un comando G1 (movimiento recto) se da en una línea, se ejecutará nuevamente en la línea siguiente si hay una o más palabras de eje disponibles en esa línea, a menos que se dé un comando explícito usando las palabras de eje o cancelando el movimiento.

Non-modal codes have effect only on the lines on which they occur. For example, G4 (dwell) is non-modal.

11.4.13. Polar Coordinates

Polar Coordinates can be used to specify the XY coordinate of a move. The @n is the distance and ^n is the angle. The advantage of this is for things like bolt hole circles which can be done very simply by moving to a point in the center of the circle, setting the offset and then moving out to the first hole then run the drill cycle. Polar Coordinates always are from the current XY zero position. To shift the Polar Coordinates from machine zero use an offset or select a coordinate system.

In Absolute Mode the distance and angle is from the XY zero position and the angle starts with 0 on the X Positive axis and increases in a CCW direction about the Z axis. The code G1 @1^90 is the same as G1 Y1.

In Relative Mode the distance and angle is also from the XY zero position but it is cumulative. This can be confusing at first how this works in incremental mode.

Por ejemplo, si tiene el siguiente programa, puede esperar que sea un patrón cuadrado:

```
F100 G1 @.5 ^90
G91 @.5 ^90
@.5 ^90
@.5 ^90
@.5 ^90
G90 G0 X0 Y0 M2
```

You can see from the following figure that the output is not what you might expect. Because we added 0.5 to the distance each time the distance from the XY zero position increased with each line.

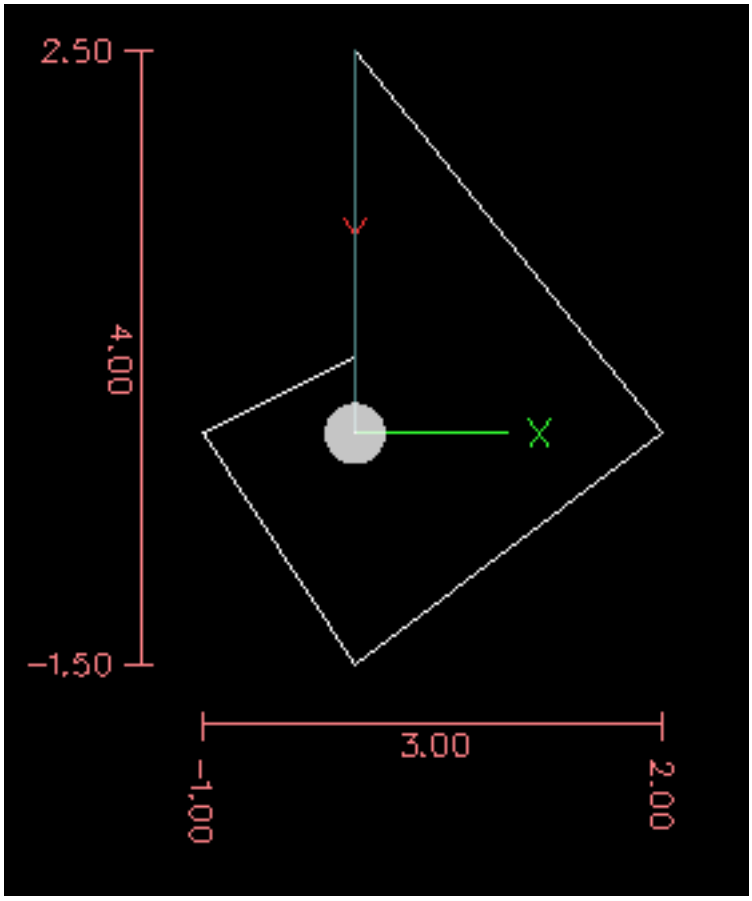


Figura 11.10: Polar Spiral

El siguiente código producirá nuestro patrón cuadrado:

```
F100 G1 @.5 ^90
G91 ^90
^90
^90
^90
G90 G0 X0 Y0 M2
```

As you can see by only adding to the angle by 90 degrees each time the end point distance is the same for each line.

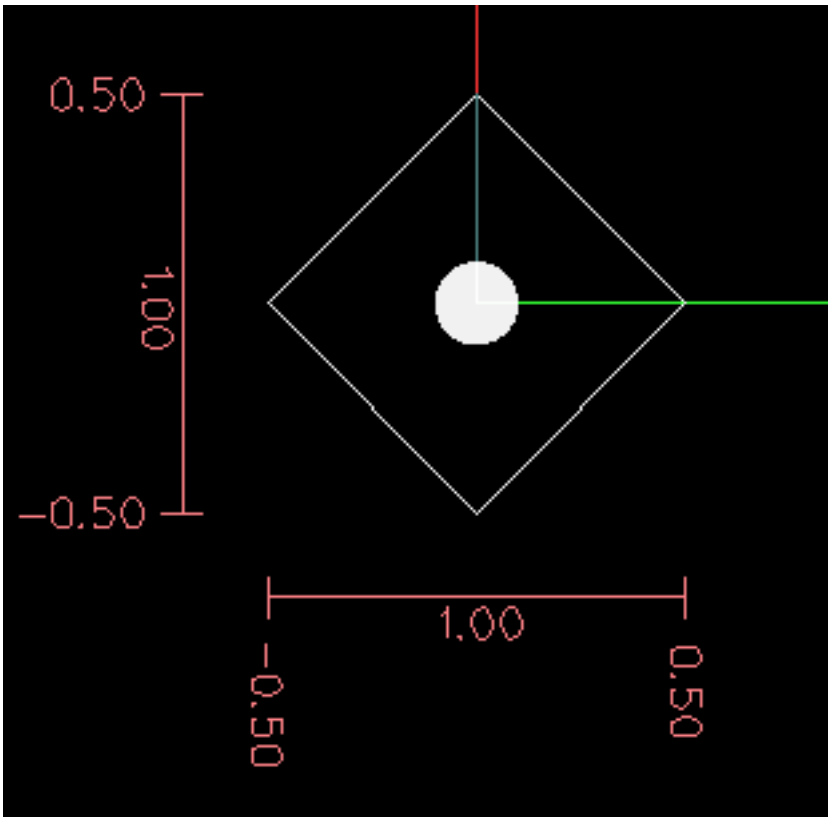


Figura 11.11: Polar Square

It is an error if:

- An incremental move is started at the origin
- A mix of Polar and X or Y words are used

11.4.14. Modal Groups

Los comandos se organizan en conjuntos llamados *grupos modales*, y solo un miembro de un grupo modal puede estar en vigor en cualquier momento. En general, un grupo modal contiene comandos para los cuales es lógicamente imposible que dos miembros entren en vigencia al mismo tiempo, como medida en pulgadas y medida en milímetros. Un centro de mecanizado puede estar en muchos modos al mismo tiempo, con un modo de cada grupo modal en vigor. Los grupos modales se muestran en la tabla siguiente.

Cuadro 11.11: G-code Modal Groups

Modal Group Meaning	**Member Words**
Non-modal codes (Group 0)	G4, G10 G28, G30, G52, G53, G92, G92.1, G92.2, G92.3,
Motion (Group 1)	G0, G1, G2, G3, G33, G38.n, G73, G76, G80, G81 G82, G83, G84, G85, G86, G87, G88, G89
Plane selection (Group 2)	G17, G18, G19, G17.1, G18.1, G19.1
Modo de distancia (Grupo 3)	G90, G91

Cuadro 11.11: (continued)

Modal Group Meaning	**Member Words**
Arc IJK Distance Mode (Group 4)	G90.1, G91.1
Feed Rate Mode (Group 5)	G93, G94, G95
Units (Group 6)	G20, G21
Cutter Diameter Compensation (Group 7)	G40, G41, G42, G41.1, G42.1
Tool Length Offset (Group 8)	G43, G43.1, G49
Canned Cycles Return Mode (Group 10)	G98, G99
Coordinate System (Group 12)	G54, G55, G56, G57, G58, G59, G59.1, G59.2, G59.3
Control Mode (Group 13)	G61, G61.1, G64
Spindle Speed Mode (Group 14)	G96, G97
Lathe Diameter Mode (Group 15)	G7, G8

Cuadro 11.12: Grupos modales de código M

Modal Group Meaning	**Member Words**
Stopping (Group 4)	M0, M1, M2, M30, M60
Pines de E/S (Grupo 5)	(M62-M65 digital output), (M66 digital or analog input), (M67, M68 analog output)
Toolchange (Group 6)	M6 T*n*
Husillo (Grupo 7)	M3, M4, M5
Coolant (Group 8)	(M7 M8 can both be on), M9
Override Switches (Group 9)	M48, M49
User Defined (Group 10)	M100-M199

For several modal groups, when a machining center is ready to accept commands, one member of the group must be in effect. There are default settings for these modal groups. When the machining center is turned on or otherwise re-initialized, the default values are automatically in effect.

El grupo 1, el primero en la tabla, es un grupo de códigos G para movimiento. Uno de ellos siempre está vigente. A ese se le llama modo de movimiento actual.

Es un error colocar un código G del grupo 1 y un código G del grupo 0 en la misma línea si ambos usan palabras de eje. Si un código G del grupo 1 que usa palabras de eje estaba implícitamente vigente en una línea (por haber sido activado en una línea anterior) y un código G del grupo 0 que usa las palabras de eje aparece en la línea, la actividad del código G del grupo 1 es suspendida para esa línea. Los códigos G del grupo 0 que usan palabras de eje son G10, G28, G30, G52 y G92.

It is an error to include any unrelated words on a line with *O*- flow control.

11.4.15. Comentarios

Los comentarios son meramente informativos y no influyen en el comportamiento de la máquina.

Se pueden agregar comentarios a las líneas de código G para ayudar a aclarar la intención del programador. Los comentarios se pueden incrustar en una línea usando paréntesis () o al final de una línea usando un punto y coma. El punto y coma no se trata como el comienzo de un comentario cuando está entre paréntesis.

Comments may appear between words, but not between words and their corresponding parameter. So, *S100(set speed)F200(feed)* is OK while *S(speed)100F(feed)* is not.

Here is an example of a commented program:

```
G0 (Rapid to start) X1 Y1
G0 X1 Y1 (Rapid to start; but don't forget the coolant)
M2 ; End of program.
```

There are several *active* comments which look like comments but cause some action, like *(debug,..)* or *(print,..)*. If there are several comments on a line, only the last comment will be interpreted according to these rules. Hence, a normal comment following an active comment will in effect disable the active comment. For example, *(foo) (debug,#1)* will print the value of parameter *#1*, however *(debug,#1)(foo)* will not.

A comment introduced by a semicolon is by definition the last comment on that line, and will always be interpreted for active comment syntax.

---

nota
Inline comments on O-codes should not be used see the O-code comments section for more information.

11.4.16. Messages

* *(MSG,)* - displays message if *MSG* appears after the left parenthesis and before any other printing characters. Variants of *MSG* which include white space and lower case characters are allowed. The rest of the characters before the right parenthesis are considered to be a message. Messages should be displayed on the message display device of the user interface if provided.

Message Example

```
(MSG, This is a message)
```

11.4.17. Probe Logging

* *(PROBEOPEN filename.txt)* - will open filename.txt and store the 9-number coordinate consisting of XYZABCUVW of each successful straight probe in it. * *(PROBECLOSE)* - will close the open probelog file.

For more information on probing see the G38 section.

11.4.18. Logging

* *(LOGOPEN,filename.txt)* - opens the named log file. If the file already exists, it is truncated. * *(LOGAPPEND,filename)* - opens the named log file. If the file already exists, the data is appended. * *(LOGCLOSE)* - closes an open log file. * *(LOG,)* - everything past the , is written to the log file if it is open. Supports expansion of parameters as described below.

Examples of logging are in *nc_files/examples/smartprobe.ngc* and in *nc_files/ngcgui_lib/rectange_probe.ngc* sample G-code files.

11.4.19. Abort Messages

* *(ABORT,)* - displays a message like *(MSG,)* with the addition of special handling for comment parameters as described below and of aborting the current running process.

11.4.20. Debug Messages

* *(DEBUG,)* - displays a message like *(MSG,)* with the addition of special handling for comment parameters as described below.

11.4.21. Print Messages

* *(PRINT,)* - messages are output to *stderr* with special handling for comment parameters as described below.

11.4.22. Comment Parameters

In the DEBUG, PRINT and LOG comments, the values of parameters in the message are expanded.

For example: to print a named global variable to stderr (the default console window).

Parameters Example

```
(print,endmill dia = #<_endmill_dia>)
(print,value of variable 123 is: #123)
```

Inside the above types of comments, sequences like *#123* are replaced by the value of the parameter 123. Sequences like *#<named parameter>* are replaced by the value of the named parameter. Named parameters will have white space removed from them. So, *#<named parameter>* will be converted to *#<namedparameter>*.

Parameter numbers can be formatted, e.g.:

```
(DEBUG, value = %d#<some_value>)
```

will print the value rounded to an integer.

- %lf is default if there is no formatting string.
- %d = 0 decimals
- %f = four decimals
- %.xf = x (0-9) number of decimals

The formatting will be performed on all parameters in the same line unless changed, i.e., multiple formatting is allowed in one line.

The formatting string does not need to be right beside the parameter.

If the formatting string is created with the wrong pattern it will be printed as characters.

11.4.23. File Requirements

A G-code file must contain one or more lines of G-code and be terminated with a Program End. Any G-code past the program end is not evaluated.

Si no se usa un código de finalización de programa, se utilizarán un par de signos de porcentaje % con el primer signo de porcentaje en la primera línea del archivo seguido de una o más líneas de código G y un segundo signo de porcentaje. Cualquier código pasado el segundo signo de porcentaje no es evaluado.



aviso
Usar % al principio y final de un archivo de código G no hará lo mismo que usar un Final de Programa. Usando %, la máquina estará en cualquier estado en que el programa la dejó, el husillo y el refrigerante aún pueden estar encendidos y cosas como G90/91 se dejan como el último programa las configuró. Si no usa un preámbulo apropiado, el próximo programa podría comenzar en una condición peligrosa.

nota
The file must be created with a text editor like Gedit and not a word processor like Open Office Word Processor.

11.4.24. File Size

El intérprete y el planificador de tareas están cuidadosamente escritos para que el único límite en el tamaño del programa sea la capacidad del disco. Las interfaces TkLinuxCNC y Axis cargan el texto del programa para mostrarlo al usuario, y entonces la RAM se convierte en un factor limitante. En Axis, donde se dibuja la gráfica de vista previa por defecto, el tiempo de redibujo también se convierte en un límite práctico en el tamaño del programa. La vista previa se puede desactivar en Axis para acelerar la carga de grandes programas. En Axis, las secciones de la vista previa se pueden desactivar usando comentarios de control de vista previa.

11.4.25. G-code Order of Execution

The order of execution of items on a line is defined not by the position of each item on the line, but by the following list:

- O-code commands (optionally followed by a comment but no other words allowed on the same line)
- Comment (including message)
- Set feed rate mode (G93, G94).
- Set feed rate (F).
- Set spindle speed (S).
- Select tool (T).
- HAL pin I/O (M62-M68).
- Change tool (M6) and Set Tool Number (M61).
- Spindle on or off (M3, M4, M5).
- Save State (M70, M73), Restore State (M72), Invalidate State (M71).

- Coolant on or off (M7, M8, M9).
- Enable or disable overrides (M48, M49,M50,M51,M52,M53).
- User-defined Commands (M100-M199).
- Dwell (G4).
- Set active plane (G17, G18, G19).
- Set length units (G20, G21).
- Cutter radius compensation on or off (G40, G41, G42)
- Cutter length compensation on or off (G43, G49)
- Coordinate system selection (G54, G55, G56, G57, G58, G59, G59.1, G59.2, G59.3).
- Set path control mode (G61, G61.1, G64)
- Set distance mode (G90, G91).
- Set retract mode (G98, G99).
- Go to reference location (G28, G30) or change coordinate system data (G10) or set axis offsets (G52, G92, G92.1, G92.2, G92.3).
- Perform motion (G0 to G3, G33, G38.n, G73, G76, G80 to G89), as modified (possibly) by G53.
- Stop (M0, M1, M2, M30, M60).

11.4.26. G-code Best Practices

Utilice una precisión decimal adecuada Use at least 3 digits after the decimal when milling in millimeters, and at least 4 digits after the decimal when milling in inches.

In particular, tolerance checks of arcs are done for .001 and .0001 according to the active units.

Use espacios en blanco consistentes G-code is most legible when at least one space appears before words. While it is permitted to insert white space in the middle of numbers, there is no reason to do so.

Utilice arcos de formato central Center-format arcs (which use *I- J- K-* instead of *R-*) behave more consistently than R-format arcs, particularly for included angles near 180 or 360 degrees.

Use a Preamble set modal groups When correct execution of your program depends on modal settings, be sure to set them at the beginning of the part program. Modes can carry over from previous programs and from the MDI commands.

Preámbulo de ejemplo para una fresadora

```
G17 G20 G40 G49 G54 G80 G90 G94
```

G17 use XY plane, G20 inch mode, G40 cancel diameter compensation, G49 cancel length offset, G54 use coordinate system 1, G80 cancel canned cycles, G90 absolute distance mode, G94 feed/minute mode.

Perhaps the most critical modal setting is the distance units—If you do not include G20 or G21, then different machines will mill the program at different scales. Other settings, such as the return mode in canned cycles may also be important.

No ponga demasiadas cosas en una línea Ignore everything in section Order of Execution, and instead write no line of code that is the slightest bit ambiguous.

No establezca y use un parámetro en la misma línea Don’t use and set a parameter on the same line, even though the semantics are well defined. Updating a variable to a new value, such as *#1=[#1+#2]* is OK.

No use números de línea Line numbers offer no benefits. When line numbers are reported in error messages, the numbers refer to the line number in the file, not the N-word value.

When several coordinate systems are moved Considere usar el modo de velocidad de tiempo invertido.

Because the meaning of an *F* word in meters per minute varies depending on the type of axis to be moved and because the amount of removed material does not depend only on the feed rate, it can be simpler to use G93, inverse speed of time, to achieve the removal of desired material.

11.4.27. Linear and Rotary Axis

Because the meaning of an F-word in feed-per-minute mode varies depending on which axes are commanded to move, and because the amount of material removed does not depend only on the feed rate, it may be easier to use G93 inverse time feed mode to achieve the desired material removal rate.

11.4.28. Common Error Messages

- *G-code out of range* - A G-code greater than G99 was used, the scope of G codes in LinuxCNC is 0 - 99. Not every number between 0 and 99 is a valid G-code.
- *Unknown G-code used* - A G-code was used that is not part of the LinuxCNC G-code language.
- *i,j,k word with no Gx to use it* - i, j and k words must be used on the same line as the G-code.
- *Cannot use axis values without a G-code that uses them* - Axis values can not be used on a line without either a modal G-code in effect or a G-code on the same line.
- *File ended with no percent sign or program end* - Every G-code file must end in a M2 or M30 or be wrapped with the percent sign %.

11.5. G-Codes

11.5.1. Conventions

Conventions used in this section

In the G-code prototypes the hyphen (-) stands for a real value and (<>) denotes an optional item.

Si está escrito en un prototipo *L-*, a menudo se hará referencia a - como el *número L*, y así sucesivamente para cualquier otra letra.

In the G-code prototypes the word *axes* stands for any axis as defined in your configuration.

An optional value will be written like this <*L-* >.

A real value may be:

- An explicit number, *4*
- An expression, *[2+2]*
- A parameter value, *#88*
- A unary function value, *acos[0]*

In most cases, if *axis* words are given (any or all of *X Y Z A B C U V W*), they specify a destination point.

Axis numbers are in the currently active coordinate system, unless explicitly described as being in the absolute coordinate system.

Where axis words are optional, any omitted axes will retain their original value.

Any items in the G-code prototypes not explicitly described as optional are required.

Los valores que siguen a las letras a menudo se dan como números explícitos. A menos que se indique lo contrario, los números explícitos pueden ser valores reales. Por ejemplo, *G10 L2* podría igualmente escribirse *G[2*5] L[1+1]*. Si el valor del parámetro 100 era 2, *G10 L#100* también significaría lo mismo.

Si está escrito en un prototipo *L-*, a menudo se hará referencia a - como el *número L*, y así sucesivamente para cualquier otra letra.

11.5.2. G-Code Quick Reference Table

Código	Descripción
G0	Coordinated Motion at Rapid Rate
G1	Coordinated Motion at Feed Rate
G2 G3	Coordinated Helical Motion at Feed Rate
G4	Espera
G5	Cubic Spline
G5.1	Quadratic B-Spline
G5.2	NURBS, add control point
G7	Diameter Mode (lathe)
G8	Radius Mode (lathe)
G10 L0	Reload Tool Table Data
G10 L1	Set Tool Table Entry
G10 L10	Set Tool Table, Calculated, Workpiece
G10 L11	Set Tool Table, Calculated, Fixture
G10 L2	Coordinate System Origin Setting
G10 L20	Coordinate System Origin Setting Calculated
G17 - G19.1	Plane Select
G20 G21	Set Units of Measure
G28 - G28.1	Go to Predefined Position
G30 - G30.1	Go to Predefined Position
G33	Movimiento sincronizado del husillo
G33.1	Rigid Tapping
G38.2 - G38.5	Probing
G40	Cancel Cutter Compensation
G41 G42	Cutter Compensation
G41.1 G42.1	Dynamic Cutter Compensation
G43	Use Tool Length Offset from Tool Table
G43.1	Dynamic Tool Length Offset
G43.2	Apply additional Tool Length Offset
G49	Cancel Tool Length Offset
G52	Local Coordinate System Offset
G53	Move in Machine Coordinates
G54-G59.3	Select Coordinate System (1 - 9)
G61	Exact Path Mode
G61.1	Exact Stop Mode
G64	Path Control Mode with Optional Tolerance
G70	Lathe finishing cycle

Código	Descripción
G71-G72	Lathe roughing cycle
G73	Drilling Cycle with Chip Breaking
G74	Left-hand Tapping Cycle with Dwell
G76	Multi-pass Threading Cycle (Lathe)
G80	Cancel Motion Modes
G81	Drilling Cycle
G82	Drilling Cycle with Dwell
G83	Drilling Cycle with Peck
G84	Right-hand Tapping Cycle with Dwell
G85	Boring Cycle, No Dwell, Feed Out
G86	Boring Cycle, Stop, Rapid Out
G87	Back-boring Cycle *(not yet implemented)*
G88	Boring Cycle, Stop, Manual Out *(not yet implemented)*
G89	Boring Cycle, Dwell, Feed Out
G90 G91	Distance Mode
G90.1 G91.1	Arc Distance Mode
G92	Coordinate System Offset
G92.1 G92.2	Cancel G92 Offsets
G92.3	Restore G92 Offsets
G93 G94 G95	Feed Modes
G96 G97	Spindle Control Mode, Constant Surface vs Rotation Speed (IPM or m/min vs RPM)
G98 G99	Canned Cycle Z Retract Mode

11.5.3. G0 Rapid Move

```
G0 <axes>
```

Para movimiento rápido, programe *G0 axes*, donde todas las palabras de eje son opcionales. *G0* es opcional si el modo de movimiento actual es *G0*. Esto producirá movimiento coordinado hacia el punto de destino a la máxima velocidad rápida (o mas lenta). *G0* se usa típicamente como un movimiento de posicionamiento.

11.5.3.1. Rapid Velocity Rate

The MAX_VELOCITY setting in the INI file [TRAJ] section defines the maximumrapid traverse rate. The maximum rapid traverse rate can be higher than the individual axes MAX_VELOCITY setting during a coordinated move. The maximum rapid traverse rate can be slower than the MAX_VELOCITY setting in the [TRAJ] section if an axis MAX_VELOCITY or trajectory constraints limit it.

G0 Example

```
G90 (set absolute distance mode)
G0 X1 Y-2.3 (Rapid linear move from current location to X1 Y-2.3)
M2 (end program)
```

- See G90 & M2 sections for more information.

If cutter compensation is active, the motion will differ from the above; see the Cutter Compensation section.

Si *G53* está programado en la misma línea, el movimiento también será diferente; Consulte la sección G53 para obtener más información.

The path of a G0 rapid motion can be rounded at direction changes and depends on the trajectory control settings and maximum acceleration of the axes.

It is an error if:

- An axis letter is without a real value.
- Se utiliza una letra de eje que no está configurada.

11.5.4. G1 Linear Move

```
G1 axes
```

For linear (straight line) motion at programmed feed rate (for cutting or not), program *G1 'axes'*, where all the axis words are optional. The *G1* is optional if the current motion mode is *G1*. This will produce coordinated motion to the destination point at the current feed rate (or slower).

G1 Example

```
G90 (set absolute distance mode)
G1 X1.2 Y-3 F10 (linear move at a feed rate of 10 from current position to X1.2 Y-3)
Z-2.3 (linear move at same feed rate from current position to Z-2.3)
Z1 F25 (linear move at a feed rate of 25 from current position to Z1)
M2 (end program)
```

- See G90 & F & M2 sections for more information.

If cutter compensation is active, the motion will differ from the above; see the Cutter Compensation section.

Si *G53* está programado en la misma línea, el movimiento también será diferente; Consulte la sección G53 para obtener más información.

It is an error if:

- No feed rate has been set.
- An axis letter is without a real value.
- An axis letter is used that is not configured

11.5.5. G2, G3 Arc Move

```
G2 or G3 axes offsets (center format)
G2 or G3 axes R- (radius format)
G2 or G3 offsets|R- <P-> (full circles)
```

A circular or helical arc is specified using either *G2* (clockwise arc) or *G3* (counterclockwise arc) at the current feed rate. The direction (CW, CCW) is as viewed from the positive end of the axis about which the circular motion occurs.

The axis of the circle or helix must be parallel to the X, Y, or Z axis of the machine coordinate system. The axis (or, equivalently, the plane perpendicular to the axis) is selected with *G17* (Z-axis, XY-plane), *G18* (Y-axis, XZ-plane), or *G19* (X-axis, YZ-plane). Planes *17.1*, *18.1*, and *19.1* are not currently supported. If the arc is circular, it lies in a plane parallel to the selected plane.

To program a helix, include the axis word perpendicular to the arc plane, for example, if in the *G17* plane, include a *Z* word. This will cause the *Z* axis to move to the programmed value during the circular *XY* motion.

To program an arc that gives more than one full turn, use the *P* word specifying the number of full turns plus the programmed arc. The *P* word must be an integer. If *P* is unspecified, the behavior is as if *P1* was given that is, only one full or partial turn will result. For example, if a 180 degree arc is programmed with a P2, the resulting motion will be 1 1/2 rotations. For each P increment above 1 an extra full circle is added to the programmed arc. Multi turn helical arcs are supported and give motion useful for milling holes or threads.



aviso
If the pitch of the helix is very small (less than the naive CAM tolerance) then the helix might be converted into a straight line. Bug #222

If a line of code makes an arc and includes rotary axis motion, the rotary axes turn at a constant rate so that the rotary motion starts and finishes when the XYZ motion starts and finishes. Lines of this sort are hardly ever programmed.

If cutter compensation is active, the motion will differ from the above; see the Cutter Compensation section.

The arc center is absolute or relative as set by G90.1 or G91.1 respectively.

Two formats are allowed for specifying an arc: Center Format and Radius Format.

It is an error if:

- No feed rate has been set.
- The P word is not an integer.

11.5.5.1. Center Format Arcs

Center format arcs are more accurate than radius format arcs and are the preferred format to use.

The end point of the arc along with the offset to the center of the arc from the current location are used to program arcs that are less than a full circle. It is OK if the end point of the arc is the same as the current location.

The offset to the center of the arc from the current location and optionally the number of turns are used to program full circles.

When programming arcs an error due to rounding can result from using a precision of less than 4 decimal places (0.0000) for inch and less than 3 decimal places (0.000) for millimeters.

Incremental Arc Distance Mode Arc center offsets are a relative distance from the start location of the arc. Incremental Arc Distance Mode is default.

One or more axis words and one or more offsets must be programmed for an arc that is less than 360 degrees.

No axis words and one or more offsets must be programmed for full circles. The *P* word defaults to 1 and is optional.

For more information on 'Incremental Arc Distance Mode see the G91.1 section.

Absolute Arc Distance Mode Arc center offsets are the absolute distance from the current 0 position of the axis.

Una o mas palabras de eje y *ambos* offsets deben programarse para arcos de menos de 360 grados.

No axis words and both offsets must be programmed for full circles. The *P* word defaults to 1 and is optional.

For more information on 'Absolute Arc Distance Mode see the G90.1 section.

XY-plane (G17)

```
G2 or G3 <X- Y- Z- I- J- P->
```

- *Z* - helix
- *I* - X offset
- *J* - Y offset
- *P* - number of turns

XZ-plane (G18)

```
G2 or G3 <X- Z- Y- I- K- P->
```

- *Y* - helix
- *I* - X offset
- *K* - Z offset
- *P* - number of turns

YZ-plane (G19)

```
G2 or G3 <Y- Z- X- J- K- P->
```

- *X* - helix
- *J* - Y offset
- *K* - Z offset
- *P* - number of turns

It is an error if:

- No feed rate is set with the F word.
- No offsets are programmed.
- When the arc is projected on the selected plane, the distance from the current point to the center differs from the distance from the end point to the center by more than (.05 inch/.5 mm) OR ((.0005 inch/.005mm) AND .1 % of radius).

Deciphering the Error message *Radius to end of arc differs from radius to start:*

- *start* - the current position
- *center* - the center position as calculated using the i, j, or k words
- *end* - the programmed end point
- *r1* - radius from the start position to the center
- *r2* - radius from the end position to the center

11.5.5.2. Center Format Examples

Calculating arcs by hand can be difficult at times. One option is to draw the arc with a CAD program to get the coordinates and offsets. Keep in mind the tolerance mentioned above, you may have to change the precision of your CAD program to get the desired results. Another option is to calculate the coordinates and offset using formulas. As you can see in the following figures a triangle can be formed from the current position the end position and the arc center.

En la siguiente figura puede ver que la posición de inicio es X0 Y0 y la posición final es X1 Y1. La posición central del arco está en X1 Y0. Esto da un offset desde la posición inicial de 1 en el eje X y 0 en el eje Y. En este caso solo se necesita un offset I.

G2 Example Line

```
G0 X0 Y0
G2 X1 Y1 I1 F10 (clockwise arc in the XY plane)
```

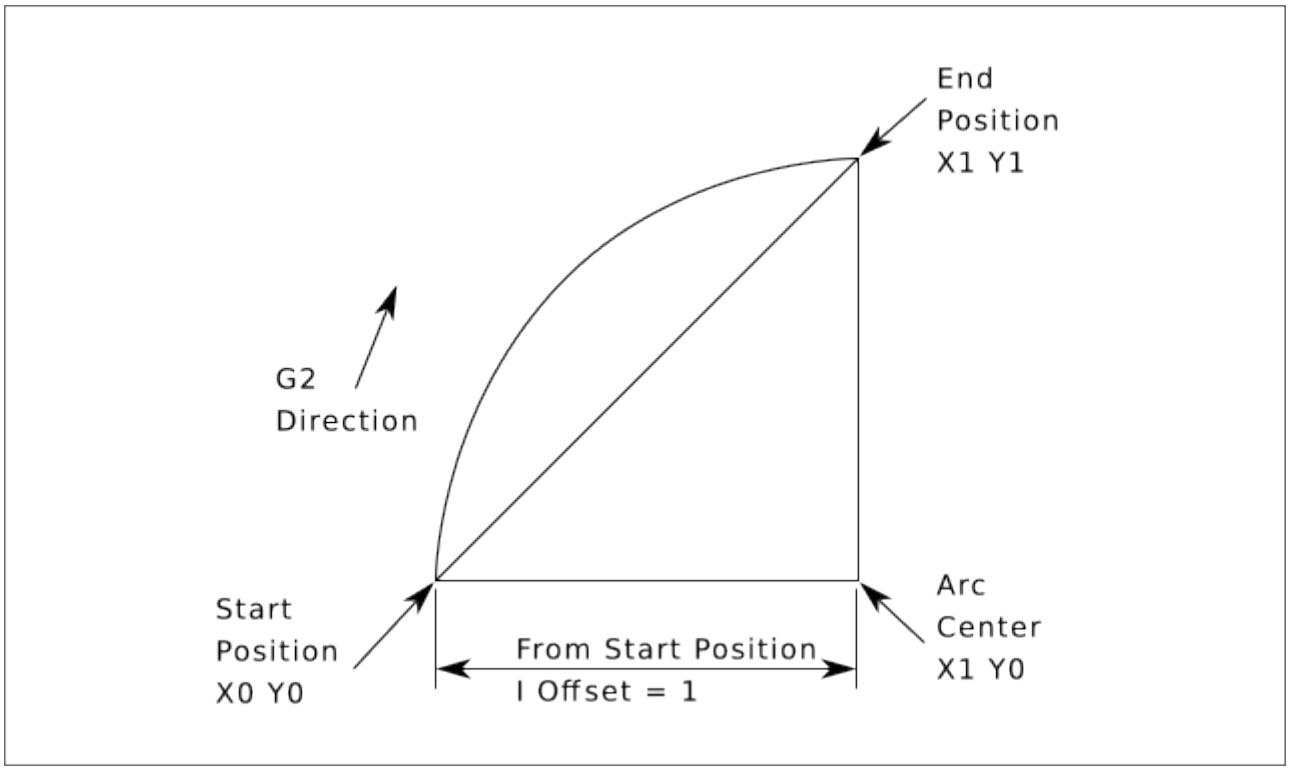


Figura 11.12: G2 Example

In the next example we see the difference between the offsets for Y if we are doing a G2 or a G3 move. For the G2 move the start position is X0 Y0, for the G3 move it is X0 Y1. The arc center is at X1 Y0.5 for both moves. The G2 move the J offset is 0.5 and the G3 move the J offset is -0.5.

G2-G3 Example Line

```
G0 X0 Y0
G2 X0 Y1 I1 J0.5 F25 (clockwise arc in the XY plane)
G3 X0 Y0 I1 J-0.5 F25 (counterclockwise arc in the XY plane)
```

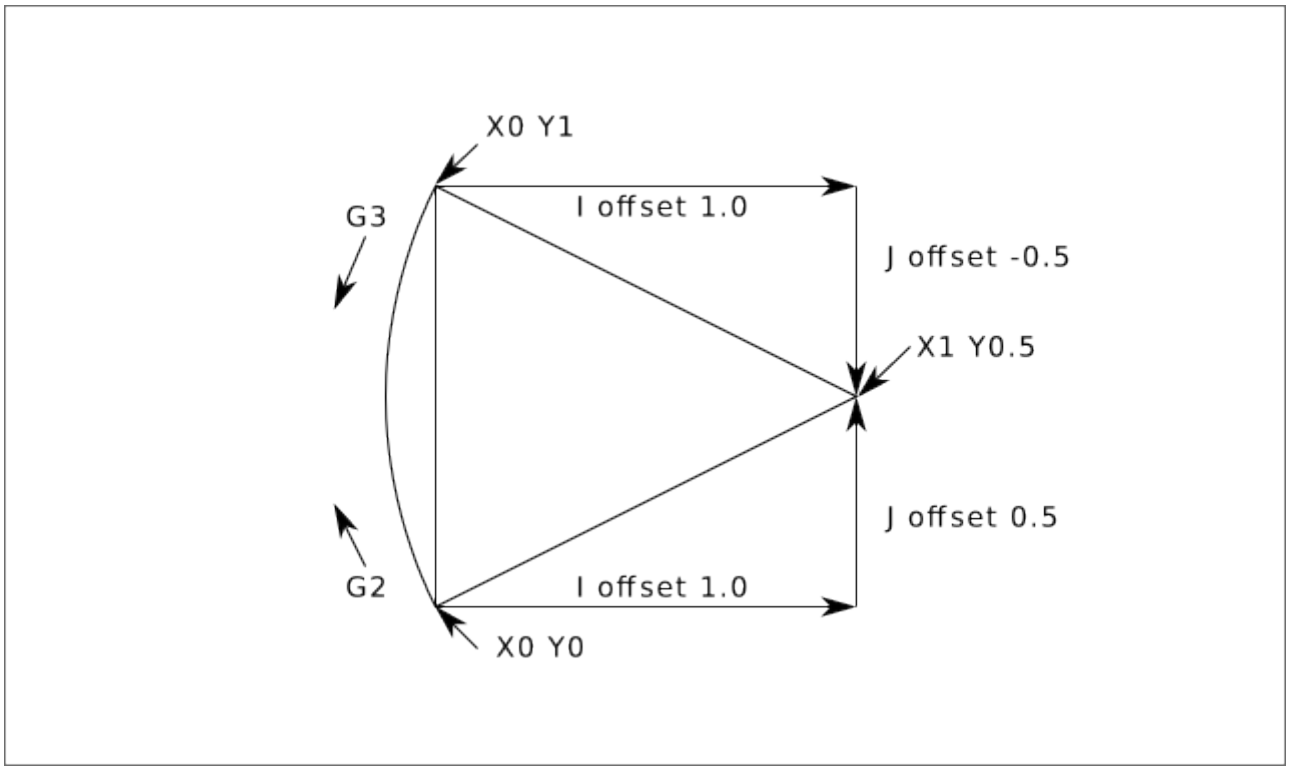


Figura 11.13: G2-G3 Example

In the next example we show how the arc can make a helix in the Z axis by adding the Z word.

G2 Example Helix

```
G0 X0 Y0 Z0
G17 G2 X10 Y16 I3 J4 Z-1 (helix arc with Z added)
```

In the next example we show how to make more than one turn using the P word.

P word Example

```
G0 X0 Y0 Z0
G2 X0 Y1 Z-1 I1 J0.5 P2 F25
```

In the center format, the radius of the arc is not specified, but it may be found easily as the distance from the center of the circle to either the current point or the end point of the arc.

11.5.5.3. Radius Format Arcs

```
G2 or G3 axes R- <P->
```

- *R* - radius from current position

It is not good practice to program radius format arcs that are nearly full circles or nearly semicircles because a small change in the location of the end point will produce a much larger change in the location of the center of the circle (and, hence, the middle of the arc). The magnification effect is large enough that rounding error in a number can produce out-of-tolerance cuts. For instance, a 1 % displacement of the endpoint of a 180 degree arc produced a 7 % displacement of the point 90 degrees

along the arc. Nearly full circles are even worse. Other size arcs (in the range tiny to 165 degrees or 195 to 345 degrees) are OK.

En el formato radio, las coordenadas del punto final del arco en el plano seleccionado se especifica junto con el radio del arco. Programe los ejes *G2 ejes '* R-*(o use 'G3* en lugar de *G2*). R es el radio. Las palabras de eje son todas opcionales, excepto que debe ser usada al menos una de las dos palabras para los ejes del plano seleccionado. El número R es el radio. Un radio positivo indica que el arco gira menos de 180 grados, mientras que un radio negativo indica un giro de más de 180 grados. Si el arco es helicoidal, también se especifica el valor del punto final del arco en el eje de coordenadas paralelo al eje de la hélice.

It is an error if:

- both of the axis words for the axes of the selected plane are omitted
- the end point of the arc is the same as the current point.

G2 Example Line

```
G17 G2 X10 Y15 R20 Z5 (radius format with arc)
```

The above example makes a clockwise (as viewed from the positive Z-axis) circular or helical arc whose axis is parallel to the Z-axis, ending where X=10, Y=15, and Z=5, with a radius of 20. If the starting value of Z is 5, this is an arc of a circle parallel to the XY-plane; otherwise it is a helical arc.

11.5.6. G4 Espera

```
G4 P-
```

- *P* - seconds to dwell (floating point)

The P number is the time in seconds that all axes will remain unmoving. The P number is a floating point number so fractions of a second may be used. G4 does not affect spindle, coolant and any I/O.

G4 Example Line

```
G4 P0.5 (wait for 0.5 seconds before proceeding)
```

It is an error if:

- the P number is negative or not specified.

11.5.7. G5 Cubic Spline

```
G5 X- Y- <I- J-> P- Q-
```

- *I* - X incremental offset from start point to first control point
- *J* - Y incremental offset from start point to first control point
- *P* - X incremental offset from end point to second control point
- *Q* - Y incremental offset from end point to second control point

G5 creates a cubic B-spline in the XY plane with the X and Y axes only. P and Q must both be specified for every G5 command.

Para el primer comando G5 en una serie de comandos G5, I y J deben especificarse. Para los comandos G5 posteriores, tanto I como J deben especificarse ambos o ninguno. Si I y J no están especificados, la dirección inicial de este cubico coincidirá automáticamente con la dirección final del anterior (como si I y J fueran la negación de P y Q del anterior).

For example, to program a curvy N shape:

G5 Sample initial cubic spline

```
G90 G17
G0 X0 Y0
G5 I0 J3 P0 Q-3 X1 Y1
```

A second curvy N that attaches smoothly to this one can now be made without specifying I and J:

G5 Sample subsequent cubic spline

```
G5 P0 Q-3 X2 Y2
```

It is an error if:

- P y Q no están especificados.
- Solo se especifica uno de I o J.
- I o J no están especificados en el primero de una serie de comandos G5.
- Se especifica un eje distinto de X o Y.
- El plano activo no es G17.

11.5.8. G5.1 Quadratic Spline

```
G5.1 X- Y- I- J-
```

- *I* - X incremental offset from start point to control point
- *J* - Y incremental offset from start point to control point

G5.1 creates a quadratic B-spline in the XY plane with the X and Y axis only. Not specifying I or J gives zero offset for the unspecified axis, so one or both must be given.

For example, to program a parabola, through the origin, from X-2 Y4 to X2 Y4:

G5.1 Sample quadratic spline

```
G90 G17
G0 X-2 Y4
G5.1 X2 I2 J-8
```

It is an error if:

- both I and J offset are unspecified or zero
- An axis other than X or Y is specified
- The active plane is not G17

11.5.9. G5.2 G5.3 NURBS Block

```
G5.2 <P-> <X- Y-> <L->
X- Y- <P->
...
G5.3
```



aviso
G5.2, G5.3 is experimental and not fully tested.

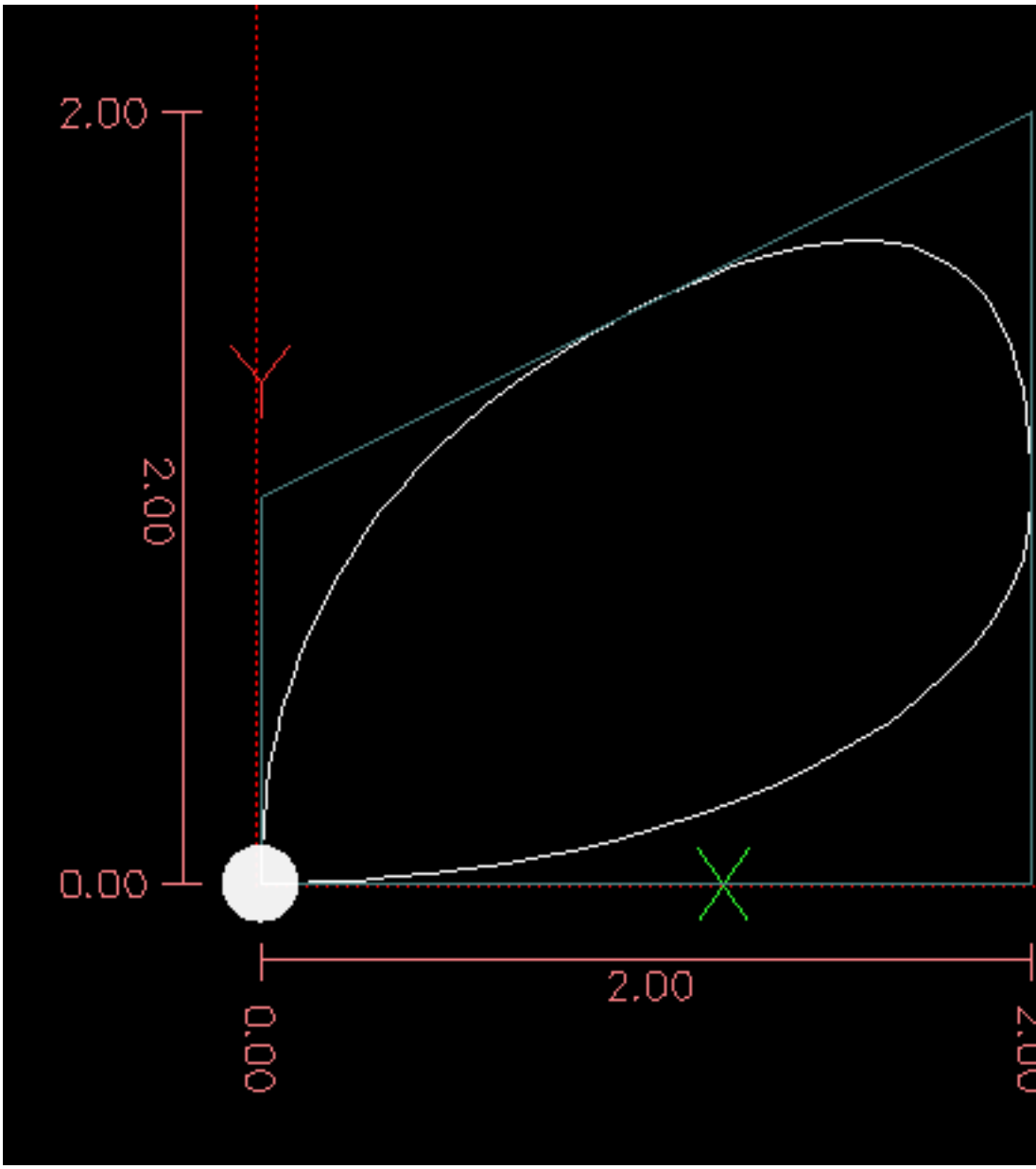
G5.2 is for opening the data block defining a NURBS and G5.3 for closing the data block. In the lines between these two codes the curve control points are defined with both their related *weights* (P) and the parameter (L) which determines the order of the curve.

The current coordinate, before the first G5.2 command, is always taken as the first NURBS control point. To set the weight for this first control point, first program G5.2 P- without giving any X Y.

The default weight if P is unspecified is 1. The default order if L is unspecified is 3.

G5.2 Example

```
G0 X0 Y0 (rapid move)
F10 (set feed rate)
G5.2 P1 L3
     X0 Y1 P1
     X2 Y2 P1
     X2 Y0 P1
     X0 Y0 P2
G5.3
; The rapid moves show the same path without the NURBS Block
G0 X0 Y1
   X2 Y2
   X2 Y0
   X0 Y0
M2
```



Sample NURBS Output

More information on NURBS can be found here:

https://wiki.linuxcnc.org/cgi-bin/wiki.pl?NURBS

11.5.10. G7 Lathe Diameter Mode

```
G7
```

Program G7 to enter the diameter mode for axis X on a lathe. When in the diameter mode the X axis moves on a lathe will be 1/2 the distance to the center of the lathe. For example X1 would move the cutter to 0.500” from the center of the lathe thus giving a 1” diameter part.

11.5.11. G8 Lathe Radius Mode

```
G8
```

Program G8 to enter the radius mode for axis X on a lathe. When in Radius mode the X axis moves on a lathe will be the distance from the center. Thus a cut at X1 would result in a part that is 2” in diameter. G8 is default at power up.

11.5.12. G10 L0 Reload Tool Table Data

```
G10 L0
```

G10 L0 reload all tool table data. Requires that there is no current tool loaded in spindle.

---

nota
When using G10 L0, tool parameters (#5401-#5413) will be updated immediately and any altered tool diameters will be used for subsequent G41,42 cutter radius compensation commands. Existing G43 tool length compensation values will remain in effect until updated by new G43 commands.

11.5.13. G10 L1 Set Tool Table

```
G10 L1 P- axes <R- I- J- Q->
```

- *P* - tool number
- *R* - radius of tool
- *I* - front angle (lathe)
- *J* - back angle (lathe)
- *Q* - orientation (lathe)

G10 L1 sets the tool table for the *P* tool number to the values of the words.

A valid G10 L1 rewrites and reloads the tool table for the specified tool.

G10 L1 Example Line

```
G10 L1 P1 Z1.5 (set tool 1 Z offset from the machine origin to 1.5)
G10 L1 P2 R0.015 Q3 (lathe example setting tool 2 radius to 0.015 and orientation to 3)
```

It is an error if:

- Cutter Compensation is on
- The P number is unspecified
- The P number is not a valid tool number from the tool table
- The P number is 0

For more information on cutter orientation used by the *Q* word, see the Lathe Tool Orientation diagram.

11.5.14. G10 L2 Set Coordinate System

```
G10 L2 P- <axes R->
```

- *P* - coordinate system (0-9)
- *R* - rotation about the Z axis

G10 L2 offsets the origin of the axes in the coordinate system specified to the value of the axis word. The offset is from the machine origin established during homing. The offset value will replace any current offsets in effect for the coordinate system specified. Axis words not used will not be changed.

Program P0 to P9 to specify which coordinate system to change.

Cuadro 11.14: Coordinate System

P Value	Coordinate System	Códigos G
0	Active	n/a
1	1	G54
2	2	G55
3	3	G56
4	4	G57
5	5	G58
6	6	G59
7	7	G59.1
8	8	G59.2
9	9	G59.3

Optionally program R to indicate the rotation of the XY axis around the Z axis. The direction of rotation is CCW as viewed from the positive end of the Z axis.

All axis words are optional.

Being in incremental distance mode (G91) has no effect on *G10 L2*.

Conceptos importantes:

- `G10 L2 Pn` does not change from the current coordinate system to the one specified by P, you have to use G54-59.3 to select a coordinate system.
- When a rotation is in effect jogging an axis will only move that axis in a positive or negative direction and not along the rotated axis.
- If a *G52* local offset or *G92* origin offset was in effect before *G10 L2*, it will continue to be in effect afterwards.
- When programming a coordinate system with R, any *G52* or *G92* will be applied **after** the rotation.
- The coordinate system whose origin is set by a *G10* command may be active or inactive at the time the *G10* is executed. If it is currently active, the new coordinates take effect immediately.

It is an error if:

- The P number does not evaluate to an integer in the range 0 to 9.

- An axis is programmed that is not defined in the configuration.

G10 L2 Example Line

```
G10 L2 P1 X3.5 Y17.2
```

In the above example the origin of the first coordinate system (the one selected by *G54*) is set to be X=3.5 and Y=17.2. Because only X and Y are specified, the origin point is only moved in X and Y; the other coordinates are not changed.

G10 L2 Example Line

```
G10 L2 P1 X0 Y0 Z0 (clear offsets for X,Y & Z axes in coordinate system 1)
```

The above example sets the XYZ coordinates of the coordinate system 1 to the machine origin.

The coordinate system is described in the Coordinate System section.

11.5.15. G10 L10 Set Tool Table

```
G10 L10 P- axes <R- I- J- Q->
```

- *P* - tool number
- *R* - radius of tool
- *I* - front angle (lathe)
- *J* - back angle (lathe)
- *Q* - orientation (lathe)

G10 L10 changes the tool table entry for tool P so that if the tool offset is reloaded, with the machine in its current position and with the current G5x and G52/G92 offsets active, the current coordinates for the given axes will become the given values. The axes that are not specified in the G10 L10 command will not be changed. This could be useful with a probe move as described in the G38 section.

G10 L10 Example

```
T1 M6 G43 (load tool 1 and tool length offsets)
G10 L10 P1 Z1.5 (set the current position for Z to be 1.5)
G43 (reload the tool length offsets from the changed tool table)
M2 (end program)
```

- See T & M6, and G43/G43.1 sections for more information.

It is an error if:

- Cutter Compensation is on
- The P number is unspecified
- The P number is not a valid tool number from the tool table
- The P number is 0

11.5.16. G10 L11 Set Tool Table

```
G10 L11 P- axes <R- I- J- Q->
```

- *P* - tool number
- *R* - radius of tool
- *I* - front angle (lathe)
- *J* - back angle (lathe)
- *Q* - orientation (lathe)

G10 L11 is just like G10 L10 except that instead of setting the entry according to the current offsets, it is set so that the current coordinates would become the given value if the new tool offset is reloaded and the machine is placed in the G59.3 coordinate system without any G52/G92 offset active.

This allows the user to set the G59.3 coordinate system according to a fixed point on the machine, and then use that fixture to measure tools without regard to other currently-active offsets. It is an error if:

- Cutter Compensation is on
- The P number is unspecified
- The P number is not a valid tool number from the tool table
- The P number is 0

11.5.17. G10 L20 Set Coordinate System

```
G10 L20 P- axes
```

- *P* - coordinate system (0-9)

G10 L20 is similar to G10 L2 except that instead of setting the offset/entry to the given value, it is set to a calculated value that makes the current coordinates become the given value.

G10 L20 Example Line

```
G10 L20 P1 X1.5 (set the X axis current location in coordinate system 1 to 1.5)
```

It is an error if:

- The P number does not evaluate to an integer in the range 0 to 9.
- An axis is programmed that is not defined in the configuration.

11.5.18. G17 - G19.1 Plane Select

These codes set the current plane as follows:

- *G17* - XY (default)
- *G18* - ZX
- *G19* - YZ
- *G17.1* - UV
- *G18.1* - WU
- *G19.1* - VW

The UV, WU and VW planes do not support arcs.

It is a good idea to include a plane selection in the preamble of each G-code file.

Los efectos de tener un plano seleccionado se discuten en las secciones Arcos G2 G3 y G81 G89.

11.5.19. G20, G21 Units

- *G20* - to use inches for length units.
- *G21* - to use millimeters for length units.

It is a good idea to include units in the preamble of each G-code file.

11.5.20. G28, G28.1 Go/Set Predefined Position



aviso
Only use G28 when your machine is homed to a repeatable position and the desired G28 position has been stored with G28.1.

G28 usa los valores almacenados en los parámetros 5161-5169 como el punto final X Y Z A B C U V W a donde moverse. Los valores de los parametros son coordenadas máquina *absolutas* en las *unidades* de máquina nativas especificadas en el archivo INI. Todos los ejes definidos en el archivo INI se moverán cuando se emite un G28. Si no se almacenan posiciones con G28.1, todos los ejes irán al origen de máquina.

- *G28* - makes a rapid move from the current position to the *absolute* position of the values in parameters 5161-5166.
- *G28 axes* - makes a rapid move to the position specified by *axes* including any offsets, then will make a rapid move to the *absolute* position of the values in parameters 5161-5166 for all *axes* specified. Any *axis* not specified will not move.
- *G28.1* - stores the current *absolute* position into parameters 5161-5166.

G28 Example Line

```
G28 Z2.5 (rapid to Z2.5 then to Z location specified in #5163)
```

It is an error if :

- Cutter Compensation is turned on

11.5.21. G30, G30.1 Go/Set Predefined Position



aviso
Only use G30 when your machine is homed to a repeatable position and the desired G30 position has been stored with G30.1.

G30 functions the same as G28 but uses the values stored in parameters 5181-5189 as the X Y Z A B C U V W final point to move to. The parameter values are *absolute* machine coordinates in the native machine *units* as specified in the INI file. All axes defined in the INI file will be moved when a G30 is issued. If no positions are stored with G30.1 then all axes will go to the machine origin.

nota
G30 parameters will be used to move the tool when a M6 is programmed if TOOL_CHANGE_AT_G30=1 is in the [EMCIO] section of the INI file.

- *G30* - makes a rapid move from the current position to the *absolute* position of the values in parameters 5181-5189.
- *G30 axes* - makes a rapid move to the position specified by *axes* including any offsets, then will make a rapid move to the *absolute* position of the values in parameters 5181-5189 for all *axes* specified. Any *axis* not specified will not move.
- *G30.1* - stores the current absolute position into parameters 5181-5186.

G30 Example Line

```
G30 Z2.5 (rapid to Z2.5 then to the Z location specified in #5183)
```

It is an error if :

- Cutter Compensation is turned on

11.5.22. G33 Spindle Synchronized Motion

```
G33 X- Y- Z- K- $-
```

- *K* - distance per revolution

For spindle-synchronized motion in one direction, code *G33 X- Y- Z- K-* where K gives the distance moved in XYZ for each revolution of the spindle. For instance, if starting at *Z=0*, *G33 Z-1 K.0625* produces a 1 inch motion in Z over 16 revolutions of the spindle. This command might be part of a program to produce a 16TPI thread. Another example in metric, *G33 Z-15 K1.5* produces a movement of 15mm while the spindle rotates 10 times for a thread of 1.5mm.

The (optional) $ argument sets which spindle the motion is synchronised to (default is zero). For example G33 Z10 K1 $1 will move the spindle in synchrony with the spindle.N.revs HAL pin value.

Spindle-synchronized motion waits for the spindle index and spindle at speed pins, so multiple passes line up. *G33* moves end at the programmed endpoint. G33 could be used to cut tapered threads or a fusee.

Todas las palabras del eje son opcionales, pero se debe utilizar al menos una.

nota
K follows the drive line described by *X- Y- Z-*. K is not parallel to the Z axis if X or Y endpoints are used for example when cutting tapered threads.

Technical Info

Al comienzo de cada pasada G33, LinuxCNC usa la velocidad del eje y los límites de aceleración de la máquina para calcular cuánto tiempo llevará acelerar Z después del pulso índice y determina cuántos grados rotará el husillo durante ese tiempo. Luego agrega ese ángulo a la posición index y calcula la posición Z utilizando el ángulo de husillo corregido. Eso significa que Z alcanzará la posición correcta justo al terminar de acelerar a la velocidad adecuada y puede comenzar de inmediato cortando un buen hilo.

HAL Connections The pin *spindle.N.at-speed* must be set or driven true for the motion to start. Additionally spindle.N.revs must increase by 1 for each revolution of the spindle and the spindle.N.index-enable pin must be connected to an encoder (or resolver) counter which resets index-enable once per rev.

See the Integrators Manual for more information on spindle synchronized motion.

G33 Example

```
G90 (absolute distance mode)
G0 X1 Z0.1 (rapid to position)
S100 M3 (start spindle turning)
G33 Z-2 K0.125 (move Z axis to -2 at a rate to equal 0.125 per revolution)
G0 X1.25 (rapid move tool away from work)
Z0.1 (rapid move to starting Z position)
M2 (end program)
```

- See G90 & G0 & M2 sections for more information.

It is an error if:

- Todas las palabras del eje se omiten.
- El husillo no gira cuando se ejecuta este comando.
- El movimiento lineal solicitado excede los límites de velocidad de la máquina debido a la velocidad del husillo.

11.5.23. G33.1 Rigid Tapping

```
G33.1 X- Y- Z- K- I- $-
```

- *K* - distance per revolution
- *I* - optional spindle speed multiplier for faster return move
- *$* - optional spindle selector



aviso
For Z only tapping preposition the XY location prior to calling G33.1 and only use a Z word in the G33.1. If the coordinates specified are not the current coordinates when calling G33.1 for tapping the move will not be along the Z axis but will be a coordinated, spindle-synchronized move from the current location to the location specified and back.

For rigid tapping (spindle synchronized motion with return), code *G33.1 X- Y- Z- K-* where *K-* gives the distance moved for each revolution of the spindle.

A rigid tapping move consists of the following sequence:

- A move from the current coordinate to the specified coordinate, synchronized with the selected spindle at the given ratio and starting from the current coordinate with a spindle index pulse.
- When reaching the endpoint, a command to reverse the spindle, and speed up by a factor set by the multiplier (e.g., from clockwise to counterclockwise).
- Continued synchronized motion beyond the specified end coordinate until the spindle actually stops and reverses.
- Continued synchronized motion back to the original coordinate.
- When reaching the original coordinate, a command to reverse the spindle a second time (e.g., from counterclockwise to clockwise).
- Continued synchronized motion beyond the original coordinate until the spindle actually stops and reverses.
- An **unsynchronized** move back to the original coordinate.

Spindle-synchronized motions wait for spindle index, so multiple passes line up.*G33.1* moves end at the original coordinate.

Todas las palabras del eje son opcionales, pero se debe utilizar al menos una.

G33.1 Example

```
G90 (set absolute mode)
G0 X1.000 Y1.000 Z0.100 (rapid move to starting position)
S100 M3 (turn on the spindle, 100 RPM)
G33.1 Z-0.750 K0.05 (rigid tap a 20 TPI thread 0.750 deep)
M2 (end program)
```

- See G90 & G0 & M2 sections for more information.

It is an error if:

- Todas las palabras del eje se omiten.
- The spindle is not turning when this command is executed
- El movimiento lineal solicitado excede los límites de velocidad de la máquina debido a la velocidad del husillo

11.5.24. G38.*n* Straight Probe

```
G38.n axes
```

- *G38.2* - probe toward workpiece, stop on contact, signal error if failure
- *G38.3* - probe toward workpiece, stop on contact
- *G38.4* - probe away from workpiece, stop on loss of contact, signal error if failure
- *G38.5* - probe away from workpiece, stop on loss of contact

importante
You will not be able to use a probe move until your machine has been set up to provide a probe input signal. The probe input signal must be connected to *motion.probe-input* in a .hal file. G38.*n* uses motion.probe-input to determine when the probe has made (or lost) contact. TRUE for probe contact closed (touching), FALSE for probe contact open.

Program *G38*.n *axes* to perform a straight probe operation. The axis words are optional, except that at least one of them must be used. The axis words together define the destination point that the probe will move towards, starting from the current location. If the probe is not tripped before the destination is reached G38.2 and G38.4 will signal an error.

The tool in the spindle must be a probe or contact a probe switch.

In response to this command, the machine moves the controlled point (which should be at the center of the probe ball) in a straight line at the current feed rate toward the programmed point. In inverse time feed mode, the feed rate is such that the whole motion from the current point to the programmed point would take the specified time. The move stops (within machine acceleration limits) when the programmed point is reached, or when the requested change in the probe input takes place, whichever occurs first.

Cuadro 11.15: Probing G-Codes

Código	Target State	Move orientation	Error Signal
G38.2	Touched	Toward piece	Yes
G38.3	Touched	Toward piece	No
G38.4	Untouched	From piece	Yes
G38.5	Untouched	From piece	No

After successful probing, parameters #5061 to #5069 will be set to the X, Y, Z, A, B, C, U, V, W coordinates of the location of the controlled point at the time the probe changed state (in the current work coordinate system). After unsuccessful probing, they are set to the coordinates of the programmed point. Parameter 5070 is set to 1 if the probe succeeded and 0 if the probe failed. If the probing operation failed, G38.2 and G38.4 will signal an error by posting an message on screen if the selected GUI supports that. And by halting program execution.

Here is an example formula to probe tool height with conversion from a local coordinate system Z offset to machine coordinates which is stored in the tool table. The existing tool height compensation is first cancelled with G49 to avoid including it in the calculation of height, and the new height is loaded from the tool table. The start position must be high enough above the tool height probe to compensate for the use of G49.

G38.2 Example

```
G49
G38.2 Z-100 F100
#<zworkoffset> = [#[5203 + #5220 * 20] + #5213 * #5210]
G10 L1 P#5400 Z#<zworkoffset> (set new tool offset)
G43
```

A comment of the form *(PROBEOPEN filename.txt)* will open *filename.txt* and store the 9-number coordinate consisting of XYZABCUVW of each successful straight probe in it. The file must be closed with *(PROBECLOSE)*. For more information see the Comments section.

An example file *smartprobe.ngc* is included (in the examples directory) to demonstrate using probe moves to log to a file the coordinates of a part. The program *smartprobe.ngc* could be used with *ngcgui* with minimal changes.

It is an error if:

- the current point is the same as the programmed point.
- no axis word is used
- cutter compensation is enabled
- the feed rate is zero
- the probe is already in the target state

11.5.25. G40 Compensation Off

- *G40* - turn cutter compensation off. If tool compensation was on the next move must be a linear move and longer than the tool diameter. It is OK to turn compensation off when it is already off.

G40 Example

```
; current location is X1 after finishing cutter compensated move
G40 (turn compensation off)
G0 X1.6 (linear move longer than current cutter diameter)
M2 (end program)
```

See G0 & M2 sections for more information.

It is an error if:

- A G2/G3 arc move is programmed next after a G40.
- The linear move after turning compensation off is less than the tool diameter.

11.5.26. G41, G42 Cutter Compensation

```
G41 <D-> (left of programmed path)
G42 <D-> (right of programmed path)
```

- *D* - tool number

The D word is optional; if there is no D word the radius of the currently loaded tool will be used (if no tool is loaded and no D word is given, a radius of 0 will be used).

If supplied, the D word is the tool number to use. This would normally be the number of the tool in the spindle (in which case the D word is redundant and need not be supplied), but it may be any valid tool number.

---

nota
G41/G42 D0 is a little special. Its behavior is different on random tool changer machines and nonrandom tool changer machines (see the Tool Change section). On nonrandom tool changer machines, *G41/G42 D0* applies the Tool Length Offset of the tool currently in the spindle, or a TLO of 0 if no tool is in the spindle. On random tool changer machines, *G41/G42 D0* applies the TLO of the tool T0 defined in the tool table file (or causes an error if T0 is not defined in the tool table).

To start cutter compensation to the left of the part profile, use G41. G41 starts cutter compensation to the left of the programmed line as viewed from the positive end of the axis perpendicular to the plane.

To start cutter compensation to the right of the part profile, use G42. G42 starts cutter compensation to the right of the programmed line as viewed from the positive end of the axis perpendicular to the plane.

The lead in move must be at least as long as the tool radius. The lead in move can be a rapid move.

Cutter compensation may be performed if the XY-plane or XZ-plane is active.

User M100-M199 commands are allowed when Cutter Compensation is on.

The behavior of the machining center when cutter compensation is on is described in the Cutter Compensation section along with code examples.

It is an error if:

- The D number is not a valid tool number or 0.
- The YZ plane is active.
- Cutter compensation is commanded to turn on when it is already on.

11.5.27. G41.1, G42.1 Dynamic Cutter Compensation

```
G41.1 D- <L-> (left of programmed path)
G42.1 D- <L-> (right of programmed path)
```

- *D* - cutter diameter
- *L* - tool orientation (see lathe tool orientation)

G41.1 & G42.1 function the same as G41 & G42 with the added scope of being able to program the tool diameter. The L word defaults to 0 if unspecified.

It is an error if:

- The YZ plane is active.
- The L number is not in the range from 0 to 9 inclusive.
- The L number is used when the XZ plane is not active.
- Cutter compensation is commanded to turn on when it is already on.

11.5.28. G43 Tool Length Offset

```
G43 <H->
```

- *H* - tool number (optional)
- *G43* - enables tool length compensation. G43 changes subsequent motions by offsetting the axis coordinates by the length of the offset. G43 does not cause any motion. The next time a compensated axis is moved, that axis's endpoint is the compensated location.

G43 without an H word uses the currently loaded tool from the last *T*n *M6*.

*G43 H*n uses the offset for tool *n*.

The active tool length compensation values are stored in the numbered parameters *5401-5409*.

nota
G43 H0 is a little special. Its behavior is different on random tool changer machines and nonrandom tool changer machines (see the Tool Changers section). On nonrandom tool changer machines, *G43 H0* applies the Tool Length Offset of the tool currently in the spindle, or a TLO of 0 if no tool is in the spindle. On random tool changer machines, *G43 H0* applies the TLO of the tool T0 defined in the tool table file (or causes an error if T0 is not defined in the tool table).

G43 H- Example Line

```
G43 H1 (set tool offsets using the values from tool 1 in the tool table)
```

It is an error if:

- the H number is not an integer, or
- the H number is negative, or
- the H number is not a valid tool number (though note that 0 is a valid tool number on nonrandom tool changer machines, it means ”the tool currently in the spindle”)

11.5.29. G43.1 Dynamic Tool Length Offset

```
G43.1 axes
```

- *G43.1 axes* - change subsequent motions by replacing the current offset(s) of axes. G43.1 does not cause any motion. The next time a compensated axis is moved, that axis’s endpoint is the compensated location.

G43.1 Example

```
G90 (set absolute mode)
T1 M6 G43 (load tool 1 and tool length offsets, Z is at machine 0 and DRO shows Z1.500)
G43.1 Z0.250 (replace current tool offset with 0.250, DRO now shows Z0.250)
M2 (end program)
```

- See G90 & T & M6 sections for more information.

It is an error if:

- motion is commanded on the same line as *G43.1*

nota
G43.1 does not write to the tool table.

11.5.30. G43.2 Apply additional Tool Length Offset

```
G43.2 H- or axes-
```

- *H* - tool number
- *G43.2 H*n - applies an additional simultaneous tool offset to subsequent motions by adding the offset(s) of tool *n*.
- *G43.2 axes* - applies an additional simultaneous tool offset to subsequent motions by adding the value(s) of any axis words.

G43.2 Hn Example

```
G90 (set absolute mode)
T1 M6 (load tool 1)
G43 (or G43 H1 - replace all tool offsets with T1's offset)
G43.2 H10 (also add in T10's tool offset)
M2 (end program)
```

G43.2 axes Example

```
G90 (set absolute mode)
T1 M6 (load tool 1)
G43 (or G43 H1 - replace all tool offsets with T1's offset)
G43.2 X0.01 Z0.02 (also add 0.01 to the x tool offset and 0.02 to the z tool offset)
M2 (end program)
```

You can sum together an arbitrary number of offsets by calling G43.2 more times. There are no built-in assumptions about which numbers are geometry offsets and which are wear offsets, or that you should have only one of each.

Like the other G43 commands, G43.2 does not cause any motion. The next time a compensated axis is moved, that axis's endpoint is the compensated location.

It is an error if:

- *H* no está especificado y no se especifican offsets de eje.
- *H* está especificado y el número de herramienta dado no existe en la tabla de herramientas.
- Se especifica *H* y también se especifican los ejes.

---

nota
G43.2 does not write to the tool table.

11.5.31. G49 Cancel Tool Length Compensation

- *G49* - cancels tool length compensation

It is OK to program using the same offset already in use. It is also OK to program using no tool length offset if none is currently being used.

11.5.32. G52 Local Coordinate System Offset

```
G52 axes
```

G52 is used in a part program as a temporary ”local coordinate system offset” within the workpiece coordinate system. For more information about G92 and G52 and how they interact see Local and Global Offsets.

11.5.33. G53 Move in Machine Coordinates

```
G53 axes
```

To move in the machine coordinate system, program *G53* on the same line as a linear move. *G53* is not modal and must be programmed on each line. *G0* or *G1* does not have to be programmed on the same line if one is currently active.

For example *G53 G0 X0 Y0 Z0* will move the axes to the home position even if the currently selected coordinate system has offsets in effect.

G53 Example

```
G53 G0 X0 Y0 Z0 (rapid linear move to the machine origin)
G53 X2 (rapid linear move to absolute coordinate X2)
```

See G0 section for more information.

It is an error if:

- G53 is used without G0 or G1 being active,
- or G53 is used while cutter compensation is on.

11.5.34. G54-G59.3 Select Coordinate System

- *G54* - seleccione el sistema de coordenadas 1
- *G55* - selecciona el sistema de coordenadas 2
- *G56* - selecciona el sistema de coordenadas 3
- *G57* - selecciona el sistema de coordenadas 4
- *G58* - selecciona el sistema de coordenadas 5
- *G59* - selecciona el sistema de coordenadas 6
- *G59.1* - selecciona el sistema de coordenadas 7
- *G59.2* - selecciona el sistema de coordenadas 8
- *G59.3* - selecciona el sistema de coordenadas 9

The coordinate systems store the axis values and the XY rotation angle around the Z axis in the parameters shown in the following table.

Cuadro 11.16: Coordinate System Parameters

Select	CS	X	Y	Z	A	B	C	U	V	W	R
G54	1	5221	5222	5223	5224	5225	5226	5227	5228	5229	5230
G55	2	5241	5242	5243	5244	5245	5246	5247	5248	5249	5250
G56	3	5261	5262	5263	5264	5265	5266	5267	5268	5269	5270
G57	4	5281	5282	5283	5284	5285	5286	5287	5288	5289	5290
G58	5	5301	5302	5303	5304	5305	5306	5307	5308	5309	5310
G59	6	5321	5322	5323	5324	5325	5326	5327	5328	5329	5330
G59.1	7	5341	5342	5343	5344	5345	5346	5347	5348	5349	5350
G59.2	8	5361	5362	5363	5364	5365	5366	5367	5368	5369	5370
G59.3	9	5381	5382	5383	5384	5385	5386	5387	5388	5389	5390

It is an error if:

- selecting a coordinate system is used while cutter compensation is on.

Consulte la sección Sistema de coordenadas para obtener una descripción general de los sistemas de coordenadas.

11.5.35. G61 Exact Path Mode

- *G61* - Exact path mode, movement exactly as programmed. Moves will slow or stop as needed to reach every programmed point. If two sequential moves are exactly co-linear movement will not stop.

11.5.36. G61.1 Exact Stop Mode

- *G61.1* - Exact stop mode, movement will stop at the end of each programmed segment.

11.5.37. G64 Path Blending

```
G64 <P- <Q->>
```

- *P* - motion blending tolerance
- *Q* - naive cam tolerance
- *G64* - best possible speed. Without P (Or a default value in RS274NGC) means to keep the best speed possible, no matter how far away from the programmed point you end up.
- *G64 P-* - Mezcla entre la mejor velocidad y tolerancia de desviación
- *G64 P- <Q- >* blending with tolerance. It is a way to fine tune your system for best compromise between speed and accuracy. The P- tolerance means that the actual path will be no more than P- away from the programmed endpoint. The velocity will be reduced if needed to maintain the path. If you set Q to a non-zero value it turns on the *Naive CAM Detector*: when there are a series of linear XYZ feed moves at the same feed rate that are less than Q- away from being collinear, they are collapsed into a single linear move. On G2/G3 moves in the G17 (XY) plane when the maximum deviation of an arc from a straight line is less than the G64 P- tolerance the arc is broken into two lines (from start of arc to midpoint, and from midpoint to end). those lines are then subject to the

naive cam algorithm for lines. Thus, line-arc, arc-arc, and arc-line cases as well as line-line benefit from the *Naive CAM Detector*. This improves contouring performance by simplifying the path. It is OK to program for the mode that is already active. See also the Trajectory Control section for more information on these modes. If Q is not specified then it will have the same behavior as before and use the value of P-. Set Q to zero to disable the *Naive CAM Detector*.

It is a good idea to include a path control specification in the preamble of each G-code file.

G64 P- Q- Example Line

```
G64 P0.015 Q0.015 (set path following to be within 0.015 of the actual path)
```

The P- and Q- values are chosen small. Usually smaller than the machine accuracy or the accuracy of commonly manufactured parts. Below are examples with extreme P- and Q- values to understand the G64 function.

G64 Without values

```
G64 (set without P- and Q- values)
```

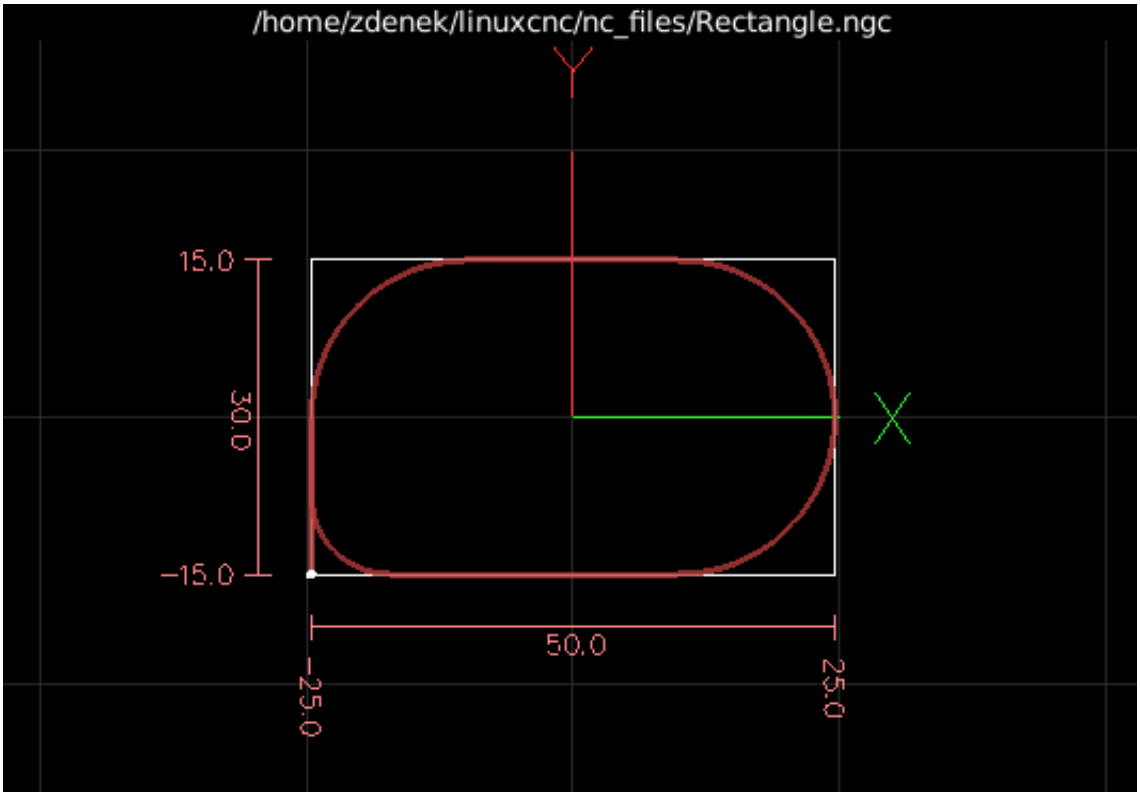


Figura 11.14: G64 Rectangle

G64 With big Q- value

```
G64 P0.015 Q6
```

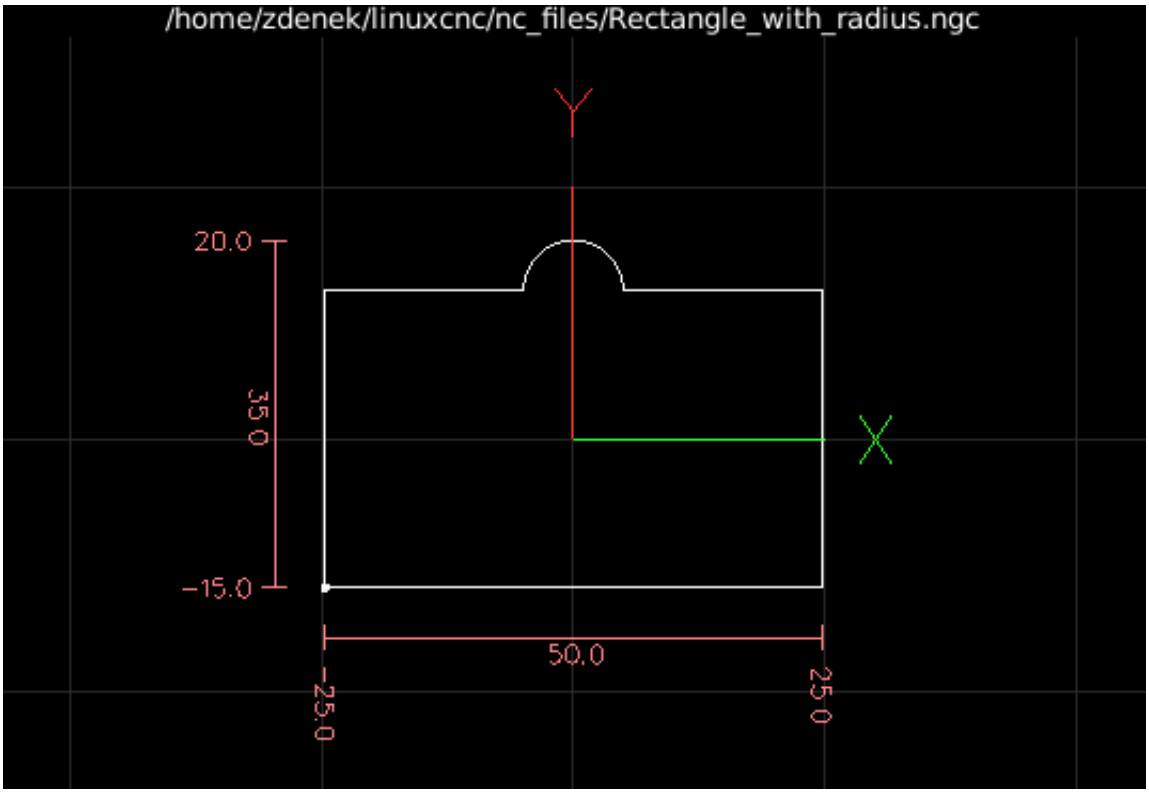


Figura 11.15: G64 Rectangle with radius before milling

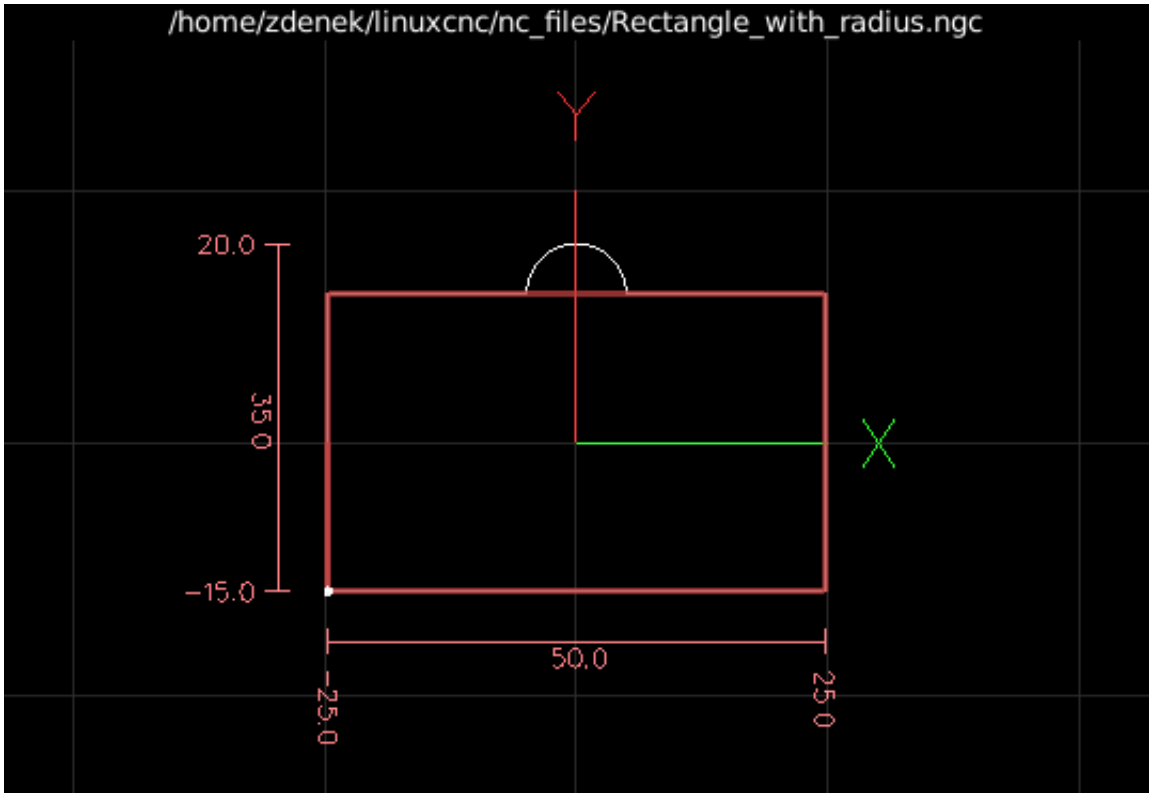


Figura 11.16: G64 Rectangle with radius after milling

```
G64 P0.015 Q2
```

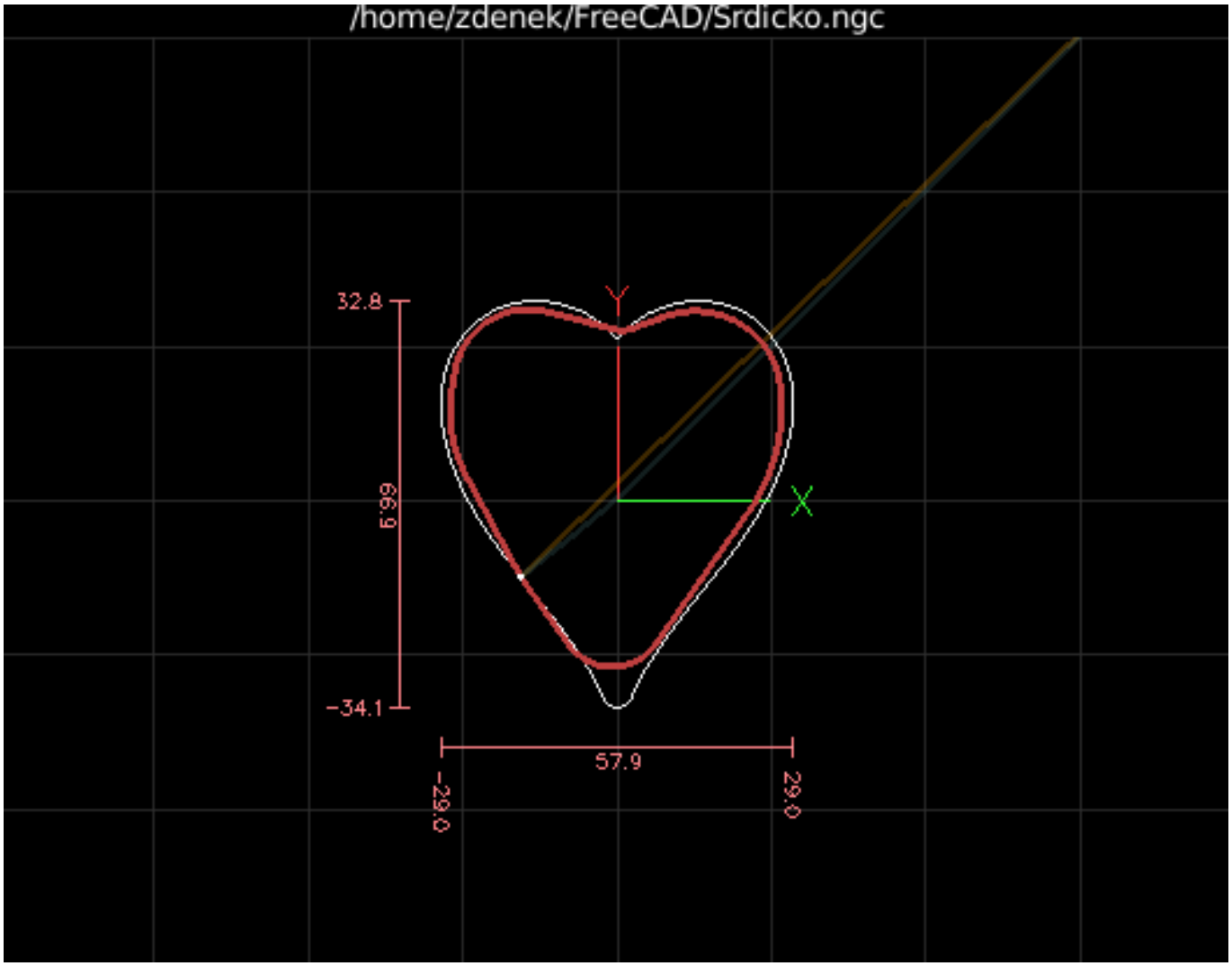


Figura 11.17: G64 Heart

11.5.38. G70 Lathe finishing cycle

```
G70 Q- <X-> <Z-> <D-> <E-> <P->
```

- *Q* - The subroutine number.
- *X* - The starting X position, defaults to the initial position.
- *Z* - The starting Z position, defaults to the initial position.
- *D* - The starting distance of the profile, defaults to 0.
- *E* - The ending distance of the profile, defaults to 0.
- *P* - The number of passes to use, defaults to 1.

The *G70* cycle is intended to be used after the shape of the profile given in the subroutine with number Q has been cut with G71 or G72.

- Movimiento preliminar.
  - If Z or X are used a rapid move to that position is done. This position is also used between each finishing pass.
  - Then a rapid move to the start of the profile is executed.
  - The path given in Q- is followed using the G1 and Sección 11.5.5 commands.
  - If a next pass is required there is another rapid to the intermediate location, before a rapid is done to the start of the profile.
  - After the final pass, the tool is left at the end of the profile including E-.
- Multiple passes. The distance between the pass and the final profile is (pass-1)*(D-E)/P+E. Where pass the pass number and D,E and P are the D/E/P numbers.
- The distance is computed using the starting position of the cycle, with a positive distance towards this point.
- Fillet and chamfers in the profile. It is possible to add fillets or chamfers in the profile, see Sección 11.5.39 for more details.

It is an error if:

- There is no subroutine defined with the number given in Q.
- The path given in the profile is not monotonic in Z or X.
- Sección 11.5.18 has not been used to select the ZX plane.

11.5.39. G71 G72 Lathe roughing cycles

nota
Los ciclos G71 y G72 son actualmente un tanto frágiles. Por ejemplo vea el issue #2939.

```
G71   Q- <X-> <Z-> <D-> <I-> <R->
G71.1 Q- <X-> <Z-> <D-> <I-> <R->
G71.2 Q- <X-> <Z-> <D-> <I-> <R->
G72   Q- <X-> <Z-> <D-> <I-> <R->
G72.1 Q- <X-> <Z-> <D-> <I-> <R->
G72.2 Q- <X-> <Z-> <D-> <I-> <R->
```

- *Q* - The subroutine number.
- *X* - The starting X position, defaults to the initial position.
- *Z* - The starting Z position, defaults to the initial position.
- *D* - The remaining distance to the profile, defaults to 0.
- *I* - The cutting increment, defaults to 1.
- *R* - The retracting distance, defaults to 0.5.

The G71/G72 cycle is intended to rough cut a profile on a lathe. The G71 cycles remove layers of the material while traversing in the Z direction. The G72 cycles remove material while traversing the X axis, the so called facing cycle. The direction of travel is the same as in the path given in the subroutine. For the G71 cycle the Z coordinate must be monotonically changing, for the G72 this is required for the X axis.

The profile is given in a subroutine with number Q-. This subroutine may contain G0, G1, G2 and G3 motion commands. All other commands are ignored, including feed and speed settings. The Sección 11.5.3 commands are interpreted as G1 commands. Each motion command may also include an optional A- or C- number. If the number A- is added a fillet with the radius given by A will be inserted at the endpoint of that motion, if this radius is too large the algorithm will fail with a non-monotonic path error. It is also possible to use the C- number, which allows a chamfer to be inserted. This chamfer has the same endpoints as a fillet of the same dimension would have but a straight line is inserted instead of an arc.

When in absolute mode the U (for X) and W (for Z) can be used as incremental displacements.

The G7x.1 cycles do not cut pockets. The G7x.2 cycles only cut after the first pocket and continue where G7x.1 stopped. It is advisible to leave some additional material to cut before the G7x.2 cycle, so if G7x.1 used a D1.0 the G7x.2 can use D0.5 and 0.5mm will be removed while moving from one pocket to the next.

The normal G7x cycles cut the entire profile in one cycle.

1. Movimiento preliminar.
   - If Z or X are used a rapid move to that position is done.
   - After the profile has been cut, the tool stops at the end of the profile, including the distance specified in D.
2. The D number is used to keep a distance from the final profile, to allow material to remain for finishing.

It is an error if:

- There is no subroutine defined with the number given in Q.
- The path given in the profile is not monotonic in Z or X.
- Sección 11.5.18 has not been used to select the ZX plane.
- Sección 11.5.26 is active.

11.5.40. G73 Drilling Cycle with Chip Breaking

```
G73 X- Y- Z- R- Q- D- <L->
```

- *R* - retract position along the Z axis.
- *Q* - delta increment along the Z axis.
- *L* - repeat

The *G73* cycle is drilling or milling with chip breaking. This cycle takes a Q number which represents a *delta* increment along the Z axis. Peck clearance can be specified by optional D number.

- Movimiento preliminar.

  - If the current Z position is below the R position, The Z axis does a rapid move to the R position.
  - Move to the X Y coordinates
- Move the Z-axis only at the current feed rate downward by delta or to the Z position, whichever is less deep.
- Rapid up either the D value, the G73_PECK_CLEARANCE specified in the INI file or the default of .010” / 0.254 mm.
- Repeat steps 2 and 3 until the Z position is reached at step 2.
- The Z axis does a rapid move to the R position.

It is an error if:

- the Q number is negative or zero.
- the R number is not specified

11.5.41. G74 Left-hand Tapping Cycle with Dwell

```
G74 (X- Y- Z-) or (U- V- W-) R- L- P- $- F-
```

- *R-* - Retract position along the Z axis.
- *L-* - Used in incremental mode; number of times to repeat the cycle. See G81 for examples.
- *P-* - Dwell time (seconds).
- *$-* - Selected spindle.
- *F-* - Feed rate (spindle speed multiplied by distance traveled per revolution (thread pitch)).



aviso
G74 does not use synchronized motion.

The *G74* cycle is intended for tapping with floating chuck and dwell at the bottom of the hole.

1. Movimiento preliminar, como se describe en la sección Movimientos preliminares e intermedios.
2. Deshabilita los ajustes de avance y velocidad.
3. Mueve el eje Z a la velocidad de avance actual a la posición Z.
4. Stop the selected spindle (chosen by the $ parameter)
5. Start spindle rotation clockwise.
6. Espera P segundos.
7. Move the Z-axis at the current feed rate to clear Z
8. Restaurar ajustes de alimentación y velocidad al estado anterior

The length of the dwell is specified by a *P-* word in the G74 block. The feed rate *F-* is spindle speed multiplied by distance per revolution (thread pitch). In example S100 with 1.25MM per revolution thread pitch gives a feed of F125.

11.5.42. G76 Threading Cycle

```
G76 P- Z- I- J- R- K- Q- H- E- L- $-
```

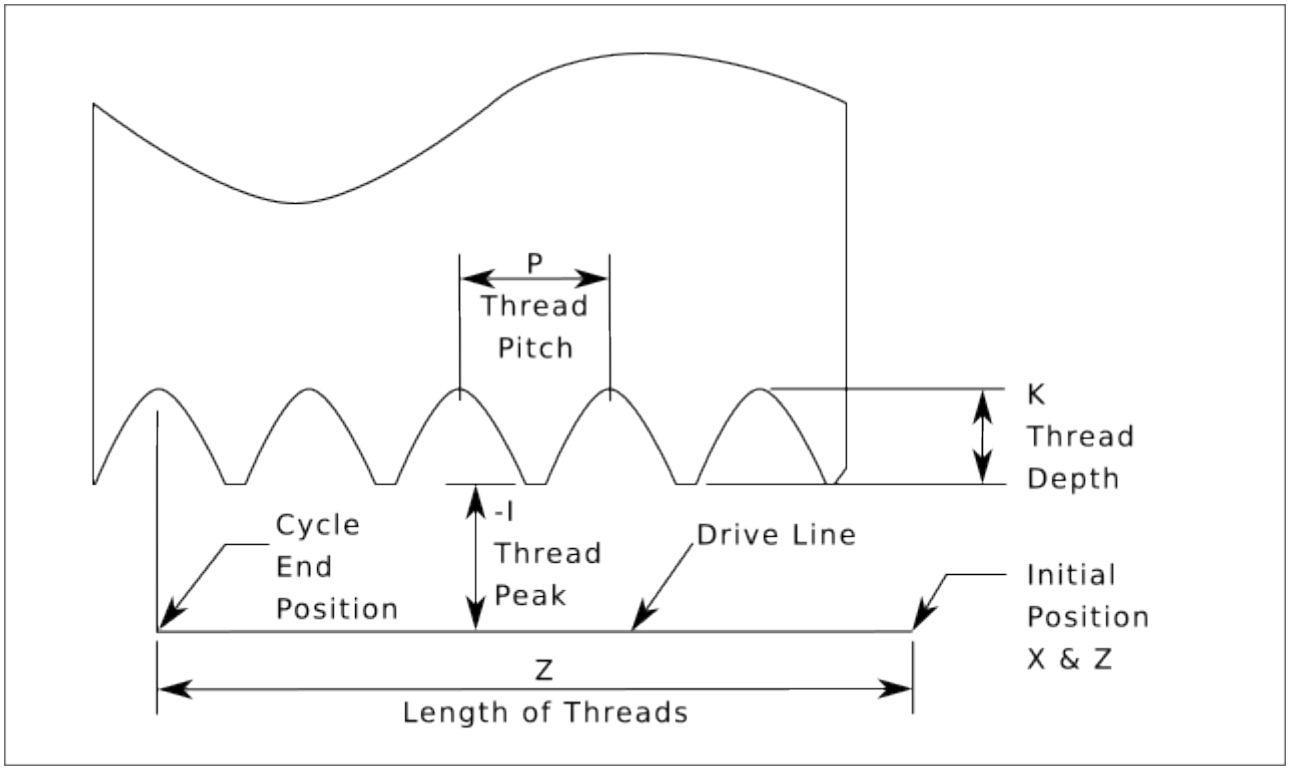


Figura 11.18: G76 Threading

- *Drive Line* - A line through the initial X position parallel to the Z.
- *P-* - The *thread pitch* in distance per revolution.
- *Z-* - La posición final de los hilos. Al final del ciclo la herramienta estará en esta posición Z.

---

nota
When G7 *Lathe Diameter Mode* is in force the values for *I*, *J* and *K* are diameter measurements. When G8 *Lathe Radius Mode* is in force the values for *I*, *J* and *K* are radius measurements.

- *I-* - The *thread peak* offset from the *drive line*. Negative *I* values are external threads, and positive *I* values are internal threads. Generally the material has been turned to this size before the *G76* cycle.
- *J-* - A positive value specifying the *initial cut depth*. The first threading cut will be *J* beyond the *thread peak* position.
- *K-* - A positive value specifying the *full thread depth*. The final threading cut will be *K* beyond the *thread peak* position.

Optional settings

- *$* - - El número de husillo con el que se sincronizará el movimiento (predeterminado 0). Por ejemplo, si está programado $1, entonces el movimiento comenzará en el reinicio de `spindle.1.index-enable` y procede en sincronía con el valor de `spindle.1.revs`.
- *R*- - La *degresión de profundidad*. *R1.0* selecciona profundidad constante en sucesivos pases de roscado. *R2.0* selecciona área constante. Valores entre 1.0 y 2.0 seleccionan profundidad decreciente pero área creciente. Los valores superiores a 2.0 seleccionan área decreciente. Tenga en cuenta que los valores de degresión innecesariamente altos causarán un gran número de pases. (degresión = un descenso por etapas o pasos.)



aviso
Unnecessarily high degression values will produce an unnecessarily high number of passes. (degressing = dive in stages)

- *Q*- - The *compound slide angle* is the angle (in degrees) describing to what extent successive passes should be offset along the drive line. This is used to cause one side of the tool to remove more material than the other. A positive *Q* value causes the leading edge of the tool to cut more heavily. Typical values are 29, 29.5 or 30.
- *H*- - The number of *spring passes*. Spring passes are additional passes at full thread depth. If no additional passes are desired, program *H0*.

Thread entries and exits can be programmed tapered with the *E* and *L* values.

- *E*- - Especifica la distancia a lo largo de la línea guia utilizada para conicidad de la entrada. El ángulo del cono será de modo que el último pase se estrecha hacia la cresta del hilo sobre la distancia especificada con E. *E0.2* dará un cono para las primeras/últimas 0.2 unidades de longitud a lo largo del hilo. Para un programa de conicidad de 45 grados, E igual que K.
- *L*- - Specifies which ends of the thread get the taper. Program *L0* for no taper (the default), *L1* for entry taper, *L2* for exit taper, or *L3* for both entry and exit tapers. Entry tapers will pause at the drive line to synchronize with the index pulse then move at the feed rate in to the beginning of the taper. No entry taper and the tool will rapid to the cut depth then synchronize and begin the cut.

The tool is moved to the initial X and Z positions prior to issuing the G76. The X position is the *drive line* and the Z position is the start of the threads.

The tool will pause briefly for synchronization before each threading pass, so a relief groove will be required at the entry unless the beginning of the thread is past the end of the material or an entry taper is used.

Unless using an exit taper, the exit move is not synchronized to the spindle speed and will be a rapid move. With a slow spindle, the exit move might take only a small fraction of a revolution. If the spindle speed is increased after several passes are complete, subsequent exit moves will require a larger portion of a revolution, resulting in a very heavy cut during the exit move. This can be avoided by providing a relief groove at the exit, or by not changing the spindle speed while threading.

The final position of the tool will be at the end of the *drive line*. A safe Z move will be needed with an internal thread to remove the tool from the hole.

It is an error if:

- El plano activo no es el plano ZX.
- Se especifican otras palabras de eje, como X- o Y-.
- The *R*- degression value is less than 1.0.

- No se especifican todas las palabras requeridas.
- *P-*, *J-*, *K-* o *H-* son negativos.
- *E-* es mayor que la mitad de la longitud de la línea guia.

HAL Connections The pins *spindle.N.at-speed* and the *encoder.*n.*phase-Z* for the spindle must be connected in your HAL file before G76 will work. See the spindle pins in the Motion section for more information.

Technical Info The G76 canned cycle is based on the G33 Spindle Synchronized Motion. For more information see the G33 Technical Info.

The sample program *g76.ngc* shows the use of the G76 canned cycle, and can be previewed and executed on any machine using the *sim/lathe.ini* configuration.

G76 Example Code

```
G0 Z-0.5 X0.2
G76 P0.05 Z-1 I-.075 J0.008 K0.045 Q29.5 L2 E0.045
```

In the figure the tool is in the final position after the G76 cycle is completed. You can see the entry path on the right from the Q29.5 and the exit path on the left from the L2 E0.045. The white lines are the cutting moves.

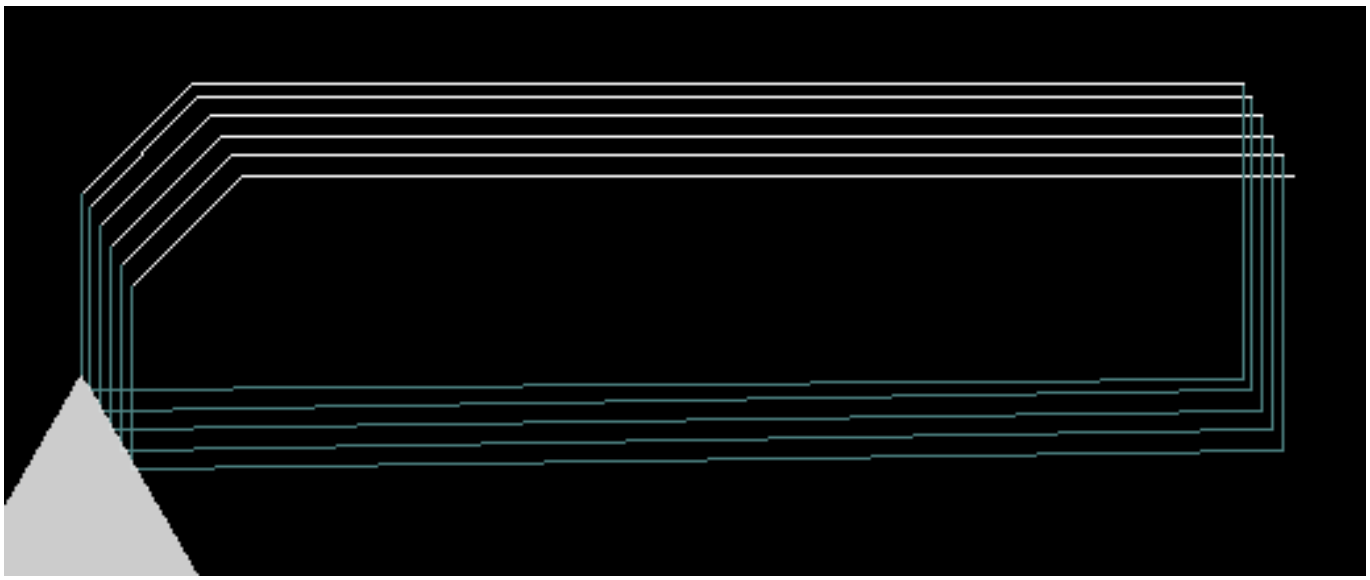


Figura 11.19: G76 Example

11.5.43. G80-G89 Canned Cycles

The canned cycles *G81* through *G89* and the canned cycle stop *G80* are described in this section.

Todos los ciclos fijos se realizan con respecto al plano seleccionado actualmente. Se puede seleccionar cualquiera de los nueve planos. A lo largo de esta sección, la mayoría de las descripciones asumen que el plano XY ha sido seleccionado. El comportamiento es análogo si se selecciona otro plano y se deben usar las palabras correctas. Por ejemplo, en el plano *G17.1*, la acción del ciclo fijo es a lo largo de W, y las ubicaciones o incrementos se dan con U y V. En este caso, sustituya U, V, W por X, Y, Z en las instrucciones a continuación.

Rotary axis words are not allowed in canned cycles. When the active plane is one of the XYZ family, the UVW axis words are not allowed. Likewise, when the active plane is one of the UVW family, the XYZ axis words are not allowed.

11.5.43.1. Common Words

All canned cycles use X, Y, Z, or U, V, W groups depending on the plane selected and R words. The R (usually meaning retract) position is along the axis perpendicular to the currently selected plane (Z-axis for XY-plane, etc.) Some canned cycles use additional arguments.

11.5.43.2. Sticky Words

Para los ciclos fijos, llamaremos a un número *sticky* si, cuando el mismo ciclo se utiliza en varias líneas de código en fila, el número debe ser usado la primera vez, pero es opcional en el resto de las líneas. Los números sticky mantienen su valor en el resto de las líneas si no son programados explícitamente con valor diferente. El número R siempre es sticky.

En el modo de distancia incremental, los números X, Y y R se tratan como incrementos desde la posición actual y Z como un incremento desde la posición del eje Z antes de que tenga lugar el movimiento que involucra a Z. En modo de distancia absoluta, los números X, Y, R y Z son posiciones absolutas en el sistema de coordenadas actual.

11.5.43.3. Repeat Cycle

The L number is optional and represents the number of repeats. L=0 is not allowed. If the repeat feature is used, it is normally used in incremental distance mode, so that the same sequence of motions is repeated in several equally spaced places along a straight line. When L- is greater than 1 in incremental mode with the XY-plane selected, the X and Y positions are determined by adding the given X and Y numbers either to the current X and Y positions (on the first go-around) or to the X and Y positions at the end of the previous go-around (on the repetitions). Thus, if you program *L10* , you will get 10 cycles. The first cycle will be distance X,Y from the original location. The R and Z positions do not change during the repeats. The L number is not sticky. In absolute distance mode, L>1 means *do the same cycle in the same place several times*, Omitting the L word is equivalent to specifying L=1.

11.5.43.4. Retract Mode

The height of the retract move at the end of each repeat (called *clear Z* in the descriptions below) is determined by the setting of the retract mode, either to the original Z position (if that is above the R position and the retract mode is *G98*, OLD_Z), or otherwise to the R position. See the G98 G99 section.

11.5.43.5. Canned Cycle Errors

It is an error if:

- axis words are all missing during a canned cycle,
- axis words from different groups (XYZ) (UVW) are used together,
- a P number is required and a negative P number is used,
- an L number is used that does not evaluate to a positive integer,
- rotary axis motion is used during a canned cycle,
- inverse time feed rate is active during a canned cycle,
- or cutter compensation is active during a canned cycle.

If the XY plane is active, the Z number is sticky, and it is an error if:

- the Z number is missing and the same canned cycle was not already active,
- o el número R es menor que el número Z.

If other planes are active, the error conditions are analogous to the XY conditions above.

11.5.43.6. Preliminary and In-Between Motion

Preliminary motion is a set of motions that is common to all of the milling canned cycles. If the current Z position is below the R position, the Z axis does a rapid move to the R position. This happens only once, regardless of the value of L.

Además, al comienzo del primer ciclo y en cada repetición, se realizan uno o dos de los movimientos siguientes:

- Un movimiento rápido paralelo al plano XY para la posición XY dada.
- The Z-axis make a rapid move to the R position, if it is not already at the R position.

If another plane is active, the preliminary and in-between motions are analogous.

11.5.43.7. Why use a canned cycle?

There are at least two reasons for using canned cycles. The first is the economy of code. A single bore would take several lines of code to execute.

The G81 Example 1 demonstrates how a canned cycle could be used to produce 8 holes with ten lines of G-code within the canned cycle mode. The program below will produce the same set of 8 holes using five lines for the canned cycle. It does not follow exactly the same path nor does it drill in the same order as the earlier example. But the program writing economy of a good canned cycle should be obvious.

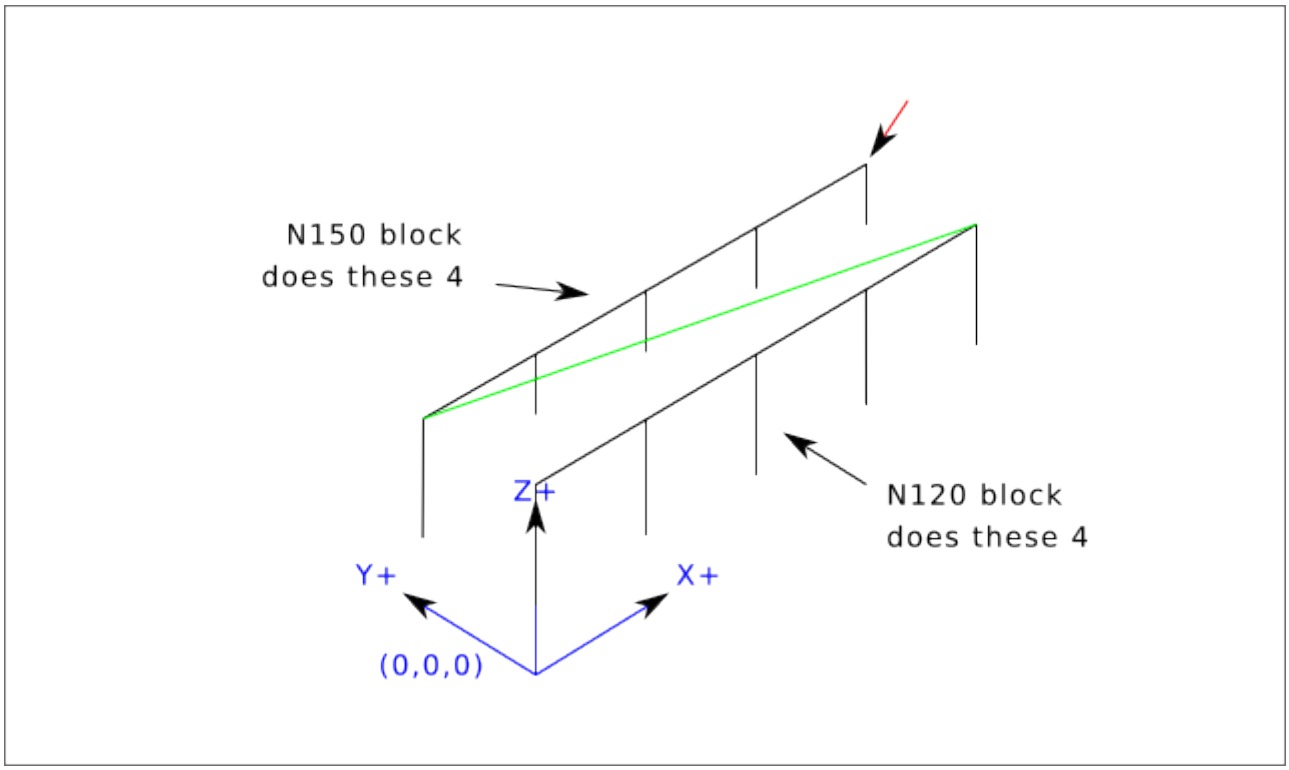
---

nota
Los números de línea no son necesarios, pero ayudan a aclarar estos ejemplos.

Eight Holes

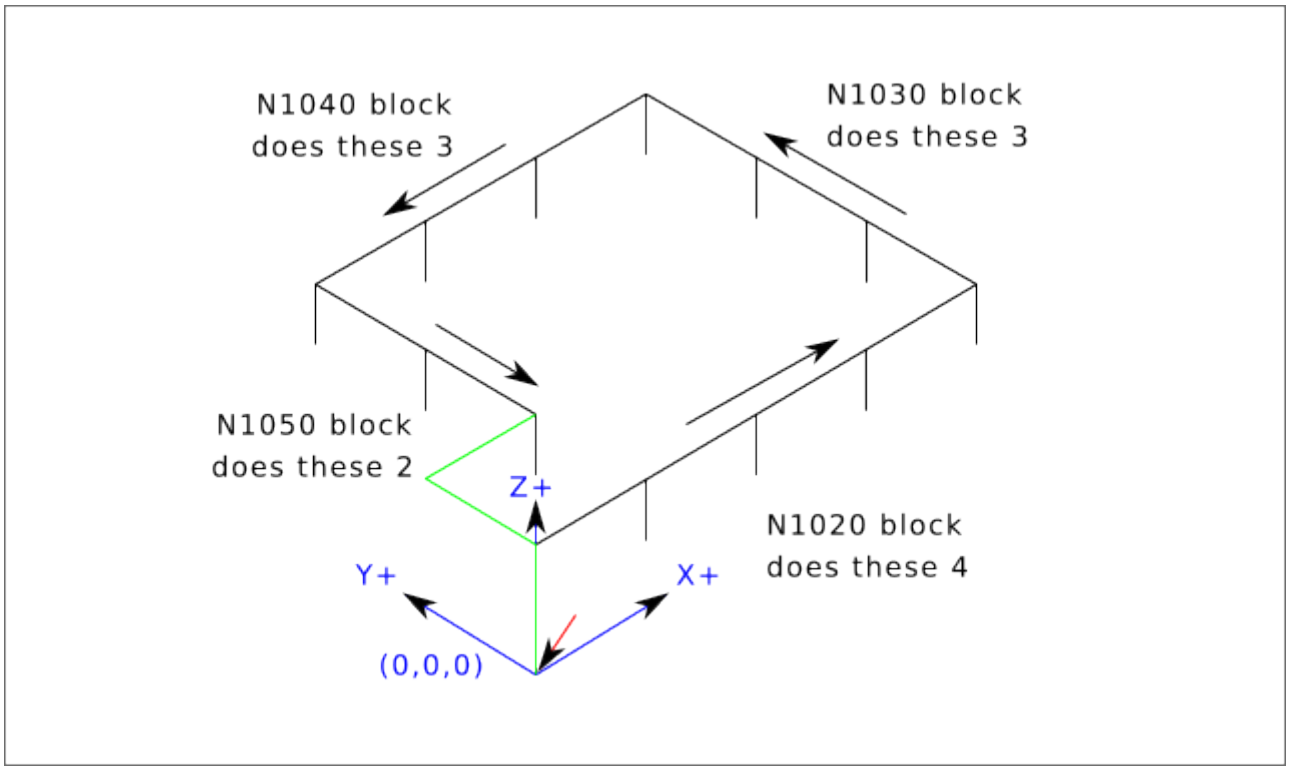
```
N100 G90 G0 X0 Y0 Z0 (move coordinate home)
N110 G1 F10 X0 G4 P0.1
N120 G91 G81 X1 Y0 Z-1 R1 L4(canned drill cycle)
N130 G90 G0 X0 Y1
N140 Z0
N150 G91 G81 X1 Y0 Z-0.5 R1 L4(canned drill cycle)
N160 G80 (turn off canned cycle)
N170 M2 (program end)
```

El G98 en la segunda línea de arriba significa que el movimiento de retorno será al valor de Z en la primera línea, ya que es más alto que el valor R especificado.



Twelve Holes in a Square This example demonstrates the use of the L word to repeat a set of incremental drill cycles for successive blocks of code within the same G81 motion mode. Here we produce 12 holes using five lines of code in the canned motion mode.

```
N1000 G90 G0 X0 Y0 Z0 (move coordinate home)
N1010 G1 F50 X0 G4 P0.1
N1020 G91 G81 X1 Y0 Z-0.5 R1 L4 (canned drill cycle)
N1030 X0 Y1 R0 L3 (repeat)
N1040 X-1 Y0 L3 (repeat)
N1050 X0 Y-1 L2 (repeat)
N1060 G80 (turn off canned cycle)
N1070 G90 G0 X0 (rapid move home)
N1080 Y0
N1090 Z0
N1100 M2 (program end)
```



The second reason to use a canned cycle is that they all produce preliminary moves and returns that you can anticipate and control regardless of the start point of the canned cycle.

11.5.44. G80 Cancel Canned Cycle

- *G80* - cancel canned cycle modal motion. *G80* is part of modal group 1, so programming any other G-code from modal group 1 will also cancel the canned cycle.

It is an error if:

- Axis words are programmed when G80 is active.

G80 Example

```
G90 G81 X1 Y1 Z1.5 R2.8 (absolute distance canned cycle)
G80 (turn off canned cycle motion)
G0 X0 Y0 Z0 (rapid move to coordinate home)
```

The following code produces the same final position and machine state as the previous code.

G0 Example

```
G90 G81 X1 Y1 Z1.5 R2.8 (absolute distance canned cycle)
G0 X0 Y0 Z0 (rapid move to coordinate home)
```

The advantage of the first set is that, the G80 line clearly turns off the G81 canned cycle. With the first set of blocks, the programmer must turn motion back on with G0, as is done in the next line, or any other motion mode G word.

If a canned cycle is not turned off with G80 or another motion word, the canned cycle will attempt to repeat itself using the next block of code that contains an X, Y, or Z word. The following file drills (G81) a set of eight holes as shown in the following caption.

G80 Example 1

```
N100 G90 G0 X0 Y0 Z0 (coordinate home)
N110 G1 X0 G4 P0.1
N120 G81 X1 Y0 Z0 R1 (canned drill cycle)
N130 X2
N140 X3
N150 X4
N160 Y1 Z0.5
N170 X3
N180 X2
N190 X1
N200 G80 (turn off canned cycle)
N210 G0 X0 (rapid move home)
N220 Y0
N230 Z0
N240 M2 (program end)
```

nota
Notice the Z position change after the first four holes. Also, this is one of the few places where line numbers have some value, being able to point a reader to a specific line of code.

The use of G80 in line N200 is optional because the G0 on the next line will turn off the G81 cycle. But using the G80 as shown in Example 1, will provide for easier to read canned cycle. Without it, it is not so obvious that all of the blocks between N120 and N200 belong to the canned cycle.

11.5.45. G81 Drilling Cycle

```
G81 (X- Y- Z-) or (U- V- W-) R- L-
```

The *G81* cycle is intended for drilling.

The cycle functions as follows:

- Movimiento preliminar, como se describe en la sección Movimientos preliminares e intermedios.
- Mover el eje Z a la valocidad de avance actual a la posición Z.
- The Z-axis does a rapid move to clear Z.

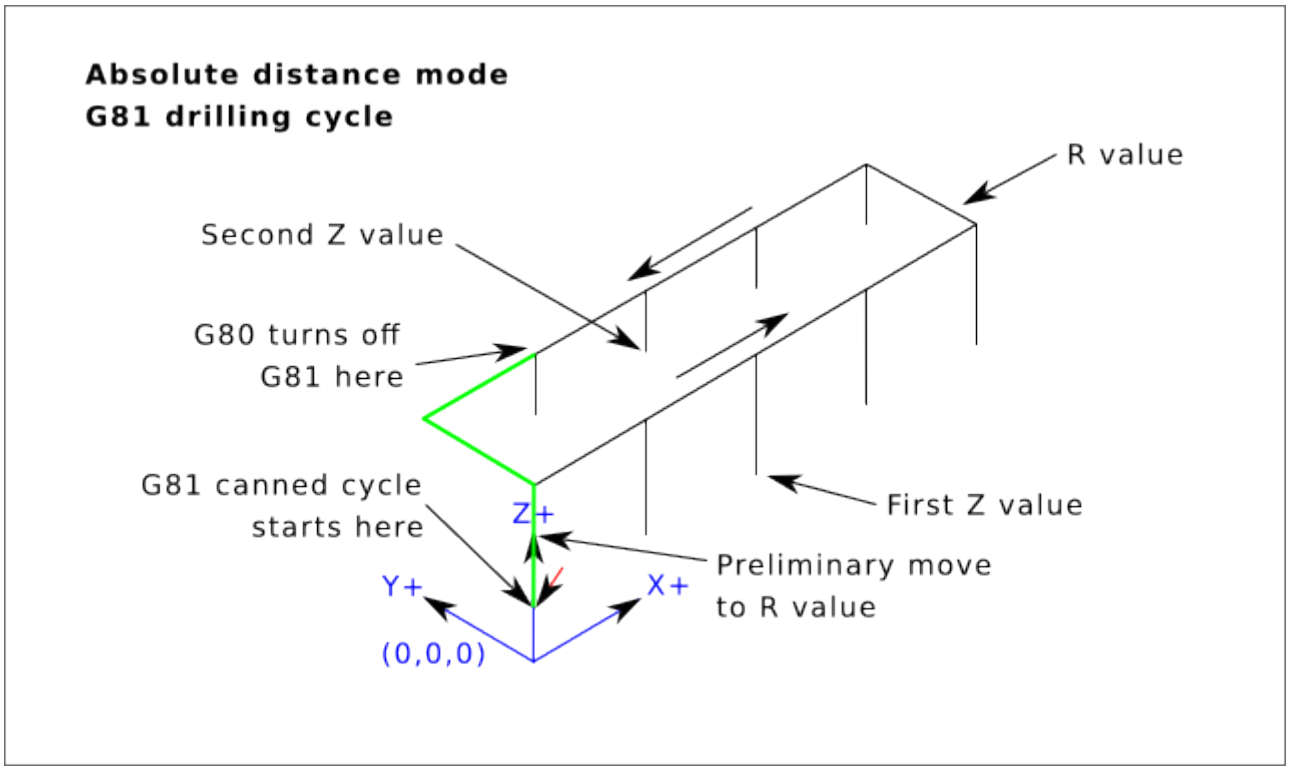


Figura 11.20: G81 Cycle

Example 1 - Absolute Position G81

```
G90 G98 G81 X4 Y5 Z1.5 R2.8
```

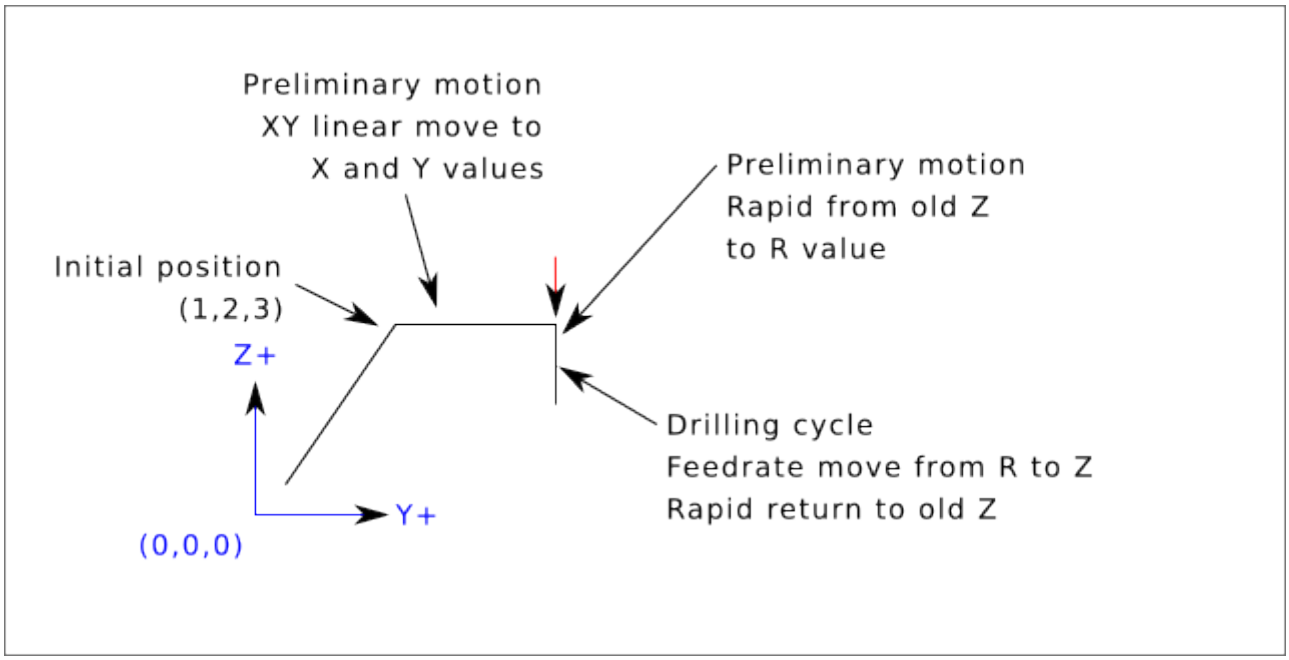
Supongamos que la posición actual es (X1, Y2, Z3) y que la línea anterior a NC es interpretada.

This calls for absolute distance mode (G90) and OLD_Z retract mode (G98) and calls for the G81 drilling cycle to be performed once.

- The X value and X position are 4.
- The Y value and Y position are 5.
- The Z value and Z position are 1.5.
- The R value and clear Z are 2.8. OLD_Z is 3.

The following moves take place:

- A rapid move parallel to the XY plane to (X4, Y5)
- A rapid move move parallel to the Z-axis to (Z2.8).
- Move parallel to the Z-axis at the feed rate to (Z1.5)
- A rapid move parallel to the Z-axis to (Z3)



Example 2 - Relative Position G81

```
G91 G98 G81 X4 Y5 Z-0.6 R1.8 L3
```

Supongamos que la posición actual es (X1, Y2, Z3) y que la línea anterior a NC es interpretada.

This calls for incremental distance mode (G91) and OLD_Z retract mode (G98). It also calls for the G81 drilling cycle to be repeated three times. The X value is 4, the Y value is 5, the Z value is -0.6 and the R value is 1.8. The initial X position is 5 (=1+4), the initial Y position is 7 (=2+5), the clear Z position is 4.8 (=1.8+3), and the Z position is 4.2 (=4.8-0.6). OLD_Z is 3.

The first preliminary move is a maximum rapid move along the Z axis to (X1,Y2,Z4.8), since OLD_Z < clear Z.

La primera repetición consiste en 3 movimientos.

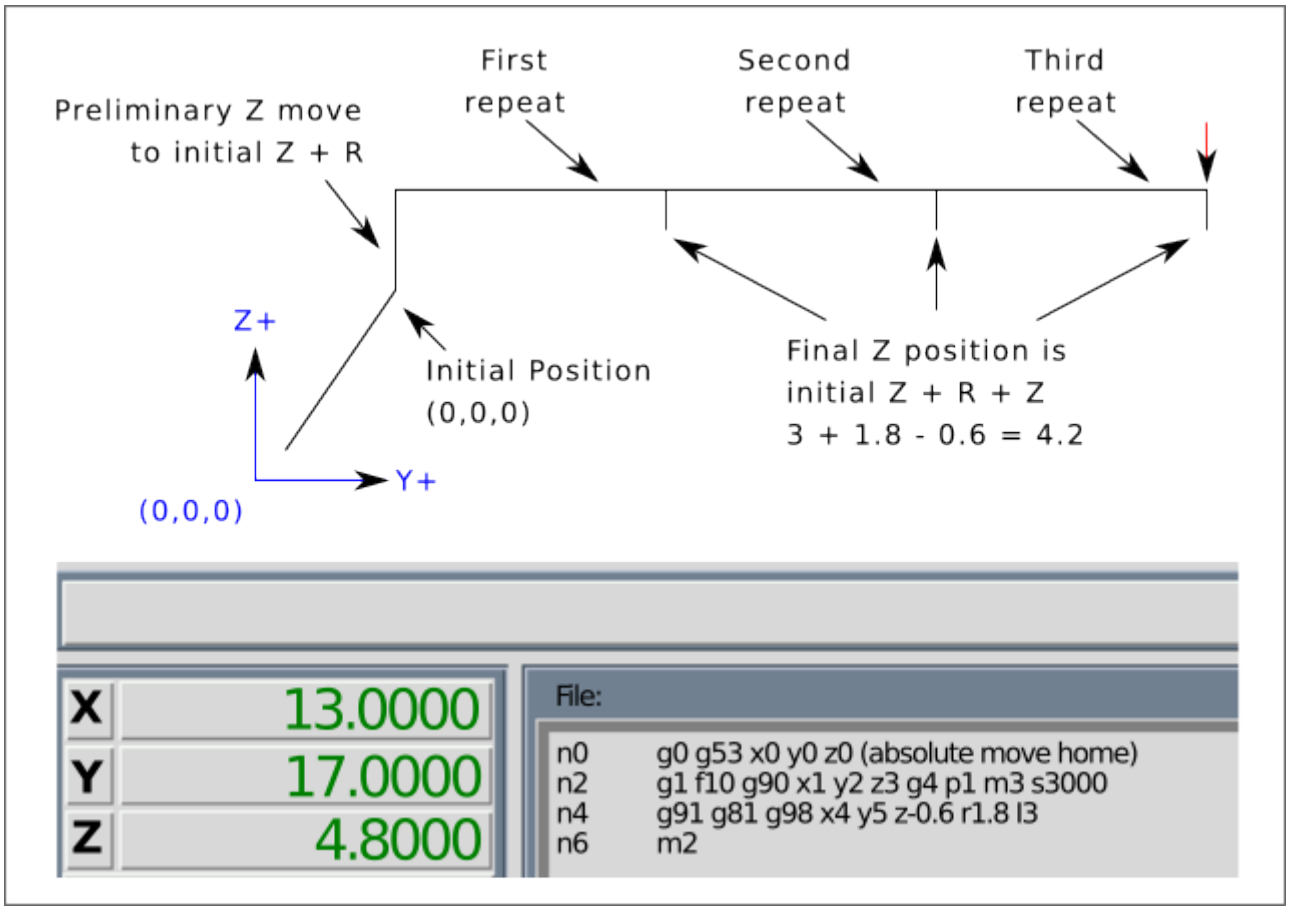
- A rapid move parallel to the XY-plane to (X5, Y7)
- Move parallel to the Z-axis at the feed rate to (Z4.2)
- A rapid move parallel to the Z-axis to (X5, Y7, Z4.8)

The second repeat consists of 3 moves. The X position is reset to 9 (=5+4) and the Y position to 12 (=7+5).

- A rapid move parallel to the XY-plane to (X9, Y12, Z4.8)
- Move parallel to the Z-axis at the feed rate to (X9, Y12, Z4.2)
- A rapid move parallel to the Z-axis to (X9, Y12, Z4.8)

The third repeat consists of 3 moves. The X position is reset to 13 (=9+4) and the Y position to 17 (=12+5).

- A rapid move parallel to the XY-plane to (X13, Y17, Z4.8)
- Move parallel to the Z-axis at the feed rate to (X13, Y17, Z4.2)
- A rapid move parallel to the Z-axis to (X13, Y17, Z4.8)

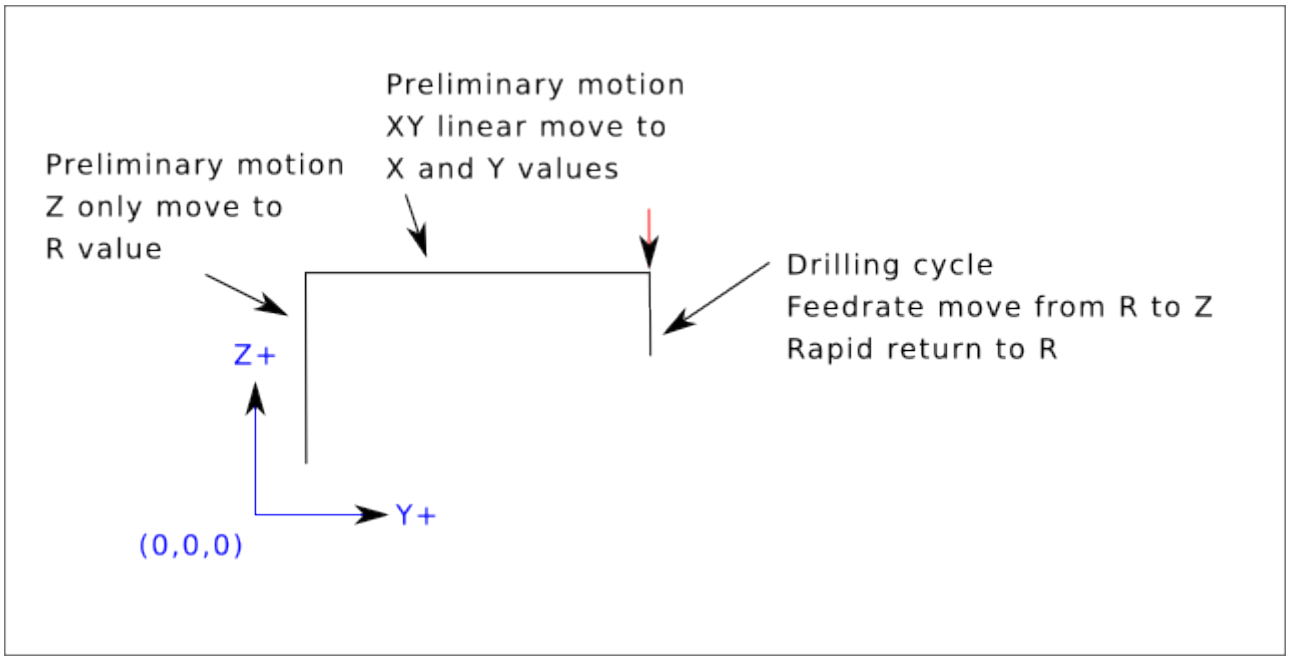


Example 3 - Relative Position G81

```
G90 G98 G81 X4 Y5 Z1.5 R2.8
```

Now suppose that you execute the first G81 block of code but from (X0, Y0, Z0) rather than from (X1, Y2, Z3).

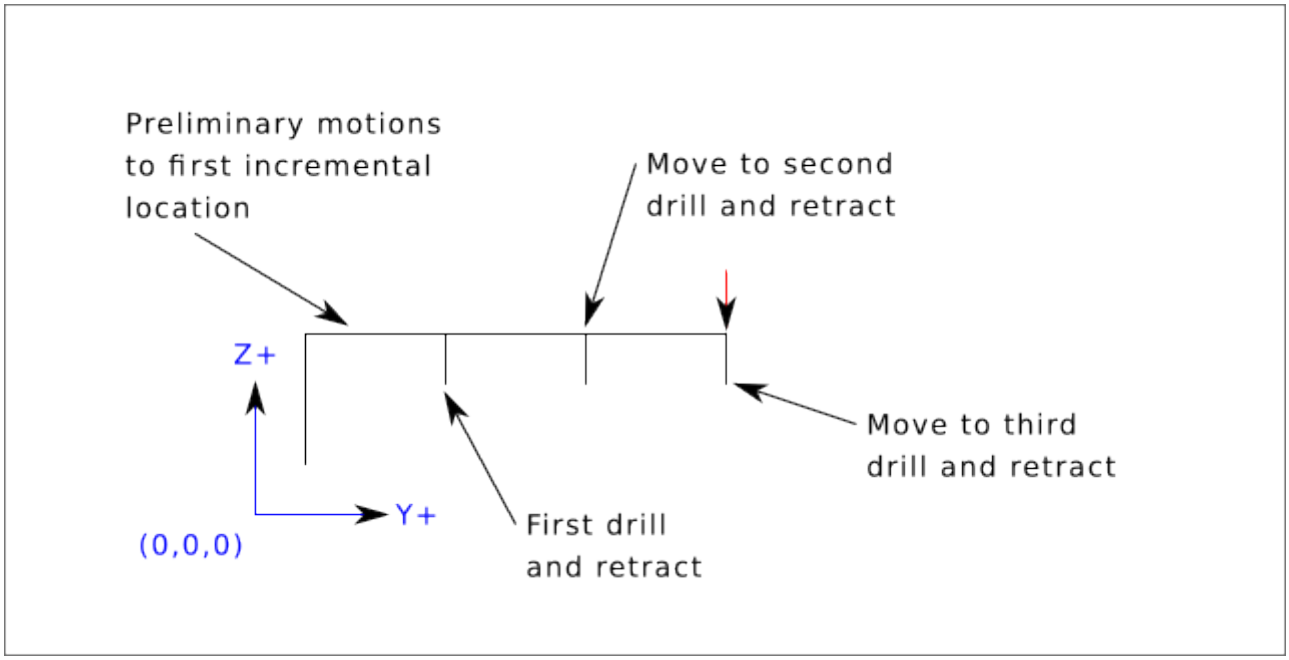
Como OLD_Z está por debajo del valor R, no agrega nada al movimiento, pero dado que el valor inicial de Z es menor que el valor especificado en R, habrá un movimiento inicial Z durante el movimiento preliminar.



Example 4 - Absolute G81 R > Z This is a plot of the path of motion for the second g81 block of code.

```
G91 G98 G81 X4 Y5 Z-0.6 R1.8 L3
```

Since this plot starts with (X0, Y0, Z0), the interpreter adds the initial Z0 and R1.8 and rapid moves to that location. After that initial Z move, the repeat feature works the same as it did in example 3 with the final Z depth being 0.6 below the R value.



Example 5 - Relative position R > Z

```
G90 G98 G81 X4 Y5 Z-0.6 R1.8
```

Since this plot starts with (X0, Y0, Z0), the interpreter adds the initial Z0 and R1.8 and rapid moves to that location as in *Example 4*. After that initial Z move, the rapid move to X4 Y5 is done. Then the

final Z depth being 0.6 below the R value. The repeat function would make the Z move in the same location again.

11.5.46. G82 Drilling Cycle, Dwell

```
G82 (X- Y- Z-) or (U- V- W-) R- L- P-
```

The *G82* cycle is intended for drilling with a dwell at the bottom of the hole.

- Movimiento preliminar, como se describe en la sección Movimientos preliminares e intermedios.
- Mover el eje Z a la valocidad de avance actual a la posición Z.
- Espera P segundos.
- The Z-axis does a rapid move to clear Z.

The motion of a G82 canned cycle looks just like G81 with the addition of a dwell at the bottom of the Z move. The length of the dwell is specified by a *P-* word in the G82 block.

```
G90 G82 G98 X4 Y5 Z1.5 R2.8 P2
```

Similar al ejemplo 3 arriba, solo con una espera adicional de 2 segundos en el fondo del agujero.

11.5.47. G83 Peck Drilling Cycle

```
G83 (X- Y- Z-) or (U- V- W-) R- L- Q- D-
```

The *G83* cycle (often called peck drilling) is intended for deep drilling or milling with chip breaking. The retracts in this cycle clear the hole of chips and cut off any long stringers (which are common when drilling in aluminum). This cycle takes a Q number which represents a *delta* increment along the Z-axis. The retract before final depth will always be to the *retract* plane even if G98 is in effect. The final retract will honor the G98/99 in effect. G83 functions the same as G81 with the addition of retracts during the drilling operation. Peck clearance can be specified by optional D number.

- Movimiento preliminar, como se describe en la sección Movimientos preliminares e intermedios.
- Move the Z-axis at the current feed rate downward by delta or to the Z position, whichever is less deep.
- Rapid move back out to the retract plane specified by the R word.
- Rapid up either the D value, the G83_PECK_CLEARANCE specified in the INI file or the default of .010” / 0.254 mm.
- Repeat steps 2, 3, and 4 until the Z position is reached at step 2.
- The Z-axis does a rapid move to clear Z.

It is an error if:

- the Q number is negative or zero.

11.5.48. G84 Right-hand Tapping Cycle, Dwell

```
G84 (X- Y- Z-) or (U- V- W-) R- L- P- $- F-
```

- *R-* - Retract position along the Z axis.
- *L-* - Used in incremental mode; number of times to repeat the cycle. See G81 for examples.
- *P-* - Dwell time (seconds).
- *$-* - Selected spindle.
- *F-* - Feed rate (spindle speed multiplied by distance traveled per revolution (thread pitch)).



aviso
G84 does not use synchronized motion.

The *G84* cycle is intended for tapping with floating chuck and dwell at the bottom of the hole.

- Movimiento preliminar, como se describe en la sección Movimientos preliminares e intermedios.
- Deshabilita los ajustes de avance y velocidad.
- Mueve el eje Z a la velocidad de avance actual a la posición Z.
- Stop the selected spindle (chosen by the $ parameter)
- Start spindle rotation counterclockwise.
- Espera P segundos.
- Move the Z-axis at the current feed rate to clear Z
- Restaurar ajustes de alimentación y velocidad al estado anterior

The length of the dwell is specified by a *P-* word in the G84 block. The feed rate *F-* is spindle speed multiplied by distance per revolution (thread pitch). In example S100 with 1.25MM per revolution thread pitch gives a feed of F125.

11.5.49. G85 Boring Cycle, Feed Out

```
G85 (X- Y- Z-) or (U- V- W-) R- L-
```

The *G85* cycle is intended for boring or reaming, but could be used for drilling or milling.

- Movimiento preliminar, como se describe en la sección Movimientos preliminares e intermedios.
- Mueve el eje Z solo a la velocidad de avance actual hacia la posición Z.
- Retract the Z-axis at the current feed rate to the R plane if it is lower than the initial Z.
- Retract at the traverse rate to clear Z.

11.5.50. G86 Boring Cycle, Spindle Stop, Rapid Move Out

```
G86 (X- Y- Z-) or (U- V- W-) R- L- P- $-
```

The *G86* cycle is intended for boring. This cycle uses a P number for the number of seconds to dwell.

- Movimiento preliminar, como se describe en la sección Movimientos preliminares e intermedios.
- Mueve el eje Z solo a la velocidad de avance actual hacia la posición Z.
- Espera P segundos.
- Detiene el giro del husillo seleccionado. (Elegido por el parámetro $)
- The Z-axis does a rapid move to clear Z.
- Restart the spindle in the direction it was going.

It is an error if:

- the spindle is not turning before this cycle is executed.

11.5.51. G87 Back Boring Cycle

Este código no está implementado actualmente en LinuxCNC. Se acepta, pero el comportamiento es indefinido.

11.5.52. G88 Boring Cycle, Spindle Stop, Manual Out

Este código no está implementado actualmente en LinuxCNC. Se acepta, pero el comportamiento es indefinido.

11.5.53. G89 Boring Cycle, Dwell, Feed Out

```
G89 (X- Y- Z-) or (U- V- W-) R- L- P-
```

The *G89* cycle is intended for boring. This cycle uses a P number, where P specifies the number of seconds to dwell.

- Movimiento preliminar, como se describe en la sección Movimientos preliminares e intermedios.
- Mueve el eje Z solo a la velocidad de avance actual hacia la posición Z.
- Espera P segundos.
- Retract the Z-axis at the current feed rate to clear Z.

11.5.54. G90, G91 Distance Mode

- *G90* - modo de distancia absoluta. En este modo, los números de eje (X, Y, Z, A, B, C, U, V, W) generalmente representan posiciones en términos del sistema coordinado activo. Las excepciones a esta regla se describen explícitamente en la sección G80 G89.
- *G91* - incremental distance mode In incremental distance mode, axis numbers usually represent increments from the current coordinate.

G90 Example

```
G90 (set absolute distance mode)
G0 X2.5 (rapid move to coordinate X2.5 including any offsets in effect)
```

G91 Example

```
G91 (set incremental distance mode)
G0 X2.5 (rapid move 2.5 from current position along the X axis)
```

- See G0 section for more information.

11.5.55. G90.1, G91.1 Arc Distance Mode

- *G90.1* - absolute distance mode for I, J & K offsets. When G90.1 is in effect I and J both must be specified with G2/3 for the XY plane or J and K for the XZ plane or it is an error.
- *G91.1* - incremental distance mode for I, J & K offsets. `G91.1` Returns I, J & K to their default behavior.

11.5.56. G92 Coordinate System Offset

```
G92 axes
```



aviso
Only use *G92* after your machine has been positioned to the desired point.

G92 makes the current point have the coordinates you want (without motion), where the axis words contain the axis numbers you want. All axis words are optional, except that at least one must be used. If an axis word is not used for a given axis, the offset for that axis will be zero.

Cuando se ejecuta *G92*, se mueven los orígenes de todos los sistemas de coordenadas. Se mueven de tal manera que el valor del punto controlado actual, en el sistema de coordenadas actualmente activo, se convierte en el valor especificado. Todos los orígenes de sistema de coordenadas (G53-G59.3) se compensan con esta misma distancia.

G92 utiliza los valores almacenados en los parámetros 5211-5219 como los valores de offset X Y Z A B C U V W para cada eje. Los valores de los parámetros son coordenadas de máquina *absolutas* en las *unidades* nativas de máquina como se especificó en el archivo INI. Todos los ejes definidos en el archivo INI se compensarán cuando G92 esté activo. Si no se ingresó un eje después del G92, el offset de ese eje será cero.

For example, suppose the current point is at X=4 and there is currently no *G92* offset active. Then *G92 X7* is programmed. This moves all origins -3 in X, which causes the current point to become X=7. This -3 is saved in parameter 5211.

Being in incremental distance mode (G91 instead of G90) has no effect on the action of G92.

G92 offsets may be already be in effect when the G92 is called. If this is the case, the offset is replaced with a new offset that makes the current point become the specified value.

It is an error if all axis words are omitted.

LinuxCNC stores the G92 offsets and reuses them on the next run of a program. To prevent this, one can program a G92.1 (to erase them), or program a G92.2 (to remove them - they are still stored).

nota
The *G52* command can also be used to change this offset; see the Local and Global Offsets section for more details about G92 and G52 and how they interact.

Consulte la sección Sistema de coordenadas para obtener una descripción general de los sistemas de coordenadas.

Vea la sección Parámetros para mas información.

11.5.57. G92.1, G92.2 Reset G92 Offsets

- *G92.1* - turn off G92 offsets and reset parameters 5211 - 5219 to zero.
- *G92.2* - turn off G92 offsets but keep parameters 5211 - 5219 available.

nota
G92.1 only clears G92 offsets, to change G53-G59.3 coordinate system offsets in G-code use either G10 L2 or G10 L20.

11.5.58. G92.3 Restore G92 Offsets

- *G92.3* - set the G92 offset to the values saved in parameters 5211 to 5219

You can set axis offsets in one program and use the same offsets in another program. Program G92 in the first program. This will set parameters 5211 to 5219. Do not use G92.1 in the remainder of the first program. The parameter values will be saved when the first program exits and restored when the second one starts up. Use G92.3 near the beginning of the second program. That will restore the offsets saved in the first program.

11.5.59. G93, G94, G95 Feed Rate Mode

- *G93* - is Inverse Time Mode. In inverse time feed rate mode, an F word means the move should be completed in [one divided by the F number] minutes. For example, if the F number is 2.0, the move should be completed in half a minute.

  Cuando el modo de velocidad de avance de tiempo inverso está activo, debe aparecer una palabra F en cada línea que tiene un movimiento G1, G2 o G3. Si la linea no tiene G1, G2 o G3, F se ignora. Estar en modo de velocidad de tiempo inverso no afecta a movimentos rápidos G0.

- *G94* - is Units per Minute Mode. In units per minute feed mode, an F word is interpreted to mean the controlled point should move at a certain number of inches per minute, millimeters per minute, or degrees per minute, depending upon what length units are being used and which axis or axes are moving.
- *G95* - Modo Unidades por Revolución. En modo de unidades por revolución, una palabra F se interpreta como que el punto controlado debe moverse un cierto número de pulgadas por revolución del husillo, dependiendo de qué unidades de longitud se estén utilizando y cuáles eje o ejes se mueven. `G95` no es adecuado para roscar, use `G33` o `G76`. `G95` requiere que se conecte `spindle.N.speed-in`. El husillo real con el que se sincroniza la alimentacion se elige con el parámetro `$`.

It is an error if:

- Inverse time feed mode is active and a line with `G1`, `G2`, or `G3` (explicitly or implicitly) does not have an F-word.
- A new feed rate is not specified after switching to `G94` or `G95`

11.5.60. G96, G97 Spindle Control Mode

```
G96 <D-> S- <$-> (Constant Surface Speed Mode)
G97 S- <$-> (RPM Mode)
```

1. *D* - maximum rotation speed (RPM), optional
2. *S* - spindle speed
3. *$* - the spindle of which the speed will be varied, optional.
   - *G96 S- <D->* - selecciona la velocidad superficial constante de *S*:
     - In feet per minute if `G20` is in effect,
     - or meters per minute if `G21` is in effect.

When using G96, ensure that X0 in the current coordinate system (including offsets and tool lengths) is the center of rotation or LinuxCNC will not give the desired surface speed. G96 is not affected by radius or diameter mode.

To achieve CSS mode on selected spindles programme successive G96 commands for each spindle prior to issuing M3.

- *G97* selects RPM mode.

G96 Example Line

```
G96 D2500 S250 (set CSS with a max rpm of 2500 and a surface speed of 250)
```

It is an error if:

- S is not specified with G96
- A feed move is specified in G96 mode while the spindle is not turning

11.5.61. G98, G99 Canned Cycle Return Level

When spindle retracts during canned cycles, there are two options to choose from for the way it does it:

- *G98* - retract to the position that axis was in just before this series of one or more contiguous canned cycles was started.
- *G99* - retract to the position specified by the R word of the canned cycle.

Programe un *G98* y el ciclo fijo usará la posición Z anterior al ciclo fijo como la posición de retorno Z si es más alto que el valor R especificado en el ciclo. Si es menor, se usará el valor R. La palabra R tiene diferentes significados en modo de distancia absoluta y en modo de distancia incremental.

G98 Retract to Origin

```
G0 X1 Y2 Z3
G90 G98 G81 X4 Y5 Z-0.6 R1.8 F10
```

G98 en la segunda línea significa que el movimiento de retorno será al valor de Z en la primera línea, ya que es más alto que el valor R especificado.

The *initial* (G98) plane is reset any time cycle motion mode is abandoned, whether explicitly (G80) or implicitly (any motion code that is not a cycle). Switching among cycle modes (say G81 to G83) does NOT reset the *initial* plane. It is possible to switch between G98 and G99 during a series of cycles.

11.6. M-Codes

11.6.1. M-Code Quick Reference Table

Código	Descripción
M0 M1	Program Pause
M2 M30	Program End
M60	Pallet Change Pause
M3 M4 M5	Spindle Control
M6	Cambio de herramienta
M7 M8 M9	Coolant Control
M19	Orient Spindle
M48 M49	Feed & Spindle Overrides Enable/Disable
M50	Feed Override Control
M51	Spindle Override Control
M52	Adaptive Feed Control
M53	Feed Stop Control
M61	Set Current Tool Number
m62-m65	Control de salida
M66	Input Control
M67	Analog Output Control
M68	Analog Output Control
M70	Save Modal State
M71	Invalidate Stored Modal State
M72	Restore Modal State
M73	Save Autorestore Modal State
M98 M99	Call and Return From Subprogram

Código	Descripción
M100-M199	User Defined M-Codes

11.6.2. M0, M1 Program Pause

* *M0* - pause a running program temporarily. LinuxCNC remains in the Auto Mode so MDI and other manual actions are not enabled. Pressing the resume button will restart the program at the following line. * *M1* - pause a running program temporarily if the optional stop switch is on. LinuxCNC remains in the Auto Mode so MDI and other manual actions are not enabled. Pressing the resume button will restart the program at the following line.

nota
It is OK to program *M0* and *M1* in MDI mode, but the effect will probably not be noticeable, because normal behavior in MDI mode is to stop after each line of input anyway.

11.6.3. M2, M30 Program End

* *M2* - end the program. Pressing `Cycle Start` (”R” in the Axis GUI) will restart the program at the beginning of the file. * *M30* - exchange pallet shuttles and end the program. Pressing `Cycle Start` will start the program at the beginning of the file.

Both of these commands have the following effects:

- Change from Auto mode to MDI mode.
- Origin offsets are set to the default (like *G54*).
- Selected plane is set to XY plane (like *G17*).
- Distance mode is set to absolute mode (like *G90*).
- Feed rate mode is set to units per minute (like *G94*).
- Feed and speed overrides are set to ON (like *M48*).
- Cutter compensation is turned off (like *G40*).
- The spindle is stopped (like *M5*).
- The current motion mode is set to feed (like *G1*).
- Coolant is turned off (like *M9*).

nota
Lines of code after M2/M30 will not be executed. Pressing `Cycle Start` will start the program at the beginning of the file.

aviso
Using % to wrap the G-code does not do the same thing as a *Program End*. See the section on File Requirements for more information on what using % does not do.

11.6.4. M60 Pallet Change Pause

* *M60* - intercambia lanzaderas de palets y luego detiene un programa en ejecución temporalmente (independientemente de la configuración de parada opcional). Presionando el botón de inicio del ciclo reiniciará el programa en la línea siguiente.

11.6.5. M3, M4, M5 Spindle Control

* *M3 [$n]* - start the selected spindle clockwise at the *S* speed. * *M4 [$n]* - start the selected spindle counterclockwise at the *S* speed. * *M5 [$n]* - stop the selected spindle.

Use $ to operate on specific spindles. If $ is omitted then the commands default to operating on spindle 0. Use $-1 to operate on all active spindles.

This example will start spindles 0, 1, and 2 simultaneously at different speeds:

```
S100 $0
S200 $1
S300 $2
M3 $-1
```

Este ejemplo invertirá el husillo 1 pero dejará los otros husillos girando hacia adelante:

```
M4 $1
```

And this will stop spindle 2 and leave the other spindles rotating:

```
M5 $2
```

Si se omite $, entonces el comportamiento es exactamente el normal para una máquina de un solo husillo.

It is OK to use *M3* or *M4* if the S spindle speed is set to zero. If this is done (or if the speed override switch is enabled and set to zero), the spindle will not start turning. If, later, the spindle speed is set above zero (or the override switch is turned up), the spindle will start turning. It is OK to use *M3* or *M4* when the spindle is already turning or to use *M5* when the spindle is already stopped.

11.6.6. M6 Tool Change

11.6.6.1. Cambio manual de herramienta

If the HAL component *hal_manualtoolchange* is loaded, M6 will stop the spindle and prompt the user to change the tool based on the last *T*- number programmed. For more information on hal_manualtoolchange see the Manual Tool Change section.

11.6.6.2. Tool Changer

To change a tool in the spindle from the tool currently in the spindle to the tool most recently selected (using a T word - see section Select Tool), program *M6*. When the tool change is complete:

- The spindle will be stopped.
- The tool that was selected (by a T word on the same line or on any line after the previous tool change) will be in the spindle.
- If the selected tool was not in the spindle before the tool change, the tool that was in the spindle (if there was one) will be placed back into the tool changer magazine.

- If configured in the INI file some axis positions may move when a M6 is issued. See the EMCIO section for more information on tool change options.
- No other changes will be made. For example, coolant will continue to flow during the tool change unless it has been turned off by an *M9*.

nota
La palabra *T-* es un número entero que designa el número de ranura de herramienta en el carrusel (no su índice).



aviso
The tool length offset is not changed by *M6*, use *G43* after the *M6* to change the tool length offset.

The tool change may include axis motion. It is OK (but not useful) to program a change to the tool already in the spindle. It is OK if there is no tool in the selected slot; in that case, the spindle will be empty after the tool change. If slot zero was last selected, there will definitely be no tool in the spindle after a tool change. The tool changer will have to be setup to perform the tool change in HAL and possibly ClassicLadder.

11.6.7. M7, M8, M9 Coolant Control

* *M7* - turn mist coolant on. M7 controls iocontrol.0.coolant-mist pin. * *M8* - turn flood coolant on. M8 controls iocontrol.0.coolant-flood pin. * *M9* - turn both M7 and M8 off.

Connect one or both of the coolant control pins in HAL before M7 or M8 will control an output. M7 and M8 can be used to turn on any output via G-code.

It is OK to use any of these commands, regardless of the current coolant state.

11.6.8. M19 Orient Spindle

```
M19 R- Q- [P-] [$-]
```

- *R* Position to rotate to from 0, valid range is 0-360 degrees
- *Q* Número de segundos de espera hasta que se complete la orientación. Si spindle.N.is-oriented no se hace true dentro del tiempo de espera Q, se produce un error.
- *P* Direction to rotate to position.
  - *0* rotate for smallest angular movement (default)
  - *1* always rotate clockwise (same as M3 direction)
  - *2* always rotate counterclockwise (same as M4 direction)
- *$* The spindle to orient (actually only determines which HAL pins carry the spindle position commands)

M19 is a command of modal group 7, like M3, M4 and M5. M19 is cleared by any of M3,M4,M5.

Spindle orientation requires a quadrature encoder with an index to sense the spindle shaft position and direction of rotation.

Configuración INI en la sección [RS274NGC]:

- ORIENT_OFFSET = 0-360 (fixed offset in degrees added to M19 R word)
- HAL Pins
  - *spindle.N.orient-angle* (out float) Desired spindle orientation for M19. Value of the M19 R word parameter plus the value of the [RS274NGC]ORIENT_OFFSET INI parameter.
  - *spindle.N.orient-mode* (out s32) Modo de rotación deseado del husillo. Refleja la palabra del parámetro M19 P, predeterminado = 0.
  - *spindle.N.orient* (out bit) Indicates start of spindle orient cycle. Set by M19. Cleared by any of M3,M4,M5. If spindle-orient-fault is not zero during spindle-orient true, the M19 command fails with an error message.
  - *spindle.N.is-oriented* (in bit) Pin de reconocimiento para orientacion del husillo. Completa el ciclo de orientación. Si spindle-orient era true cuando se aseveró spindle-is-oriented, el pin spindle-orient se limpia y se asevera el pin spindle-locked. Además, se asevera el pin spindle-brake.
  - *spindle.N.orient-fault* (in s32) Fault code input for orient cycle. Any value other than zero will cause the orient cycle to abort.
  - *spindle.N.locked* (out bit) Spindle orient complete pin. Cleared by any of M3,M4,M5.

11.6.9. M48, M49 Speed and Feed Override Control

* *M48* - enable the spindle speed and feed rate override controls. * *M49* - disable both controls.

These commands also take an optional $ parameter to determine which spindle they operate on.

It is OK to enable or disable the controls when they are already enabled or disabled. See the Feed Rate section for more details.

They also can be be toggled individually using *M50* and *M51*, see below.

11.6.10. M50 Feed Override Control

* *M50 <P1>* - enable the feed rate override control. The P1 is optional. * *M50 P0* - disable the feed rate control.

While disabled the feed override will have no influence, and the motion will be executed at programmed feed rate. (unless there is an adaptive feed rate override active).

11.6.11. M51 Spindle Speed Override Control

* *M51 <P1> <$->*- enable the spindle speed override control for the selected spindle. The P1 is optional. * *M51 P0 <$->* - disable the spindle speed override control program.

Mientras está deshabilitado, el ajuste de velocidad del husillo no tendrá influencia, y la velocidad del husillo tendrá el valor exacto de programa especificado de la palabra-S (descrita en la sección Velocidad del Husillo).

11.6.12. M52 Adaptive Feed Control

* *M52 <P1>* - use an adaptive feed. The P1 is optional. * *M52 P0* - stop using adaptive feed.

When adaptive feed is enabled, some external input value is used together with the user interface feed override value and the commanded feed rate to set the actual feed rate. In LinuxCNC, the HAL pin *motion.adaptive-feed* is used for this purpose. The range for *motion.adaptive-feed* is defined by the MAX_FEED_OVERRIDE value in the [DISPLAY] section of the ini file and will be limited to a range of -MAX_FEED_OVERRIDE to MAX_FEED_OVERRIDE. 0 is equivalent to feed-hold.

nota
The use of negative adaptive-feed for reverse run is a new feature and is not very well tested as yet. The intended use is for plasma cutters and wire spark eroders but it is not limited to such applications.

11.6.13. M53 Feed Stop Control

* *M53 <P1>* - enable the feed stop switch. The P1 is optional. Enabling the feed stop switch will allow motion to be interrupted by means of the feed stop control. In LinuxCNC, the HAL pin *motion.feed-hold* is used for this purpose. A *true* value will cause the motion to stop when *M53* is active. * *M53 P0* - disable the feed stop switch. The state of *motion.feed-hold* will have no effect on feed when M53 is not active.

11.6.14. M61 Set Current Tool

* *M61 Q-* - change the current tool number while in MDI or Manual mode without a tool change. One use is when you power up LinuxCNC with a tool currently in the spindle you can set that tool number without doing a tool change.



aviso
The tool length offset is not changed by *M61*, use *G43* after the *M61* to change the tool length offset.

It is an error if:

- Q- is not 0 or greater

11.6.15. M62 - M65 Digital Output Control

* *M62 P-* - activa la salida digital sincronizada con el movimiento. * *M63 P-* - apaga la salida digital sincronizada con el movimiento. * *M64 P-* - activa la salida digital de inmediato. * *M65 P-* - apaga la salida digital inmediatamente.

The P- word specifies the digital output number. The P-word ranges from 0 to a default value of 3. If needed the the number of I/O can be increased by using the num_dio parameter when loading the motion controller. See the Motion section for more information.

Los comandos M62 y M63 se pondrán en cola. Comandos posteriores referentes al mismo número de salida sobrescribirá la configuración anterior. Se puede especificar mas de un cambio de salida emitiendo más de un comando M62/M63.

The actual change of the specified outputs will happen at the beginning of the next motion command. If there is no subsequent motion command, the queued output changes won't happen. It's best to always program a motion G-code (G0, G1, etc) right after the M62/63.

M64 y M65 suceden inmediatamente cuando son recibidos por el controlador de movimiento. No están sincronizados con el movimiento, y romperán la mezcla.

nota
M62-65 will not function unless the appropriate motion.digital-out-*nn* pins are connected in your HAL file to outputs.

11.6.16. M66 Wait on Input

```
M66 P- | E- <L->
```

- *P-* - especifica el número de entrada digital de 0 a 3. (Ajustable desde motmod con argumento num_dio)
- *E-* - especifica el número de entrada analógica de 0 a 3. (Ajustable desde motmod con el argumento num_aio)
- *L-* - specifies the wait mode type.
  - *Mode 0: IMMEDIATE* - no waiting, returns immediately. The current value of the input is stored in parameter #5399
  - *Mode 1: RISE* - waits for the selected input to perform a rise event.
  - *Mode 2: FALL* - waits for the selected input to perform a fall event.
  - *Mode 3: HIGH* - waits for the selected input to go to the HIGH state.
  - *Mode 4: LOW* - waits for the selected input to go to the LOW state.
- *Q-* - specifies the timeout in seconds for waiting. If the timeout is exceeded, the wait is interrupt, and the variable #5399 will be holding the value -1. The Q value is ignored if the L-word is zero (IMMEDIATE). A Q value of zero is an error if the L-word is non-zero.
- Mode 0 is the only one permitted for an analog input.

M66 Example Lines

```
M66 P0 L3 Q5 (wait up to 5 seconds for digital input 0 to turn on)
```

M66 wait on an input stops further execution of the program, until the selected event (or the programmed timeout) occurs.

It is an error to program M66 with both a P-word and an E-word (thus selecting both an analog and a digital input). In LinuxCNC these inputs are not monitored in real time and thus should not be used for timing-critical applications.

El número de E/S se puede aumentar utilizando el parámetro num_dio o num_aio al cargar el controlador de movimiento. Vea la sección Movimiento para mas información.

---

nota
M66 will not function unless the appropriate motion.digital-in-*nn* pins or motion.analog-in-*nn* pins are connected in your HAL file to an input.

Example HAL Connection

```
net signal-name motion.digital-in-00 <= parport.0.pin10-in
```

11.6.17. M67 Analog Output, Synchronized

```
M67 E- Q-
```

- *M67* - set an analog output synchronized with motion.

- *E-* - número de salida de 0 a 3 (Ajustable desde motmod con el argumento num_aio)
- *Q-* - is the value to set (set to 0 to turn off).

The actual change of the specified outputs will happen at the beginning of the next motion command. If there is no subsequent motion command, the queued output changes won't happen. It's best to always program a motion G-code (G0, G1, etc) right after the M67. M67 functions the same as M62-63.

El número de E/S se puede aumentar utilizando el parámetro num_dio o num_aio al cargar el controlador de movimiento. Vea la sección Movimiento para mas información.

nota
M67 will not function unless the appropriate motion.analog-out-*nn* pins are connected in your HAL file to outputs.

11.6.18. M68 Analog Output, Immediate

```
M68 E- Q-
```

- *M68* - set an analog output immediately.
- *E-* - número de salida que va de 0 a 3. (Ajustable desde motmod con el argumento num_aio)
- *Q-* - is the value to set (set to 0 to turn off).

M68 output happen immediately as they are received by the motion controller. They are not synchronized with movement, and they will break blending. M68 functions the same as M64-65.

El número de E/S se puede aumentar utilizando el parámetro num_dio o num_aio al cargar el controlador de movimiento. Vea la sección Movimiento para mas información.

nota
M68 will not function unless the appropriate motion.analog-out-*nn* pins are connected in your HAL file to outputs.

11.6.19. M70 Save Modal State

To explicitly save the modal state at the current call level, program *M70*. Once modal state has been saved with *M70*, it can be restored to exactly that state by executing an *M72*.

Un par de instrucciones *M70* y *M72* generalmente se utilizarán para proteger un programa contra cambios modales involuntarios dentro de subrutinas.

M70 Saved state

The state saved consists of:

- current G20/G21 settings (imperial/metric)
- selected plane (G17/G18/G19 G17.1,G18.1,G19.1)
- status of cutter compensation (G40,G41,G42,G41.1,G42,1)
- distance mode - relative/absolute (G90/G91)

- feed mode (G93/G94,G95)
- current coordinate system (G54-G59.3)
- tool length compensation status (G43,G43.1,G49)
- retract mode (G98,G99)
- spindle mode (G96-css or G97-RPM)
- arc distance mode (G90.1, G91.1)
- lathe radius/diameter mode (G7,G8)
- path control mode (G61, G61.1, G64)
- current feed and speed (*F* and *S* values)
- spindle status (M3,M4,M5) - on/off and direction
- mist (M7) and flood (M8) status
- speed override (M51) and feed override (M50) settings
- adaptive feed setting (M52)
- feed hold setting (M53)

Note that in particular, the motion mode (G1 etc) is NOT restored.

current call level means either:

- executing in the main program. There is a single storage location for state at the main program level; if several *M70* instructions are executed in turn, only the most recently saved state is restored when an *M72* is executed.
- executing within a G-code subroutine. The state saved with *M70* within a subroutine behaves exactly like a local named parameter - it can be referred to only within this subroutine invocation with an *M72* and when the subroutine exits, the parameter goes away.

A recursive invocation of a subroutine introduces a new call level.

11.6.20. M71 Invalidate Stored Modal State

Modal state saved with an *M70* or by an *M73* at the current call level is invalidated (cannot be restored from anymore).

A subsequent *M72* at the same call level will fail.

If executed in a subroutine which protects modal state by an *M73*, a subsequent return or endsub will **not** restore modal state.

The usefulness of this feature is dubious. It should not be relied upon as it might go away.

11.6.21. M72 Restore Modal State

Modal state saved with an M70 code can be restored by executing an *M72*.

The handling of G20/G21 is specially treated as feeds are interpreted differently depending on G20/G21: if length units (mm/in) are about to be changed by the restore operation, 'M72 'will restore the distance mode first, and then all other state including feed to make sure the feed value is interpreted in the correct unit setting.

It is an error to execute an *M72* with no previous *M70* save operation at that level.

The following example demonstrates saving and explicitly restoring modal state around a subroutine call using *M70* and *M72*. Note that the *imperialsub* subroutine is not "aware" of the M7x features and can be used unmodified:

```
O<showstate> sub
(DEBUG, imperial=#<_imperial> absolute=#<_absolute> feed=#<_feed> rpm=#<_rpm>)
O<showstate> endsub

O<imperialsub> sub
g20 (imperial)
g91 (relative mode)
F5 (low feed)
S300 (low rpm)
(debug, in subroutine, state now:)
o<showstate> call
O<imperialsub> endsub

; main program
g21 (metric)
g90 (absolute)
f200 (fast speed)
S2500 (high rpm)

(debug, in main, state now:)
o<showstate> call

M70 (save caller state in at global level)
O<imperialsub> call
M72 (explicitly restore state)

(debug, back in main, state now:)
o<showstate> call
m2
```

11.6.22. M73 Save and Autorestore Modal State

To save modal state within a subroutine, and restore state on subroutine *endsub* or any *return* path, program *M73*.

Aborting a running program in a subroutine which has an *M73* operation will **not** restore state .

Also, the normal end (*M2*) of a main program which contains an *M73* will **not** restore state.

The suggested use is at the beginning of a O-word subroutine as in the following example. Using *M73* this way enables designing subroutines which need to modify modal state but will protect the calling program against inadvertent modal changes. Note the use of predefined named parameters in the *showstate* subroutine.

```
O<showstate> sub
(DEBUG, imperial=#<_imperial> absolute=#<_absolute> feed=#<_feed> rpm=#<_rpm>)
```

```
O<showstate> endsub

O<imperialsub> sub
M73 (save caller state in current call context, restore on return or endsub)
g20 (imperial)
g91 (relative mode)
F5 (low feed)
S300 (low rpm)
(debug, in subroutine, state now:)
o<showstate> call

; note - no M72 is needed here - the following endsub or an
; explicit 'return' will restore caller state
O<imperialsub> endsub

; main program
g21 (metric)
g90 (absolute)
f200 (fast speed)
S2500 (high rpm)
(debug, in main, state now:)
o<showstate> call
o<imperialsub> call
(debug, back in main, state now:)
o<showstate> call
m2
```

11.6.23. M98 and M99

The interpreter supports Fanuc-style main- and sub-programs with the *M98* and *M99* M-codes. See Fanuc-Style Programs.

11.6.23.1. Selectively Restoring Modal State

Executing an *M72* or returning from a subroutine which contains an *M73* will restore all modal state saved.

Si solo se deben preservar algunos aspectos del estado modal, una alternativa es el uso de parámetros con nombre predefinido, parámetros locales y declaraciones condicionales. La idea es recordar los modos que se restaurarán en el comienzo de la subrutina y restaurar estos antes de salir. Aquí está un ejemplo, basado en el fragmento de *nc_files/tool-length-probe.ngc*:

```
O<measure> sub   (measure reference tool)
;
#<absolute> = #<_absolute>  (remember in local variable if G90 was set)
;
g30 (above switch)
g38.2 z0 f15 (measure)
g91 g0z.2 (off the switch)
#1000=#5063 (save reference tool length)
(print,reference length is #1000)
;
O<restore_abs> if [#<absolute>]
    g90 (restore G90 only if it was set on entry:)
O<restore_abs> endif
;
O<measure> endsub
```

11.6.24. M100-M199 Comandos definidos por el usuario

```
M1-- <P- Q->
```

- *M1--* - an integer in the range of 100 - 199.
- *P-* - a number passed to the file as the first parameter.
- *Q-* - a number passed to the file as the second parameter.

nota
After creating a new *M1nn* file you must restart the GUI so it is aware of the new file, otherwise you will get an *Unknown m code* error.

El programa externo llamado *M100* a *M199* (sin extensión, M mayúscula, que se encuentra en el directorio señalado en el parámetro *[DISPLAY] PROGRAM_PREFIX* del archivo INI) se ejecuta con los valores opcionales P y Q como sus dos argumentos.

Execution of the G-code file pauses until the external program exits. If the external program exits with exit code other than *0* G-code program execution is stopped. Any valid executable file can be used. The file must be located in the search path specified in the INI file configuration. See the Display section for more information on search paths.

After creating a new M1*nn* program, the GUI should be restarted so that the new program is taken into account, otherwise a *Unknown M-code* error will occur.

aviso
Do not use a word processor to create or edit the files. A word processor will leave unseen codes that will cause problems and may prevent a bash or python file from working. Use a text editor like Geany in Debian or Notepad++ in other operating systems to create or edit the files.

El error *Código M desconocido utilizado* denota uno de los siguientes:

- El comando definido por el usuario no existe.
- El archivo no es ejecutable.
- El nombre del archivo tiene una extensión.
- El nombre del archivo no sigue este formato Mnnn donde nnn = 100 a 199.
- El nombre del archivo utiliza una M minúscula.

For example to open and close a collet closer that is controlled by a parallel port pin using a bash script file using M101 and M102. Create two files named M101 and M102. Set them as executable files (typically right click/properties/permissions) before running LinuxCNC. Make sure the parallel port pin is not connected to anything in a HAL file.

M101 Example File

```
#!/bin/bash
# file to turn on parport pin 14 to open the collet closer
halcmd setp parport.0.pin-14-out True
exit 0
```

M102 Example File

```
#!/bin/bash
# file to turn off parport pin 14 to open the collet closer
halcmd setp parport.0.pin-14-out False
exit 0
```

To pass a variable to a M1nn file you use the P and Q option like this:

```
M100 P123.456 Q321.654
```

M100 Example file

```
#!/bin/bash
voltage=$1
feedrate=$2
halcmd setp thc.voltage $voltage
halcmd setp thc.feedrate $feedrate
exit 0
```

To display a graphic message and stop until the message window is closed use a graphic display program like Eye of Gnome to display the graphic file. When you close it the program will resume.

M110 Example file

```
#!/bin/bash
eog /home/john/linuxcnc/nc_files/message.png
exit 0
```

To display a graphic message and continue processing the G-code file suffix an ampersand to the command.

M110 Example display and keep going

```
#!/bin/bash
eog /home/john/linuxcnc/nc_files/message.png &
exit 0
```

11.7. O Codes

11.7.1. Use of O-codes

O-codes provide for flow control in NC programs. Each block has an associated number, which is the number used after O. Care must be taken to properly match the O-numbers. O-codes use the letter *O* not the number zero as the first character in the number like O100 or o100. O-codes are also sometimes called o-words and these terms are interchangeable.

---

nota
Using the lower case *o* makes it easier to distinguish from a 0 that might have been mistyped. For example *o100* is easier to see than *O100* that it is not a *0*.

11.7.2. Numbering

There are two categories of o-codes with different scoping rules:

Subroutine definitions (`sub`/`endsub`, `call`) are **global**. Each subroutine must have a unique number or name across the entire program and all called files.

Control flow (`if`/`endif`, `while`/`endwhile`, `do`/`while`, `repeat`/`endrepeat`, `break`, `continue`) are **local** to the subroutine or main program where they appear. The interpreter automatically scopes them, so `o100 if` inside `o<helper>` does not conflict with `o100 if` in the main program or in any other subroutine. You can safely reuse the same o-numbers for control flow in different subroutines.

Within a single subroutine body (or the main program), each o-number should still be used for only one control flow block.

Numbering Example

```
(the start of o100)
o100 sub
(notice that the if-endif block uses a different number within this sub)
  (the start of o110)
  o110 if [#2 GT 5]
    (some code here)
  (the end of o110)
  o110 endif
  (some more code here)
(the end of o100)
o100 endsub
```

Reusing Control Flow Numbers Across Subroutines (valid)

```
(o100 if in helper.ngc does not conflict with o100 if in another.ngc)

(file: helper.ngc)
o<helper> sub
  o100 if [#1 GT 5]
    (do something)
  o100 endif
o<helper> endsub

(file: another.ngc)
o<another> sub
  o100 if [#1 LT 0]
    (do something else)
  o100 endif
o<another> endsub
```

11.7.3. Comentarios

Comments on the same line as the o-word should not be used as the behavior can change in the future.

The behavior is undefined if:

- The same number is used for more than one block within the same scope (see [?] for scoping rules).
- Se usan otras palabras en una línea con una palabra O.
- Se usan comentarios en una línea con una palabra O.

11.7.4. Subroutines

Subroutines starts at *oNNN sub* and ends at *oNNN endsub*. The lines between *oNNN sub* and *oNNN endsub* are not executed until the subroutine is called with *oNNN call*. Each subroutine must use a unique number.

Subroutine Example

```
o100 sub
  G53 G0 X0 Y0 Z0 (rapid move to machine home)
o100 endsub

(the subroutine is called)
o100 call
M2
```

See G53, G0 and M2 sections for more information.

O- Return Inside a subroutine, *o- return* can be executed. This immediately returns to the calling code, just as though *o- endsub* was encountered.

O- Return Example

```
o100 sub
  (test if parameter #2 is greater than 5)
  o110 if [#2 GT 5]
    (return to top of subroutine if test is true)
    o100 return
  o110 endif
    (this only gets executed if parameter #2 is not greater than 5)
    (DEBUG, parameter 1 is [#1])
o100 endsub
```

See the Binary Operators and Parameters sections for more information.

O- Call *o- Call* takes up to 30 optional arguments, which are passed to the subroutine as *#1*, *#2* , ..., #N. Parameters from #N+1 to #30 have the same value as in the calling context. On return from the subroutine, the values of parameters #1 through #30 (regardless of the number of arguments) will be restored to the values they had before the call. Parameters #1 - #30 are local to the subroutine.

Because *1 2 3* is parsed as the number 123, the parameters must be enclosed in square brackets. The following calls a subroutine with 3 arguments:

O- Call Example

```
o100 sub
  (test if parameter #2 is greater than 5)
  o110 if [#2 GT 5]
    (return to top of subroutine if test is true)
    o100 return
  o110 endif
    (this only gets executed if parameter #2 is not greater than 5)
    (DEBUG, parameter 1 is [#1])
    (DEBUG, parameter 3 is [#3])
o100 endsub

o100 call [100] [2] [325]
```

Subroutine **definitions** may not be nested. Defining a subroutine inside another subroutine is not allowed and will produce an error. For example, the following is **invalid**:

Invalid Nested Subroutine Definition (produces an error)

```
o<outer> sub
  o100 sub       (INVALID: nested subroutine definition)
    (code here)
  o100 endsub
o<outer> endsub
```

However, **calling** a subroutine from within another subroutine is perfectly valid. It is the `sub`/`endsub` definition that cannot be nested, not the `call`:

Valid Nested Subroutine Call

```
o<outer> sub
  (code here)
  o<helper> call [1] [2]  (OK: calling another sub)
o<outer> endsub
```

Other O-code control flow such as `if`/`endif`, `while`/`endwhile`, `do`/`while`, and `repeat`/`endrepeat` may be used freely inside subroutines.

Subroutines may only be called after they are defined. They may be called from other functions, and may call themselves recursively if it makes sense to do so. The maximum subroutine nesting level is 10.

Subroutines can change the value of parameters above #30 and those changes will be visible to the calling code. Subroutines may also change the value of global named parameters (i.e. parameters whose names begin with the underscore character "_").

11.7.4.1. Fanuc-Style Numbered Programs

Numbered programs (both main and subprograms), the *M98* call and *M99* return M-codes, and their respective semantic differences are an alternative to the rs274ngc subroutines described above, provided for compatibility with Fanuc and other machine controllers.

Numbered programs are enabled by default, and may be disabled by placing `DISABLE_FANUC_STYLE_SUB = 1` in the `[RS274NGC]` section of the `.ini` file.

---

nota
Numbered main and subprogram definitions and calls differ from traditional rs274ngc both in syntax and execution. To reduce the possibility of confusion, the interpreter will raise an error if definitions of one style are mixed with calls of another.

Numbered Subprogram Simple Example

```
o1 (Example 1)    ; Main program 1, "Example 1"
M98 P100          ; Call subprogram 100
M30               ; End main program

o100              ; Beginning of subprogram 100
  G53 G0 X0 Y0 Z0 ; Rapid move to machine home
M99               ; Return from subprogram 100
```

o1 (Title) The optional main program beginning block gives the main program the number `1`. Some controllers treat an optional following parenthesized comment as a program title, `Example 1` in this example, but this has no special meaning in the rs274ngc interpreter.

M98 P- <L-> Call a numbered subprogram. The block `M98 P100` is analogous to the traditional `o100 call` syntax, but may only be used to call a following numbered subprogram defined with `o100`...`M99`. An optional *L*-word specifies a loop count.

M30 The main program must be terminated with M02 or M30 (or M99; see below).

O- subprogram definition start Marks the start of a numbered subprogram definition. The block o100 is similar to o100 sub, except that it must be placed later in the file than the M98 P100 calling block.

M99 return from numbered subroutine The block M99 is analogous to the traditional o100 endsub syntax, but may only terminate a numbered program (o100 in this example), and may not terminate a subroutine beginning with the o100 sub syntax.

The M98 subprogram call differs from rs274ngc o call in the following ways:

- The numbered subprogram must follow the M98 call in the program file. The interpreter will throw an error if the subprogram precedes the call block.
- Parameters #1, #2, ..., #30 are global and accessible in numbered subprograms, similar to higher-numbered parameters in traditional style calls. Modifications to these parameters within a subprogram are global modifications, and will be persist after subprogram return.
- M98 subprogram calls have no return value.
- M98 subprogram call blocks may contain an optional L-word specifying a loop repeat count. Without the L-word, the subprogram will execute once only (equivalent to M98 L1). An M98 L0 block will not execute the subprogram.

En casos raros, el código-M M99 puede usarse para terminar el programa principal, donde indica un *programa sin fin*. Cuando el intérprete alcanza un M99 en el programa principal, saltará de nuevo al comienzo del archivo y reanudará la ejecución en la primera línea. Un ejemplo de un programa sin fin es en un ciclo de calentamiento de la máquina; se puede usar un final de programa de eliminación de bloque /M30 para detener el ciclo en un punto ordenado cuando el operador está listo.

Numbered Subprogram Full Example

```
o1                              ; Main program 1
  #1 = 0
  (PRINT,X MAIN BEGIN:  1=#1)
  M98 P100 L5                   ; Call subprogram 100
  (PRINT,X MAIN END:  1=#1)
M30                             ; End main program

o100                            ; Subprogram 100
  #1 = [#1 + 1]
  M98 P200 L5                   ; Call subprogram 200
  (PRINT,>> o100:  #1)
M99                             ; Return from Subprogram 100

o200                            ; Subprogram 200
  #1 = [#1 + 0.01]
  (PRINT,>>>> o200:  #1)
M99                             ; Return from Subprogram 200
```

In this example, parameter #1 is initialized to 0. Subprogram o100 is called five times in a loop. Nested within each call to o100, subprogram o200 is called five times in a loop, for 25 times total.

Note that parameter #1 is global. At the end of the main program, after updates within o100 and o200, its value will equal 5.25.

11.7.5. Looping

El *bucle while* tiene dos estructuras: *while/endwhile* y *do/while*. En cada caso, se sale del ciclo cuando la condición *while* se evalúa como falso. La diferencia es cuando se realiza la condición de prueba.

El bucle *do/while* ejecuta el código en el bucle y luego verifica la condición de prueba. El bucle *while/endwhile* hace primero la prueba.

While Endwhile Example

```
(draw a sawtooth shape)
G0 X1 Y0 (move to start position)
#1 = 0 (assign parameter #1 the value of 0)
F25 (set a feed rate)
o101 while [#1 LT 10]
  G1 X0
  G1 Y[#1/10] X1
  #1 = [#1+1] (increment the test counter)
o101 endwhile
M2 (end program)
```

Do While Example

```
#1 = 0 (assign parameter #1 the value of 0)
o100 do
  (debug, parameter 1 = #1)
  o110 if [#1 EQ 2]
    #1 = 3 (assign the value of 3 to parameter #1)
    (msg, #1 has been assigned the value of 3)
    o100 continue (skip to start of loop)
  o110 endif
  (some code here)
  #1 = [#1 + 1] (increment the test counter)
o100 while [#1 LT 3]
(msg, Loop Done!)
M2
```

Dentro de un ciclo while, *o- break* sale inmediatamente del ciclo, y *o- continue* salta inmediatamente a la próxima evaluación de la condición *while*. Si aún es cierta, el ciclo comienza nuevamente en la parte superior. Si es falsa, sale del bucle.

11.7.6. Conditional

The *if* conditional consists of a group of statements with the same *o* number that start with *if* and end with *endif*. Optional *elseif* and *else* conditions may be between the starting *if* and the ending *endif*.

If the *if* conditional evaluates to true then the group of statements following the *if* up to the next conditional line are executed.

Si el condicional *if* se evalúa como falso, entonces las condiciones *elseif* son evaluadas en orden hasta que una evalúa como verdadera. Si la condición *elseif* es verdadera entonces se ejecutan las declaraciones que siguen al *elseif* hasta el próximo condicional. Si ninguna de las condiciones *if* o *elseif* se evalúa como verdadera, entonces se ejecutan las declaraciones que siguen al *else*. Cuando una condición es evaluada como verdadero, no se evalúan más condiciones en el grupo.

If Endif Example

```
(if parameter #31 is equal to 3 set S2000)
o101 if [#31 EQ 3]
  S2000
o101 endif
```

If ElseIf Else EndIf Example

```
(if parameter #2 is greater than 5 set F100)
o102 if [#2 GT 5]
```

```
  F100
o102 elseif [#2 LT 2]
(else if parameter #2 is less than 2 set F200)
  F200
(else if parameter #2 is 2 through 5 set F150)
o102 else
  F150
o102 endif
```

Several conditions may be tested for by *elseif* statements until the *else* path is finally executed if all preceding conditions are false:

If Elseif Else Endif Example

```
(if parameter #2 is greater than 5 set F100)
o102 if [#2 GT 5]
  F100
(else if parameter #2 less than 2 set F200)
o102 elseif [#2 LT 2]
  F20
(parameter #2 is between 2 and 5)
o102 else
  F200
o102 endif
```

11.7.7. Repeat

The *repeat* will execute the statements inside of the repeat/endrepeat the specified number of times. The example shows how you might mill a diagonal series of shapes starting at the present position.

Example with repeat

```
(Mill 5 diagonal shapes)
G91 (Incremental mode)
o103 repeat [5]
... (insert milling code here)
G0 X1 Y1 (diagonal move to next position)
o103 endrepeat
G90 (Absolute mode)
```

11.7.8. Indirection

The o-number may be given by a parameter and/or calculation.

Indirection Example

```
o[#101+2] call
```

Computing values in O-words Para obtener más información sobre los valores, consulte las siguientes secciones:

- Parameters
- Expressions
- Binary Operators
- Functions

11.7.9. Calling Files

To call a separate file with a subroutine name the file the same as your call and include a sub and endsub in the file. The file must be in the directory pointed to by *PROGRAM_PREFIX* or *SUBROUTINE_PATH* in the INI file. The file name can include **lowercase** letters, numbers, dash, and underscore only. A named subroutine file can contain only a single subroutine definition. Do not place additional numbered subroutine definitions (e.g. `o100 sub`) in the same file as a named subroutine. Numbered subroutines share a global namespace and can silently conflict across files. If you need helper subroutines, place each one in its own file.

Named File Example

```
o<myfile> call
```

Numbered File Example

```
o123 call
```

In the called file you must include the oxxx sub and endsub and the file must be a valid file.

Called File Example

```
(filename myfile.ngc)
o<myfile> sub
  (code here)
o<myfile> endsub
M2
```

---

nota
The file names are lowercase letters only so *o<MyFile>* is converted to *o<myfile>* by the interpreter. More information about the search path and options for the search path are in the INI configuration section.

11.7.10. Subroutine return values

Subroutines may optionally return a value by an optional expression at an *endsub* or *return* statement.

Return value example

```
o123 return [#2 *5]
...
o123 endsub [3 * 4]
```

A subroutine return value is stored in the *<_value>* predefined named parameter , and the *<_value_returned>* predefined parameter is set to 1, to indicate a value was returned. Both parameters are global, and are cleared just before the next subroutine call.

11.7.11. Errors

The following statements cause an error message and abort the interpreter:

- a `return` or `endsub` not within a sub definition
- a label on `repeat` which is defined elsewhere
- a label on `while` which is defińed elsewhere and not referring to a `do`

- a label on `if` defined elsewhere
- a undefined label on `else` or `elseif`
- a label on `else`, `elseif` or `endif` not pointing to a matching `if`
- a label on `break` or `continue` which does not point to a matching `while` or `do`
- a label on `endrepeat` or `endwhile` no referring to a corresponding `while` or `repeat`
- a subroutine definition (`sub`) inside another subroutine body (nested definitions)
- a numbered subroutine called from within a named subroutine file but not found in the offset table or as a separate file

To make these errors non-fatal warnings on stderr, set bit 0x20 in the `[RS274NGC]FEATURE=` mask ini option.

11.8. Other Codes

11.8.1. F: Set Feed Rate

Fx - set the feed rate to *x*. *x* is usually in machine units (inches or millimeters) per minute.

The application of the feed rate is as described in the Feed Rate Section, unless *inverse time feed rate mode* or *feed per revolution mode* are in effect, in which case the feed rate is as described in the G93 G94 G95 section.

11.8.2. S: Set Spindle Speed

Sx [$n] - establece la velocidad del husillo a *x* revoluciones por minuto (RPM). Con el $ opcional establece la velocidad del husillo para un husillo específico. Sin el $, el comando pasará por defecto a spindle.0.

The spindle(s) or selected spindle will turn at that speed when a *M3* or *M4* is in effect. It is OK to program an S word whether the spindle is turning or not. If the speed override switch is enabled and not set at 100 %, the speed will be different from what is programmed.

It is OK to program S0, the spindle will not turn if that is done.

It is an error if:

- the S number is negative.

As described in the section Right-hand Tapping Cycle with Dwell, if a *G84* (tapping) drilling cycle is active and the speed and feed potentiometers are enabled, the one with the lowest setting will be used. The rotational speed and feed rate will remain synchronized. In this case, the speed may differ from the one programmed, even if the speed correction potentiometer is set to 100 %.

11.8.3. T: Seleccionar herramienta

Tx - prepare to change to tool *x*.

The tool is not changed until an *M6* is programmed (see Section M6). The T word may appear on the same line as the *M6* or on a previous line. It is OK if T words appear on two or more lines with no tool change. Only the the most recent T word will take effect at the next tool change.

nota
When LinuxCNC is configured for a nonrandom toolchanger (see the entry for RANDOM_TOOLCHANGER in the EMCIO Section), *T0* gets special handling: no tool will be selected. This is useful if you want the spindle to be empty after a tool change.

nota
When LinuxCNC is configured for a random toolchanger (see the entry for RANDOM_TOOLCHANGER in the EMCIO Section), *T0* does not get any special treatment: T0 is a valid tool like any other. It is customary to use T0 on a random toolchanger machine to track an empty pocket, so that it behaves like a nonrandom toolchanger machine and unloads the spindle.

It is an error if:

- a negative T number is used,
- T number is used that does not appear in the tool table file (with the exception that T0 on nonrandom toolchangers **is** accepted, as noted above).

On some machines, the carousel will move when a T word is programmed, at the same time machining is occurring. On such machines, programming the T word several lines before a tool change will save time. A common programming practice for such machines is to put the T word for the next tool to be used on the line after a tool change. This maximizes the time available for the carousel to move.

Rapid moves after a *T<n>* will not show on the AXIS preview until after a feed move. This is for machines that travel long distances to change the tool like a lathe. This can be very confusing at first. To turn this feature off for the current tool program a G1 without any move after the *T<n>*.

11.9. G-Code Examples

After you install LinuxCNC several sample files are placed in the /nc_files folder. Make sure the sample file is appropriate for your machine before running.

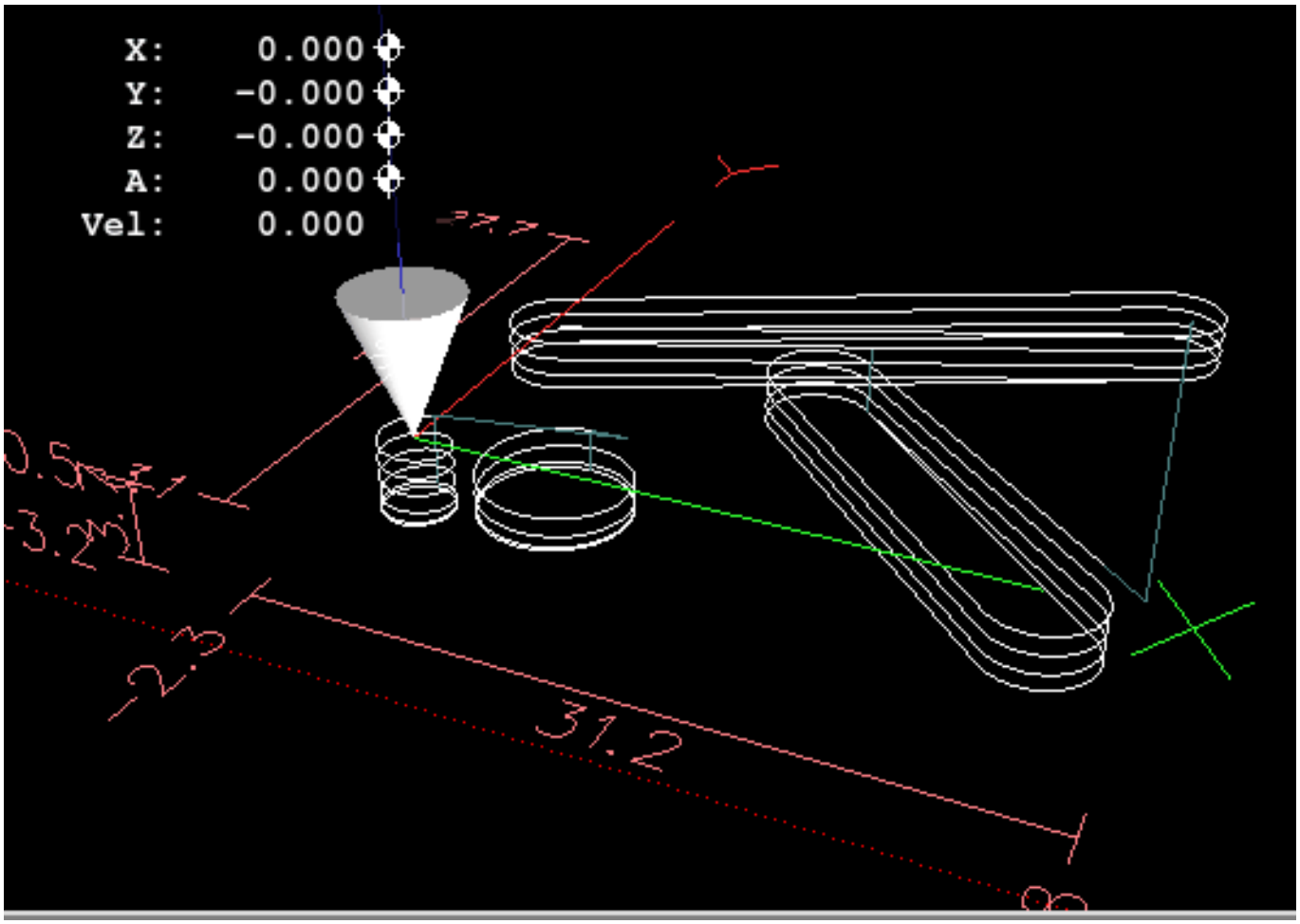
11.9.1. Mill Examples

11.9.1.1. Helical Hole Milling

- File Name: useful-subroutines.ngc
- Description: Subroutine for milling a hole using parameters.

11.9.1.2. Slotting

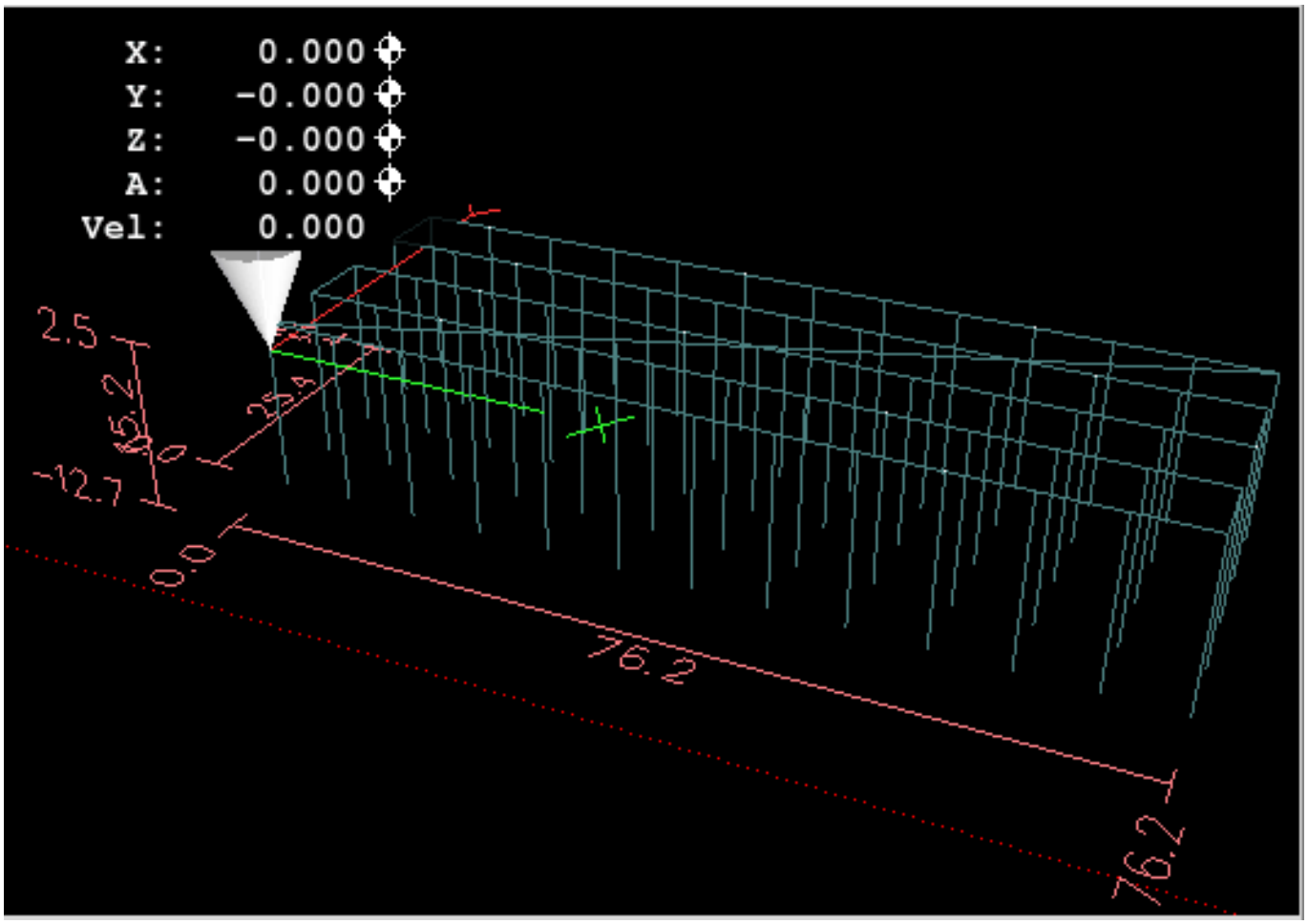
- File Name: useful-subroutines.ngc
- Description: Subroutine for milling a slot using parameters.



11.9.1.3. Grid Probe

- File Name: gridprobe.ngc
- Description: Rectangular Probing

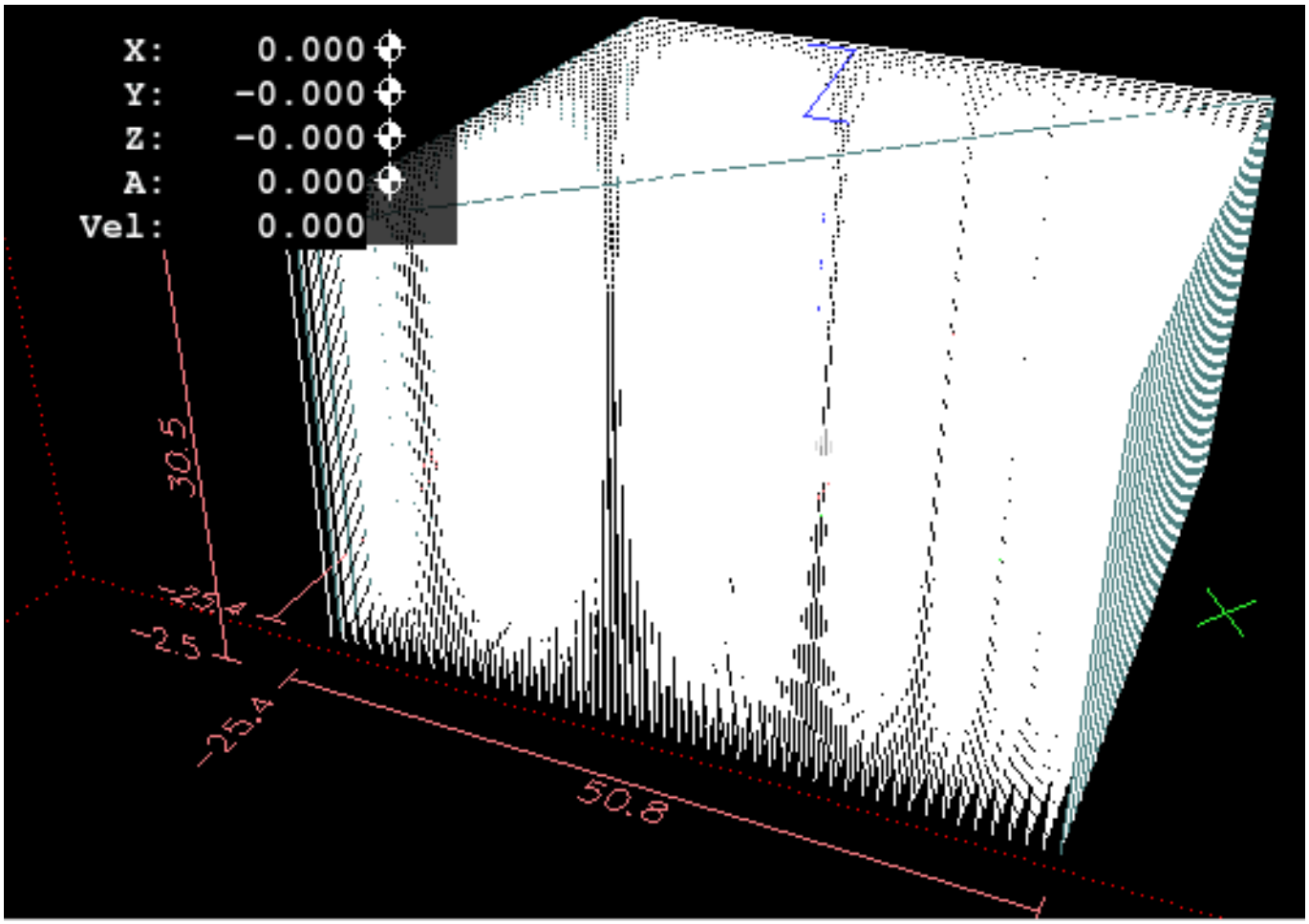
This program repeatedly probes in a regular XY grid and writes the probed location to the file *probe-results.txt* in the same directory as the .ini file.



11.9.1.4. Smart Probe

- File Name: smartprobe.ngc
- Description: Rectangular Probing

This program repeatedly probes in a regular XY grid and writes the probed location to the file *probe-results.txt* in the same directory as the .ini file. This is improved from the grid probe file.



11.9.1.5. Tool Length Probe

- File Name: tool-length-probe.ngc
- Description: Tool Length Probing

This program shows an example of how to measure tool lengths automatically using a switch hooked to the probe input. This is useful for machines without tool holders, where the length of a tool is different every time it is inserted.

11.9.1.6. Hole Probe

- File Name: probe-hole.ngc
- Description: Finding the Center and Diameter of a hole.

The program demonstrates how to find the center of a hole, measure the hole diameter and record the results.

11.9.1.7. Cutter Compensation

- File Name: comp-g1.ngc

- Description: Entry and exit movements with compensation of tool radius.

This program demonstrates the peculiarity of the toolpath without and with tool radius compensation. The tool radius is taken from the tool table.

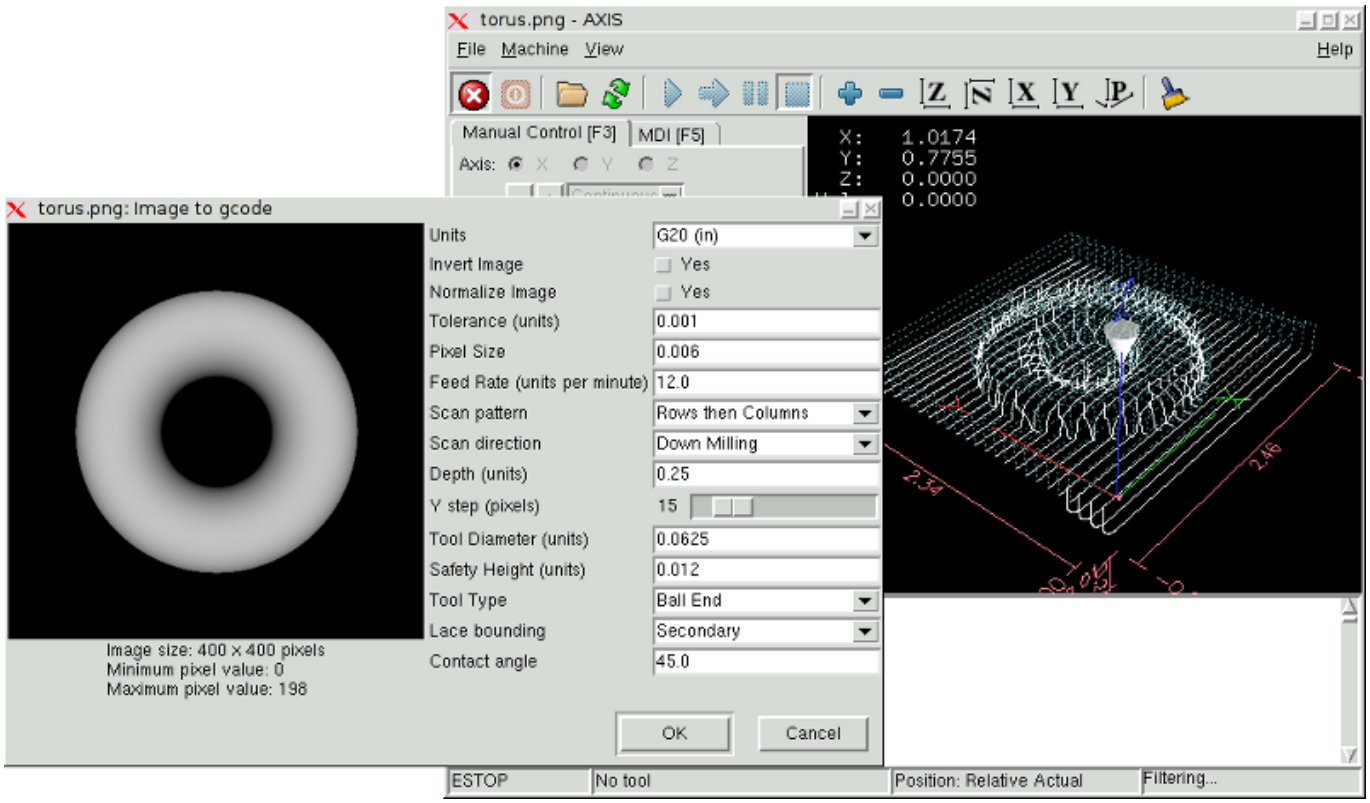
11.9.2. Lathe Examples

11.9.2.1. Roscado

- File Name lathe-g76.ngc
- Description: Facing, threading and parting off.

This file shows an example of threading on a lathe using parameters.

11.10. Image to G-Code



11.10.1. What is a depth map?

A depth map is a greyscale image where the brightness of each pixel corresponds to the depth (or height) of the object at each point.

11.10.2. Integrating image-to-gcode with the AXIS user interface

Add the following lines to the *[FILTER]* section of your INI file to make AXIS automatically invoke image-to-gcode when you open a PNG, GIF, or JPEG image:

```
PROGRAM_EXTENSION = .png,.gif,.jpg # Imagen de profundidad de escala de grises
png = image-to-gcode
gif = image-to-gcode
jpg = image-to-gcode
```

The standard *sim/axis.ini* configuration file is already prepared this way.

11.10.3. Using `image-to-gcode`

Start image-to-gcode either by opening an image file in AXIS, or by invoking image-to-gcode from the terminal, as follows:

```
image-to-gcode torus.png > torus.ngc
```

Verify all the settings in the right-hand column, then press OK to create the G-code. Depending on the image size and options chosen, this may take from a few seconds to a few minutes. If you are loading the image in AXIS, the G-code will automatically be loaded and previewed once image-to-gcode completes. In AXIS, hitting reload will show the image-to-gcode option screen again, allowing you to tweak them.

11.10.4. Option Reference

11.10.4.1. Unidades

Specifies whether to use G20 (inches) or G21 (mm) in the generated G-code and as the units for each option labeled *(units)*.

11.10.4.2. Invert Image

If ”no”, the black pixel is the lowest point and the white pixel is the highest point. If ”yes”, the black pixel is the highest point and the white pixel is the lowest point.

11.10.4.3. Normalize Image

If *yes*, the darkest pixel is remapped to black, the lightest pixel is remapped to white.

11.10.4.4. Expand Image Border

If *None*, the input image is used as-is, and details which are at the very edges of the image may be cut off. If *White* or *Black*, then a border of pixels equal to the tool diameter is added on all sides, and details which are at the very edges of the images will not be cut off.

11.10.4.5. Tolerance (units)

When a series of points are within *tolerance* of being a straight line, they are output as a straight line. Increasing tolerance can lead to better contouring performance in LinuxCNC, but can also remove or blur small details in the image.

11.10.4.6. Pixel Size (units)

One pixel in the input image will be this many units—usually this number is much smaller than 1.0. For instance, to mill a 2.5x2.5-inch object from a 400x400 image file, use a pixel size of .00625, because 2.5 / 400 = .00625.

11.10.4.7. Velocidad de alimentación por inmersión (unidades por minuto)

The feed rate for the initial plunge movement.

11.10.4.8. Velocidad de alimentación (unidades por minuto)

The feed rate for other parts of the path.

11.10.4.9. Velocidad del husillo (RPM)

The spindle speed S-code that should be put into the G-code file.

11.10.4.10. Scan Pattern

Possible scan patterns are:

- Rows
- Columns
- Rows, then Columns
- Columns, then Rows

11.10.4.11. Scan Direction

Possible scan directions are:

- Positive: Start milling at a low X or Y axis value, and move towards a high X or Y axis value.
- Negative: Start milling at a high X or Y axis value, and move towards a low X or Y axis value.
- Alternating: Start on the same end of the X or Y axis travel that the last move ended on. This reduces the amount of traverse movements.
- Up Milling: Start milling at low points, moving towards high points.
- Down Milling: Start milling at high points, moving towards low points.

11.10.4.12. Depth (units)

The top of material is always at *Z=0*. The deepest cut into the material is at *Z=-depth*.

11.10.4.13. Step Over (pixels)

The distance between adjacent rows or columns. To find the number of pixels for a given units distance, compute *distance/pixel size* and round to the nearest whole number. For example, if *pixel size=.006* and the desired step over *distance=.015*, then use a Step Over of 2 or 3 pixels, because *.015/.006=2.5*.

11.10.4.14. Tool Diameter

The diameter of the cutting part of the tool.

11.10.4.15. Safety Height

The height to move to for traverse movements. image-to-gcode always assumes the top of material is at *Z=0*.

11.10.4.16. Tool Type

The shape of the cutting part of the tool. Possible tool shapes are:

- Ball End
- Flat End
- 45 degree ”vee”
- 60 degree ”vee”

11.10.4.17. Lace bounding

This controls whether areas that are relatively flat along a row or column are skipped. This option only makes sense when both rows and columns are being milled. Possible bounding options are:

- None: Rows and columns are both fully milled.
- Secondary: When milling in the second direction, areas that do not strongly slope in that direction are skipped.
- Full: When milling in the first direction, areas that strongly slope in the second direction are skipped. When milling in the second direction, areas that do not strongly slope in that direction are skipped.

11.10.4.18. Contact angle

When *Lace bounding* is not *None*, slopes greater than *Contact angle* are considered to be *strong* slopes, and slopes less than that angle are considered to be weak slopes.

11.10.4.19. Roughing offset and depth per pass

Image-to-gcode can optionally perform roughing passes. The depth of successive roughing passes is given by *Roughing depth per pass*. For instance, entering 0.2 will perform the first roughing pass with a depth of 0.2, the second roughing pass with a depth of 0.4, and so on until the full Depth of the image is reached. No part of any roughing pass will cut closer than Roughing Offset to the final part. The following figure shows a tall vertical feature being milled. In this image, Roughing depth per pass is 0.2 inches and roughing offset is 0.1 inches.

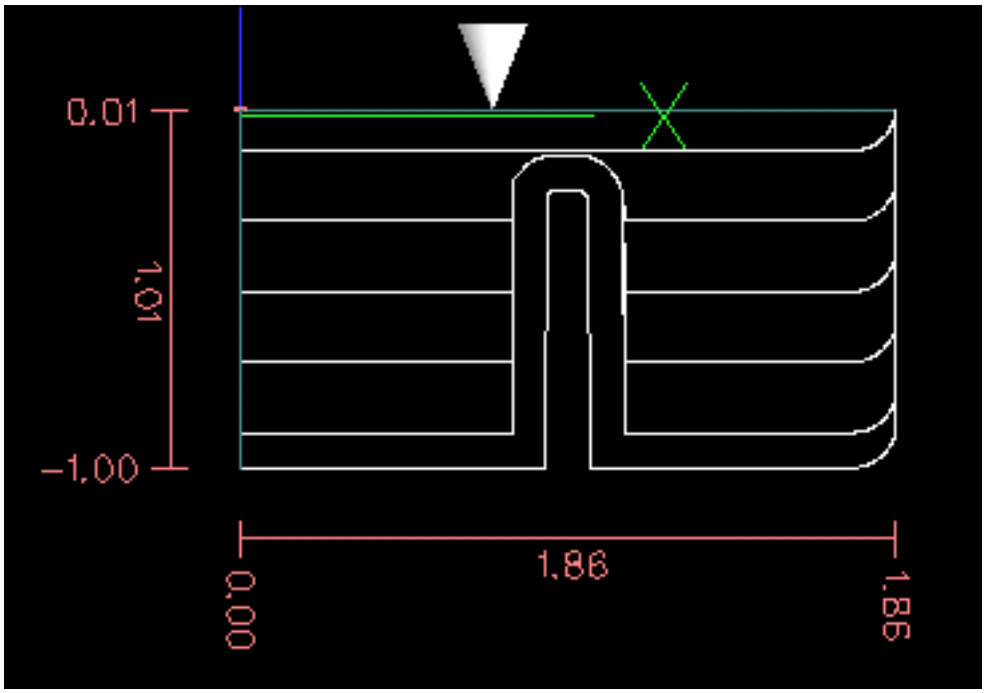


Figura 11.21: Roughing passes and final pass

11.11. RS274/NGC Differences

11.11.1. Changes from RS274/NGC

Differences that change the meaning of RS274/NGC programs

Location after a tool change
In LinuxCNC, the machine does not return to its original position after a tool change. This change was made because the new tool might be longer than the old tool, and the move to the original machine position could therefore leave the tool tip too low.

Offset parameters are INI file units
In LinuxCNC, the values stored in parameters for the G28 and G30 home locations, the P1…P9 coordinate systems, and the G92 offset are in ”INI file units”. This change was made because otherwise the meaning of a location changed depending on whether G20 or G21 was active when G28, G30, G10 L2, or G92.3 is programmed.

Tool table lengths/diameters are in INI file units
In LinuxCNC, the tool lengths (offsets) and diameters in the tool table are specified in INI file units only. This change was made because otherwise the length of a tool and its diameter would change based on whether G20 or G21 was active when initiating G43, G41, G42 modes. This made it impossible to run G-code in the machine’s non-native units, even when the G-code was simple and well-formed (starting with G20 or G21, and didn’t change units throughout the program), without changing the tool table.

G84, G87 not implemented
G84 and G87 are not currently implemented, but may be added to a future release of LinuxCNC.

G28, G30 with axis words
Cuando G28 o G30 se programan con solo algunas palabras de eje presentes, LinuxCNC solo

mueve los ejes nombrados. Esto es común en otros controles de máquinas. Para mover algunos ejes a un punto intermedio y luego mover todos los ejes al punto predefinido, escriba dos líneas de código G:

```
G0 X- Y- (axes to move to intermediate point)
G28 (move all axes to predefined point)
```

11.11.2. Additions to RS274/NGC

Differences that do not change the meaning of RS274/NGC programs

G33, G76 threading codes
These codes are not defined in RS274/NGC.

G38.2
The probe tip is not retracted after a G38.2 movement. This retraction move may be added in a future release of LinuxCNC.

G38.3...G38.5
These codes are not defined in RS274/NGC

O-codes
These codes are not defined in RS274/NGC

M50...M53 overrides
These codes are not defined in RS274/NGC

M61..M66
These codes are not defined in RS274/NGC

G43, G43.1
Negative Tool Lengths

The RS274/NGC spec says ”it is expected that” all tool lengths will be positive. However, G43 works for negative tool lengths.

Lathe tools

G43 tool length compensation can offset the tool in both the X and Z dimensions. This feature is primarily useful on lathes.

Dynamic tool lengths

LinuxCNC allows specification of a computed tool length through G43.1 I K.

G41.1, G42.1
LinuxCNC allows specification of a tool diameter and, if in lathe mode, orientation in the G-code. The format is G41.1/G42.1 D L, where D is diameter and L (if specified) is the lathe tool orientation.

G43 without H word
En NGC, esto no está permitido. En LinuxCNC, establece compensaciones de longitud para la herramienta cargada actualmente. Si no hay una herramienta cargada actualmente, es un error. Este cambio se realizó para que el usuario no tenga que especificar el número de herramienta en dos lugares para cada cambio de herramienta, y porque es consistente con la forma en que funciona G41/G42 cuando la palabra D no está especificada.

U, V, and W axes
LinuxCNC allows machines with up to 9 axes by defining an additional set of 3 linear axes known as U, V and W

Capítulo 12

Paneles Virtuales de Control

12.1. PyVCP

12.1.1. Introducción

PyVCP, **Py**thon **V**irtual **C**ontrol **P**anel, is designed to give the integrator the ability to customize the AXIS interface with buttons and indicators to do special tasks.

Hardware machine control panels can use up a lot of I/O pins and can be expensive. That is where Virtual Control Panels have the advantage as well as it cost nothing to build a PyVCP.

Virtual Control Panels can be used for testing or monitoring things to temporarily replace real I/O devices while debugging ladder logic, or to simulate a physical panel before you build it and wire it to an I/O board.

The following graphic displays many of the PyVCP widgets.

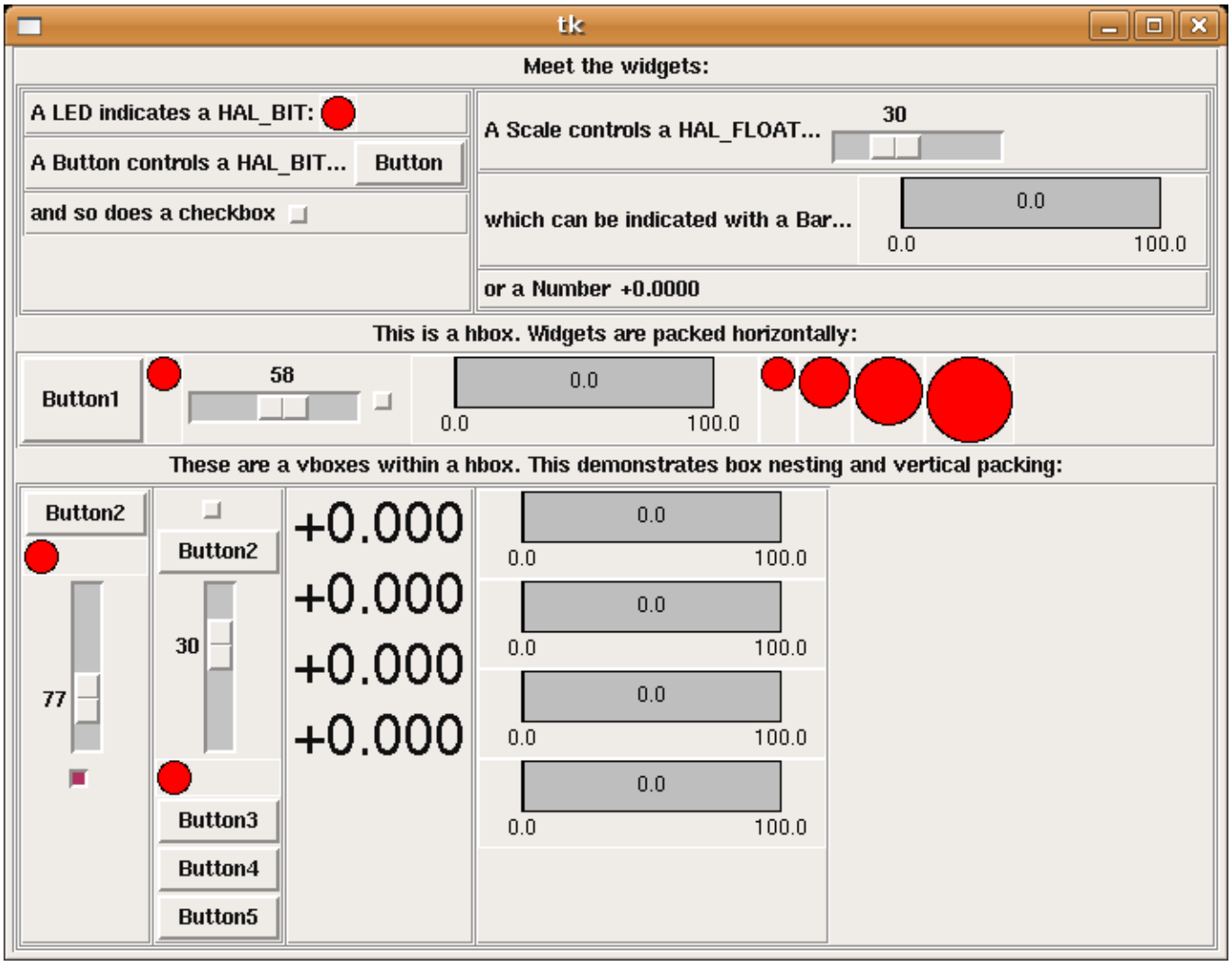


Figura 12.1: PyVCP Widgets Showcase

12.1.2. Panel Construction

The layout of a PyVCP panel is specified with an XML file that contains widget tags between <pyvcp> and </pyvcp>. For example:

```
<pyvcp>
    <label text="This is a LED indicator"/>
    <led/>
</pyvcp>
```



Figura 12.2: Simple PyVCP LED Panel Example

If you place this text in a file called tiny.xml, and run

```
halcmd loadusr pyvcp -c mypanel tiny.xml
```

PyVCP will create the panel for you, which includes two widgets, a Label with the text *This is a LED indicator*, and a LED, used for displaying the state of a HAL BIT signal. It will also create a HAL component named *mypanel* (all widgets in this panel are connected to pins that start with *mypanel.*). Since no <halpin> tag was present inside the <led> tag, PyVCP will automatically name the HAL pin for the LED widget mypanel.led.0

For a list of widgets and their tags and options, see the widget reference below.

Once you have created your panel, connecting HAL signals to and from the PyVCP pins is done with the halcmd:

```
net <signal-name> <pin-name> <opt-direction> <opt-pin-name>signal-name
```

If you are new to HAL, the HAL basics chapter in the Integrator Manual is a good place to start.

12.1.3. Security

Parts of PyVCP files are evaluated as Python code, and can take any action available to Python programs. Only use PyVCP XML files from a source that you trust.

12.1.4. AXIS

Since AXIS uses the same GUI toolkit (Tkinter) as PyVCP, it is possible to include a PyVCP panel at either the right side or the bottom of the AXIS user interface. It is not possible to display a panel in both of these positions simultaneously. A typical example is explained below.

In addition to or instead of displaying a PyVCP panel as described above, it is possible to display one or more PyVCP panels as embedded tabs in the AXIS GUI. This is achieved by the following in the `[DISPLAY]` section of the INI file:

```
EMBED_TAB_NAME    = Spindle
EMBED_TAB_COMMAND = pyvcp spindle.xml
```

The text label of the AXIS tab will display `Spindle`.

12.1.4.1. Panel de ejemplo

Place your PyVCP XML file describing the panel in the same directory where your INI file is. Say we we want to display the current spindle speed using a Bar widget. Place the following in a file called spindle.xml:

```
<pyvcp>
    <label>
        <text>”Spindle speed:”</text>
    </label>
    <bar>
        <halpin>”spindle-speed”</halpin>
        <max_>5000</max_>
    </bar>
</pyvcp>
```

Here we’ve made a panel with a Label and a Bar widget, specified that the HAL pin connected to the Bar should be named *spindle-speed*, and set the maximum value of the bar to 5000 (see widget reference below for all options). To make AXIS aware of this file, and call it at start up, we need to specify the following in the `[DISPLAY]` section of the INI file:

```
PYVCP = spindle.xml
```

Si el panel debe aparecer hasta abajo de la interfase de usuario de AXIS, entonces necesitamos especificar lo siguiente en la sección [DISPLAY] del archivo INI:

```
PYVCP_POSITION = BOTTOM
```

Anything other than BOTTOM or omitting this variable will place the PYVCP panel at the right.

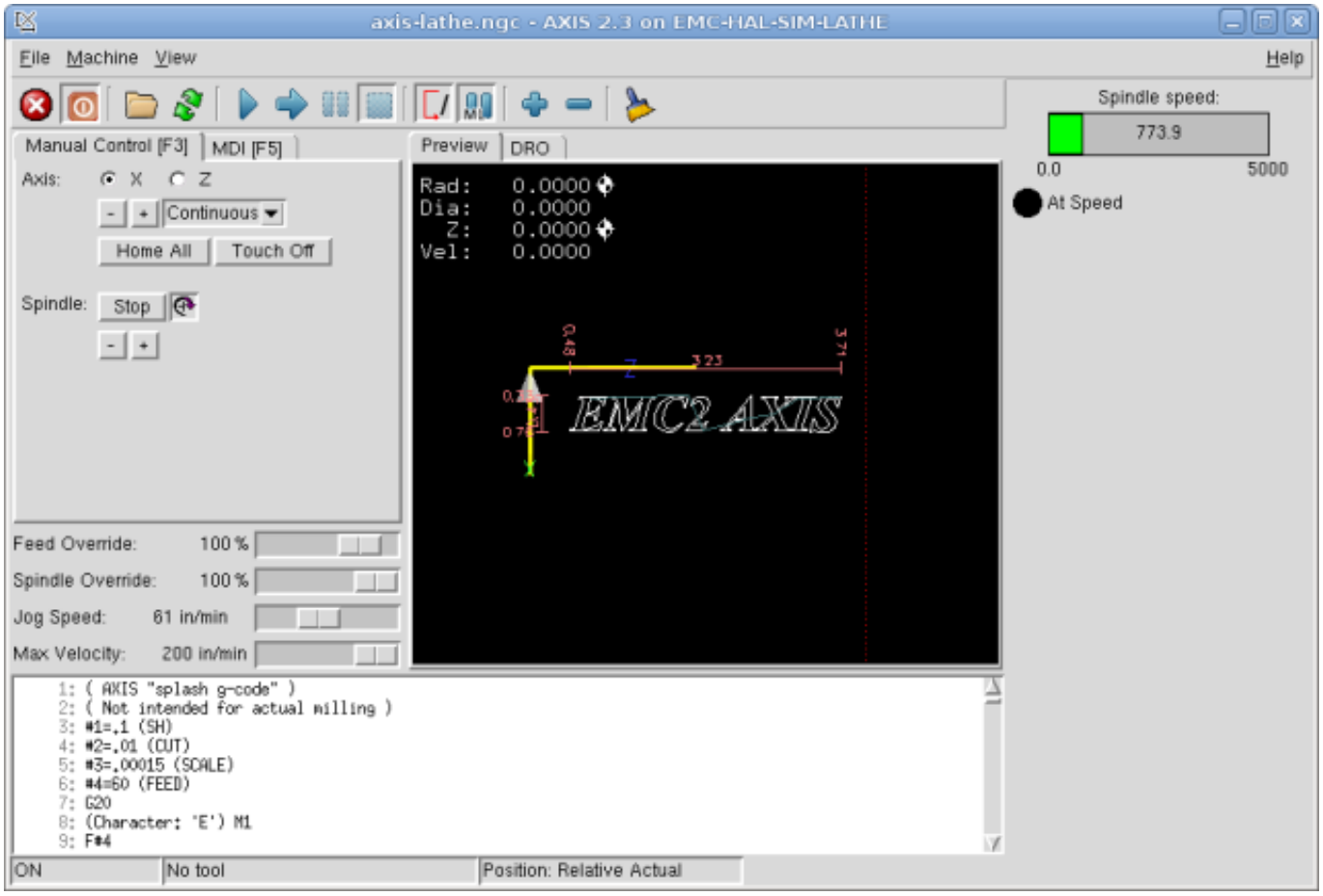
To make our widget actually display the spindle-speed it needs to be hooked up to the appropriate HAL signal. A HAL file that will be run once AXIS and PyVCP have started can be specified in the [HAL] section of the INI file:

```
POSTGUI_HALFILE = spindle_to_pyvcp.hal
```

This change will run the HAL commands specified in *spindle_to_pyvcp.hal*. In our example the contents could look like this:

```
net spindle-rpm-filtered => pyvcp.spindle-speed
```

assuming that a signal called *spindle-rpm-filtered* already exists. Note that when running together with AXIS, all PyVCP panel widget HAL pins have names that start with *pyvcp.*, all PyVCP embedded tab widget HAL pins start with the name specified as EMBED_TAB_NAME converted to lower case.



This is what the newly created PyVCP panel should look like in AXIS. The *sim/lathe* configuration is already configured this way.

12.1.5. Stand Alone

This section describes how PyVCP panels can be displayed on their own with or without LinuxCNC's machine controller.

To load a stand alone PyVCP panel with LinuxCNC use these commands:

```
loadusr -Wn mypanel pyvcp -g WxH+X+Y -c mypanel <path/>panel_file.xml
```

You would use this if you wanted a floating panel or a panel with a GUI other than AXIS.

- *-Wn panelname* - makes HAL wait for the component *panelname* to finish loading (*become ready* in HAL speak) before processing more HAL commands. This is important because PyVCP panels export HAL pins, and other HAL components will need them present to connect to them. Note the capital W and lowercase n. If you use the -Wn option you must use the -c option to name the panel.
- *pyvcp < -g> < -c> panel.xml* - builds the panel with the optional geometry and/or panelname from the XML panel file. The panel.xml can be any name that ends in .xml. The .xml file is the file that describes how to build the panel. You must add the path name if the panel is not in the directory that the HAL script is in.
- *-g <WxH><+X+Y>* - specifies the geometry to be used when constructing the panel. The syntax is *Width x Height + X Anchor + Y Anchor*. You can set the size or position or both. The anchor point is the upper left corner of the panel. An example is -g 250x500+800+0 This sets the panel at 250 pixels wide, 500 pixels tall, and anchors it at X800 Y0.
- *-c panelname* - tells PyVCP what to call the component and also the title of the window. The panelname can be any name without spaces.

To load a *stand alone* PyVCP panel without LinuxCNC use this command:

```
loadusr -Wn mypanel pyvcp -g 250x500+800+0 -c mypanel mypanel.xml
```

The minimum command to load a PyVCP panel is:

```
loadusr pyvcp mypanel.xml
```

You would use this if you want a panel without LinuxCNC's machine controller such as for testing or a standalone DRO.

The loadusr command is used when you also load a component that will stop HAL from closing until it's done. If you loaded a panel and then loaded Classic Ladder using *loadusr -w classicladder*, CL would hold HAL open (and the panel) until you closed CL. The *-Wn* above means wait for the component *-Wn "name"* to become ready. (*name* can be any name. Note the capital W and lowercase n.) The -c tells PyVCP to build a panel with the name *panelname* using the info in *panel_file_name.xml*. The name *panel_file_name.xml* can be any name but must end in .xml - it is the file that describes how to build the panel. You must add the path name if the panel is not in the directory that the HAL script is in.

An optional command to use if you want the panel to stop HAL from continuing commands / shutting down. After loading any other components you want the last HAL command to be:

```
waitusr panelname
```

This tells HAL to wait for component *panelname* to close before continuing HAL commands. This is usually set as the last command so that HAL shuts down when the panel is closed.

12.1.6. Widgets

HAL signals come in two variants, bits and numbers. Bits are off/on signals. Numbers can be *float*, *s32*, *u32*, *s64* or *u64*. For more information on HAL data types see the HAL Data section. The PyVCP widget can either display the value of the signal with an indicator widget, or modify the signal value with a control widget. Thus there are four classes of PyVCP widgets that you can connect to a HAL signal. A fifth class of helper widgets allow you to organize and label your panel.

- Widgets for indicating *bit* signals: `led`, `rectled`.
- Widgets for controlling *bit* signals: `button`, `checkbutton`, `radiobutton`.
- Widgets for indicating *number* signals: `number`, `s32`, `u32`, `bar`, `meter`.
- Widgets for controlling *number* signals: `spinbox`, `scale`, `jogwheel`.
- Helper widgets: `hbox`, `vbox`, `table`, `label`, `labelframe`.

12.1.6.1. Sintaxis

Each widget is described briefly, followed by the markup used, and a screen shot. All tags inside the main widget tag are optional.

12.1.6.2. General Notes

At the present time, both a tag-based and an attribute-based syntax are supported. For instance, the following XML fragments are treated identically:

```
<led halpin="my-led"/>
```

y

```
<led><halpin>"my-led"</halpin></led>
```

When the attribute-based syntax is used, the following rules are used to turn the attributes value into a Python value:

1. If the first character of the attribute is one of the following, it is evaluated as a Python expression: `{([”'` .
2. If the string is accepted by int(), the value is treated as an integer.
3. If the string is accepted by float(), the value is treated as floating-point.
4. Otherwise, the string is accepted as a string.

When the tag-based syntax is used, the text within the tag is always evaluated as a Python expression.

The examples below show a mix of formats.

Comentarios To add a comment use the xml syntax for a comment.

```
<!-- My Comment -->
```

Editing the XML file Edit the XML file with a text editor. In most cases you can right click on the file and select *open with text editor* or similar.

Colors

Colors can be specified using the X11 rgb colors by name *gray75* or hex *#0000ff*. A complete list is located here https://sedition.com/perl/rgb.html.

Common Colors (colors with numbers indicate shades of that color)

- white
- black
- blue and blue1 - 4
- cyan and cyan1 - 4
- green and green1 - 4
- yellow and yellow1 - 4
- red and red1 - 4
- purple and purple1 - 4
- gray and gray0 - 100

HAL Pins HAL pins provide a means to *connect* the widget to something. Once you create a HAL pin for your widget you can *connect* it to another HAL pin with a *net* command in a .hal file. For more information on the *net* command see the HAL Commands section.

12.1.6.3. Label

A label is a way to add text to your panel.

- *<label></label>* - creates a label.
- *<text>"text"</text>* - the text to put in your label, a blank label can be used as a spacer to align other objects.
- *("Helvetica",20)* - specify the font and size of the text.
- *<relief>FLAT</relief>* - specify the border around the label (*FLAT*, *RAISED*, *SUNKEN*) default is *FLAT*.
- *<bd>_n_</bd>* - where *n* is the border width when *RAISED* or *SUNKEN* borders are used.
- *<padx>_n_</padx>* - where *n* is the amount of extra horizontal extra space.
- *<pady>_n_</pady>* - where *n* is the amount of extra vertical extra space.

The label has an optional disable pin that is created when you add *<disable_pin>True</disable_pin>*.

```
<label>
    <text>"This is a Label:"</text>
    <font>("Helvetica",20)</font>
</label>
```

The above code produced this example:



Figura 12.3: Simple Label Example

12.1.6.4. Multi_Label

An extension of the text label.

Selectable text label, can display up to 6 label legends when associated bit pin is activated.

Attach each legend pin to a signal and get a descriptive label when the signal is TRUE.

If more than one legend pin is TRUE, the highest numbered *TRUE* legend will be displayed.

If a disable pin is created with *<disable_pin>True</disable_pin>* and that pin is set to true the label changes to a grayed out state.

```
<multilabel>
    <legends>["Label1", "Label2", "Label3", "Label4", "Label5", "Label6"]</legends>
    <font>("Helvetica",20)</font>
    <disable_pin>True</disable_pin>
</multilabel>
```

El ejemplo anterior creará los pines siguientes.

```
pyvcp.multilabel.0.disable
pyvcp.multilabel.0.legend0
pyvcp.multilabel.0.legend1
pyvcp.multilabel.0.legend2
pyvcp.multilabel.0.legend3
pyvcp.multilabel.0.legend4
pyvcp.multilabel.0.legend5
```

If you have more than one multilabel the pins created would increment the number like this *pyvcp.multilabel.*

12.1.6.5. LEDs

A LED is used to indicate the status of a *bit* halpin. The LED color will be on_color when the halpin is true, and off_color otherwise.

- *<led></led>* - makes a round LED
- *<rectled></rectled>* - makes a rectangle LED
- *<halpin>*name*</halpin>* - *name* of the pin, default is *led*.n, where *n* is an integer that is incremented for each LED.
- *<size>*n*</size>* - *n* is the size of the led in pixels, default is 20.
- *<on_color>*color*</on_color>* - sets the color of the LED to *color* when the pin is true. Default is *green*. See section on colors for more info.
- *<off_color>*color*</off_color>* - sets the color of the LED to *color* when the pin is false. Default is *red*.
- *<height>*n*</height>* - sets the height of the LED in pixels.
- *<width>*n*</width>* - sets the width of the LED in pixels.
- *<disable_pin>false</disable_pin>* - when true adds a disable pin to the led.
- *<disabled_color>*color*</disabled_color>* - sets the color of the LED to *color* when the pin is disabled.

Round LED

```
<led>
    <halpin>”my-led”</halpin>
    <size>50</size>
    <on_color>”green”</on_color>
    <off_color>”red”</off_color>
</led>
```

The above code produced this example:

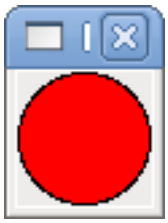


Figura 12.4: Round LED Example

Rectangle LED This is a variant of the *led* widget.

```
<vbox>
    <relief>Cresta</relief>
    <bd>6</bd>
    <rectled>
        <halpin>”mi-LED”</halpin>
        <height>”50”</height>
        <width>”100”</width>
        <on_color>”Verde”</on_color>
        <off_color>”Rojo”</off_color>
    </rectled>
</vbox>
```

The above code produced this example. Also showing a vertical box with relief.



Figura 12.5: Simple Rectangle LED Example

12.1.6.6. Buttons

A button is used to control a BIT pin. The pin will be set True when the button is pressed and held down, and will be set False when the button is released. Buttons can use the following optional options.

- <*padx*>n</*padx*> - where *n* is the amount of extra horizontal extra space.
- <*pady*>n</*pady*> - where *n* is the amount of extra vertical extra space.

- <*activebackground*>”color”</*activebackground*> - the cursor over color set to *color*.
- <*fg*>”color”</*fg*> - the foreground color set to *color*.
- <*bg*>”color”</*bg*> - the background color set to *color*.
- <*disable_pin*>*True*</*disable_pin*> - disable pin.

Text Button A text button controls a *bit* halpin. The halpin is false until the button is pressed then it is true. The button is a momentary button.

The text button has an optional disable pin that is created when you add <disable_pin>True</disable_pin>.

```
<button>
    <halpin>”ok-button”</halpin>
    <text>”OK”</text>
</button>
<button>
    <halpin>”abort-button”</halpin>
    <text>”Abort”</text>
</button>
```

The above code produced this example:



Figura 12.6: Simple Buttons Example

Checkbutton A checkbutton controls a bit halpin. The halpin will be set True when the button is checked, and false when the button is unchecked. The checkbutton is a toggle type button. The checkbuttons may be set initially as TRUE or FALSE the initval field A pin called changepin is also created automatically, which can toggle the Checkbutton via HAL, if the value linked is changed, to update the display remotely.



Figura 12.7: Unchecked button



Figura 12.8: Botón seleccionado

Checkbutton Code Example

```
<checkbutton>
    <halpin>”coolant-chkbtn”</halpin>
    <text>”Coolant”</text>
```

```
    <initval>1</initval>
</checkbutton>
<checkbutton>
    <halpin>"chip-chkbtn"</halpin>
    <text>"Chips    "</text>
    <initval>0</initval>
</checkbutton>
```

The above code produced this example:

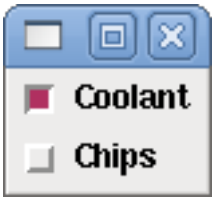


Figura 12.9: Simple Checkbutton Example

The coolant checkbutton is checked.
Notice the extra spaces in the "Chips" text to keep the checkbuttons aligned.

Radiobutton A radiobutton will set one of the halpins true. The other pins are set false. The initval field may be set to choose the default selection when the panel displays. Only one radio button may be set to TRUE (1) or only the highest number pin set TRUE will have that value.

```
<radiobutton>
    <choices>["one","two","three"]</choices>
    <halpin>"my-radio"</halpin>
    <initval>0</initval>
</radiobutton>
```

The above code produced this example:

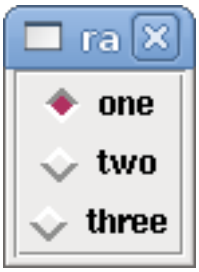


Figura 12.10: Simple Radiobutton Example

Note that the HAL pins in the example above will be named my-radio.one, my-radio.two, and my-radio.three. In the image above, *one* is the selected value. Use the tag `<orient>HORIZONTAL</orient>` to display horizontally.

12.1.6.7. Number Displays

Number displays can use the following formatting options

- `<font>("Font Name",n)</font>`, where *n* is the font size.

- <width>_n_</width>, where *n* is the overall width of the space used.
- <justify>_pos_</justify>, where *pos* is LEFT, CENTER, or RIGHT (doesn't work).
- <padx>_n_</padx>, where *n* is the amount of extra horizontal extra space.
- <pady>_n_</pady>, where *n* is the amount of extra vertical extra space.

Number The number widget displays the value of a float signal.

```
<number>
    <halpin>"my-number"</halpin>
    <font>("Helvetica",24)</font>
    <format>"+4.4f"</format>
</number>
```

The above code produced this example:



Figura 12.11: Simple Number Example

- *<font>* - is a Tkinter font type and size specification. One font that will show up to at least size 200 is *courier 10 pitch*, so for a really big Number widget you could specify:

```
<font>("courier 10 pitch",100)</font>
```

- *<format>* - is a *C-style* format specified that determines how the number is displayed.

s32 Number The s32 number widget displays the value of a s32 number. The syntax is the same as *number* except the name which is <s32>. Make sure the width is wide enough to cover the largest number you expect to use.

```
<s32>
    <halpin>"my-number"</halpin>
    <font>("Helvetica",24)</font>
    <format>"6d"</format>
    <width>6</width>
</s32>
```

The above code produced this example:

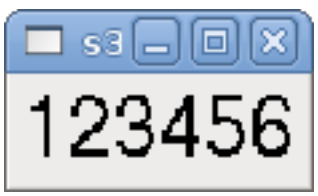


Figura 12.12: Simple s32 Number Example

u32 Number The u32 number widget displays the value of a u32 number. The syntax is the same as *number* except the name which is <u32>.

Bar A bar widget displays the value of a FLOAT signal both graphically using a bar display and numerically. The color of the bar can be set as one color throughout its range (default using fillcolor) or set to change color, dependent upon the value of the halpin (range1, range2 range3 must all be set, if you only want 2 ranges, set 2 of them to the same color).

- <halpin>”my-bar”</halpin> (text), derives and sets the pin name: pyvcp.my-bar.
- <min_>0</min_> (number), sets the minimum scale.
- <max_>140</max_> (number), sets the maximum scale.
- <format>”3.1f”</format> (text), sets the number format using Python number formatting.
- <bgcolor>”grey”</bgcolor> (text), sets the background color.
- <fillcolor>”red”</fillcolor> (text), sets the fill color.
- <range1>0,100,”green”</range1> (number, number, text), sets the first range and color.
- <range2>101,135,”orange”</range2> (number, number, text), sets the first range and color.
- <range3>136, 150,”red”</range3> number, number, text), sets the first range and color.
- <canvas_width>200</canvas_width> (number), sets the overall width.
- <canvas_height>50</canvas_height> (number), sets the overall height.
- <bar_height>30</bar_height> (number), sets the bar height, must be less than canvas_height.
- <bar_width>150</bar_width> (number), sets the bar width, must be less than canvas_width.

```
<bar>
    <halpin>”my-bar”</halpin>
    <min_>0</min_>
    <max_>123</max_>
    <format>”3.1f”</format>
    <bgcolor>”grey”</bgcolor>
    <fillcolor>”red”</fillcolor>
    <range1>0,100,”green”</range1>
    <range2>101,135,”orange”</range2>
    <range3>136, 150,”red”</range3>
    <canvas_width>200</canvas_width>
    <canvas_height>50</canvas_height>
    <bar_height>30</bar_height>
    <bar_width>150</bar_width>
</bar>
```

The above code produced this example:



Figura 12.13: Simple Bar Example

Meter Meter displays the value of a FLOAT signal using a traditional dial indicator.

```
<meter>
    <halpin>"mymeter"</halpin>
    <text>"Battery"</text>
    <subtext>"Volts"</subtext>
    <size>250</size>
    <min_>0</min_>
    <max_>15.5</max_>
    <majorscale>1</majorscale>
    <minorscale>0.2</minorscale>
    <region1>(14.5,15.5,"yellow")</region1>
    <region2>(12,14.5,"green")</region2>
    <region3>(0,12,"red")</region3>
</meter>
```

The above code produced this example:

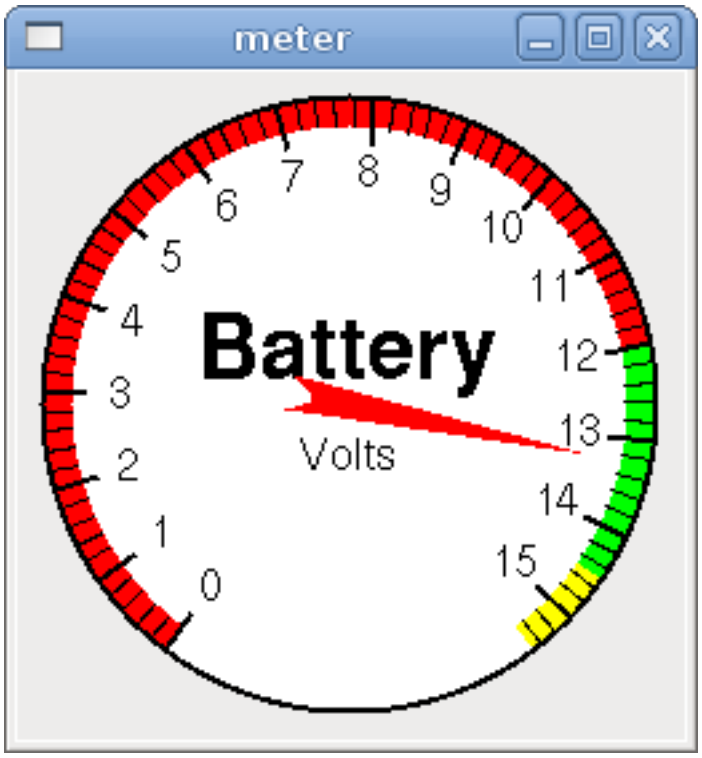


Figura 12.14: Simple Meter Example

12.1.6.8. Number Inputs

Spinbox A spinbox controls a FLOAT pin. You increase or decrease the value of the pin by either pressing on the arrows, or pointing at the spinbox and rolling your mouse-wheel. If the param_pin field is set TRUE(1), a pin will be created that can be used to set the spinbox to an initial value and to remotely alter its value without HID input.

```
<spinbox>
    <halpin>"my-spinbox"</halpin>
    <min_>-12</min_>
    <max_>33</max_>
    <initval>0</initval>
    <resolution>0.1</resolution>
```

```
    <format>”2.3f”</format>
    <font>(”Arial”,30)</font>
    <param_pin>1</param_pin>
</spinbox>
```

The above code produced this example:



Figura 12.15: Simple Spinbox Example

Scale A scale controls a float or a s32 pin. You increase or decrease the value of the pin be either dragging the slider, or pointing at the scale and rolling your mouse-wheel. The *halpin* will have both *-f* and *-i* added to it to form the float and s32 pins. Width is the width of the slider in vertical and the height of the slider in horizontal orientation. If the param_pin field is set TRUE(1), a pin will be created that can be used to set the spinbox to an initial value and to remotely alter its value without HID input.

```
<scale>
    <font>(”Helvetica”,16)</font>
    <width>”25”</width>
    <halpin>”my-hscale”</halpin>
    <resolution>0.1</resolution>
    <orient>HORIZONTAL</orient>
    <initval>-15</initval>
    <min_>-33</min_>
    <max_>26</max_>
    <param_pin>1</param_pin>
</scale>
<scale>
    <font>(”Helvetica”,16)</font>
    <width>”50”</width>
    <halpin>”my-vscale”</halpin>
    <resolution>1</resolution>
    <orient>VERTICAL</orient>
    <min_>100</min_>
    <max_>0</max_>
    <param_pin>1</param_pin>
</scale>
```

The above code produced this example:

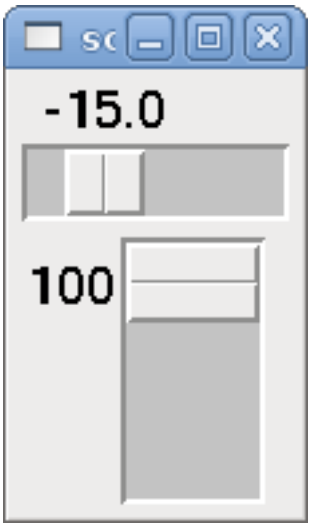


Figura 12.16: Simple Scale Example

nota
Note that by default it is ”min” which is displayed even if it is greater than ”max”, unless ”min” is negative.

Dial The Dial outputs a HAL float and reacts to both mouse wheel and dragging. Double left click to increase the resolution and double right click to reduce the resolution by one digit. The output is capped by the min and max values. The <cpr> is how many tick marks are on the outside of the ring (beware of high numbers). If the param_pin field is set TRUE(1), a pin will be created that can be used to set the spinbox to an initial value and to remotely alter its value without HID input.

```
<dial>
    <size>200</size>
    <cpr>100</cpr>
    <min_>-15</min_>
    <max_>15</max_>
    <text>”Dial”</text>
    <initval>0</initval>
    <resolution>0.001</resolution>
    <halpin>”anaout”</halpin>
    <dialcolor>”yellow”</dialcolor>
    <edgecolor>”green”</edgecolor>
    <dotcolor>”black”</dotcolor>
    <param_pin>1</param_pin>
</dial>
```

The above code produced this example:

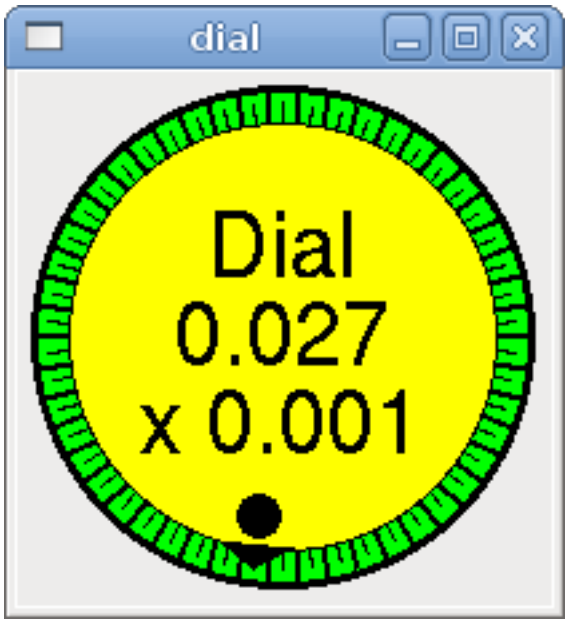


Figura 12.17: Simple Dial Example

Perilla de trote Jogwheel mimics a real jogwheel by outputting a FLOAT pin which counts up or down as the wheel is turned, either by dragging in a circular motion, or by rolling the mouse-wheel.

Etiquetas opcionales: * *<text>”My Text”</text>* displays text * *<bgcolor>”grey”</bgcolor>* *<fillcolor>”green”</fillcolor>* background & active colors * *<scale_pin>1</scale_pin>* creates scale text and a `FLOAT.scale` pin to display jog scale * *<clear_pin>1</clear_pin>* creates `DRO` and a `BIT.reset` pin to reset DRO. Needs scale_pin for scaled DRO. Shift+click resets DRO also

```
<jogwheel>
    <halpin>”my-wheel”</halpin>
    <cpr>45</cpr>
    <size>250</size>
</jogwheel>
```

The above code produced this example:

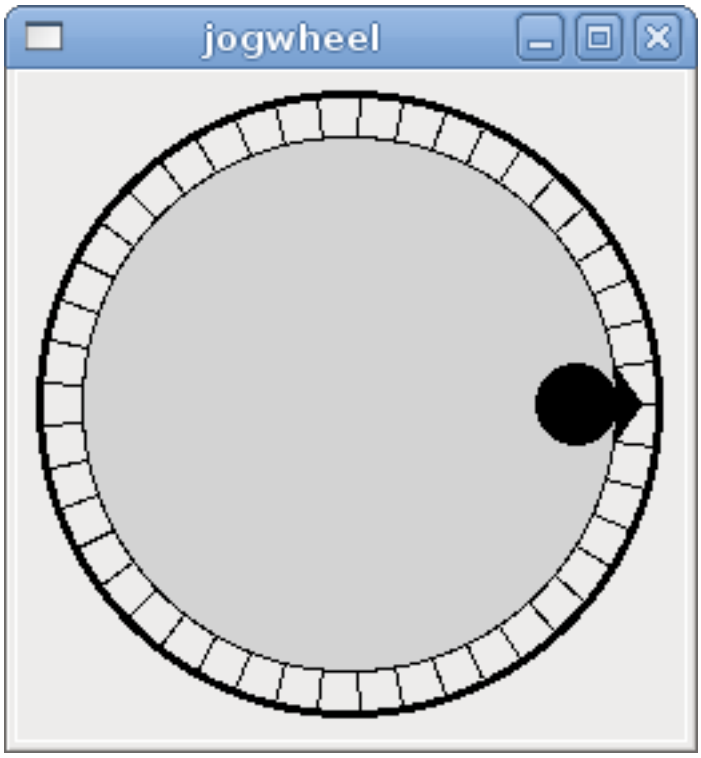


Figura 12.18: Simple Jogwheel Example

12.1.6.9. Images

Image displays use only .gif image format. All of the images must be the same size. The images must be in the same directory as your INI file (or in the current directory if running from the command line with halrun/halcmd).

Image Bit The *image_bit* toggles between two images by setting the halpin to true or false.

```
<image name='fwd' file='fwd.gif'/>
<image name='rev' file='rev.gif'/>
<vbox>
    <image_bit halpin='selectimage' images='fwd rev'/>
</vbox>
```

This example was produced from the above code. Using the two image files fwd.gif and rev.gif. FWD is displayed when *selectimage* is false and REV is displayed when *selectimage* is true.



Figura 12.19: Selectimage False Example



Figura 12.20: Selectimage True Example

Image u32 The *image_u32* is the same as *image_bit*, except you have essentially an unlimited number of images and you *select* the image by setting the halpin to a integer value with 0 for the first image in the images list and 1 for the second image, etc.

```
<image name='stb' file='stb.gif'/>
<image name='fwd' file='fwd.gif'/>
<image name='rev' file='rev.gif'/>
<vbox>
    <image_u32 halpin='selectimage' images='stb fwd rev'/>
</vbox>
```

The above code produced the following example by adding the stb.gif image.



Figura 12.21: Simple image_u32 Example with halpin=0



Figura 12.22: Simple image_u32 Example withhalpin=1



Figura 12.23: Simple image_u32 Example withhalpin=2

Notice that the default is the min even though it is set higher than max unless there is a negative min.

12.1.6.10. Containers

Containers are widgets that contain other widgets. Containers are used to group other widgets.

Borders Container borders are specified with two tags used together. The <relief> tag specifies the type of border and the <bd> specifies the width of the border.

<relief>_type_</relief>
: Where *type* is FLAT, SUNKEN, RAISED, GROOVE, or RIDGE.

<bd>_n_</bd>
: Where *n* is the width of the border.

```
<hbox>
    <button>
        <relief>FLAT</relief>
        <text>"FLAT"</text>
        <bd>3</bd>
    </button>
    <button>
        <relief>SUNKEN</relief>
        <text>"SUNKEN"</text>
        <bd>3</bd>
    </button>
    <button>
        <relief>RAISED</relief>
        <text>"RAISED"</text>
        <bd>3</bd>
    </button>
    <button>
        <relief>GROOVE</relief>
        <text>"GROOVE"</text>
        <bd>3</bd>
    </button>
    <button>
        <relief>RIDGE</relief>
        <text>"RIDGE"</text>
        <bd>3</bd>
    </button>
</hbox>
```

The above code produced this example:



Figura 12.24: Containers Borders Showcase

Fill Container fill are specified with the <*boxfill fill=””/*> tag. Valid entries are none, x, y and both. The x fill is a horizontal fill and the y fill is a vertical fill

<boxfill fill =”style”/>
Where *style* is none, x, y, or both. Default is x for Vbox and y for Hbox.

Anchor Container anchors are specified with the <boxanchor anchor=””/> tag. The anchor specifies where to position each slave in its parcel. Valid entries are center, n, s, e, w, for center, north, south, east and west. Combinations like sw, se, nw and ne are also valid.

<boxanchor anchor=”position”/>
Where *position* is center, n, s, e, w, ne, nw, se or sw. Default is center.

Expand Container expand is specified with the boolean <boxexpand expand=””/> tag. Valid entries are ”yes”, ”no”.

<boxexpand expand=”boolean”/>
Where *boolean* is either ”yes” or ”no”. Default is yes.

Hbox Use an Hbox when you want to stack widgets horizontally next to each other.

```
<hbox>
    <relief>RIDGE</relief>
    <bd>6</bd>
    <label><text>”a hbox:”</text></label>
    <led></led>
    <number></number>
    <bar></bar>
</hbox>
```

The above code produced this example:

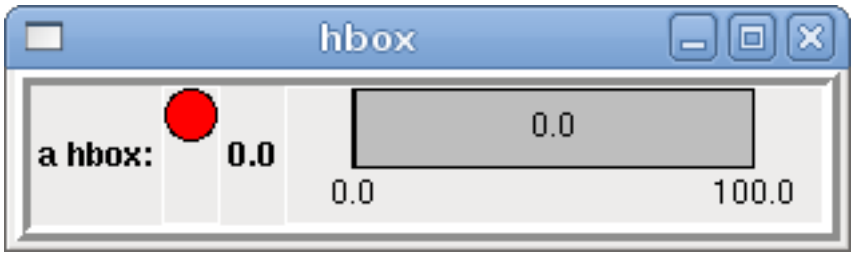


Figura 12.25: Simple hbox Example

Inside an Hbox, you can use the <*boxfill fill=””/*>, <*boxanchor anchor=””/*>, and <*boxexpand expand=””/*> tags to choose how items in the box behave when the window is re-sized. The default is *fill=”y”*, *anchor=”center”*, *expand=”yes”* for an Hbox.

Vbox Use a Vbox when you want to stack widgets vertically on top of each other.

```
<vbox>
    <relief>RIDGE</relief>
    <bd>6</bd>
    <label><text>"a vbox:"</text></label>
    <led></led>
    <number></number>
    <bar></bar>
</vbox>
```

The above code produced this example:

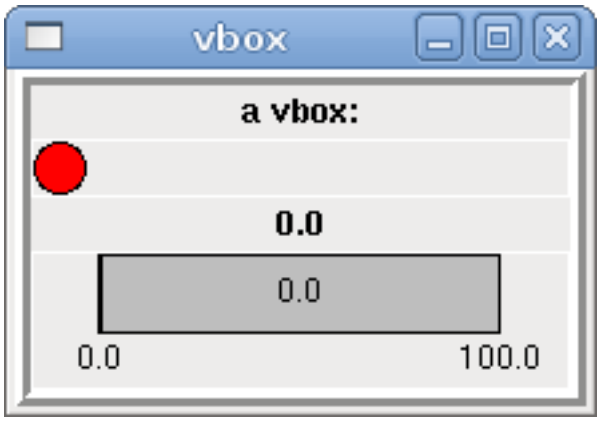


Figura 12.26: Simple vbox Example

Inside a Vbox, you can use the *<boxfill fill=""/>*, *<boxanchor anchor=""/>*, and *<boxexpand expand=""/>* tags to choose how items in the box behave when the window is re-sized. The default is *fill="x"*, *anchor="center"*, *expand="yes"* for a Vbox.

Labelframe A labelframe is a frame with a groove and a label at the upper-left corner.

```
<labelframe text="Label: Leds groupées">
```

```
<labelframe text="Group Title">
    <font>("Helvetica",16)</font>
    <hbox>
    <led/>
    <led/>
    </hbox>
</labelframe>
```

The above code produced this example:



Figura 12.27: Simple labelframe Example

Table A table is a container that allows layout in a grid of rows and columns. Each row is started by a *<tablerow/>* tag. A contained widget may span rows or columns through the use of the *<tablespan*

rows= cols=/> tag. The sides of the cells to which the contained widgets ”stick” may be set through the use of the *<tablesticky sticky=/>* tag. A table expands on its flexible rows and columns.

Table Code Example

```
<table flexible_rows=”[2]” flexible_columns=”[1,4]”>
<tablesticky sticky=”new”/>
<tablerow/>
    <label>
        <text>” A (cell 1,1) ”</text>
        <relief>RIDGE</relief>
        <bd>3</bd>
    </label>
    <label text=”B (cell 1,2)”/>
    <tablespan columns=”2”/>
    <label text=”C, D (cells 1,3 and 1,4)”/>
<tablerow/>
    <label text=”E (cell 2,1)”/>
    <tablesticky sticky=”nsew”/>
    <tablespan rows=”2”/>
    <label text=”’spans\n2 rows’”/>
    <tablesticky sticky=”new”/>
    <label text=”G (cell 2,3)”/>
    <label text=”H (cell 2,4)”/>
<tablerow/>
    <label text=”J (cell 3,1)”/>
    <label text=”K (cell 3,2)”/>
    <u32 halpin=”test”/>
</table>
```

The above code produced this example:

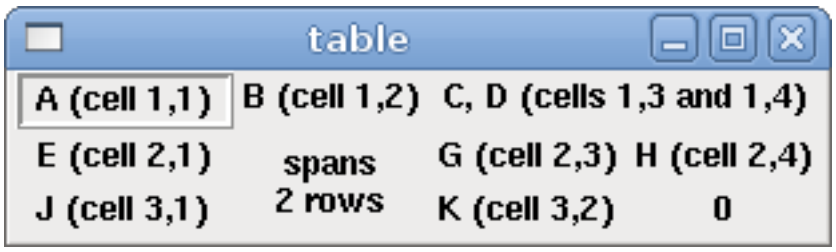


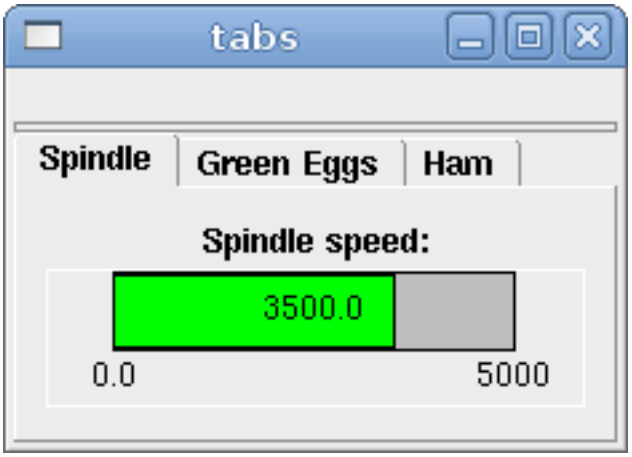
Figura 12.28: Table Example

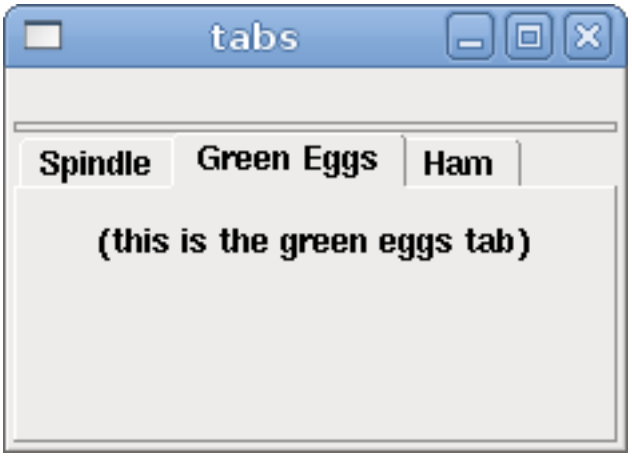
Tabs A tabbed interface can save quite a bit of space.

```
<tabs>
    <names> [”spindle”,”green eggs”]</names>
</tabs>
<tabs>
    <names>[”Spindle”, ”Green Eggs”, ”Ham”]</names>
    <vbox>
        <label>
            <text>”Spindle speed:”</text>
        </label>
        <bar>
            <halpin>”spindle-speed”</halpin>
            <max_>5000</max_>
        </bar>
    </vbox>
    <vbox>
```

```
        <label>
            <text>”(this is the green eggs tab)”</text>
        </label>
    </vbox>
    <vbox>
        <label>
            <text>”(this tab has nothing on it)”</text>
        </label>
    </vbox>
</tabs>
```

The above code produced this example showing each tab selected.





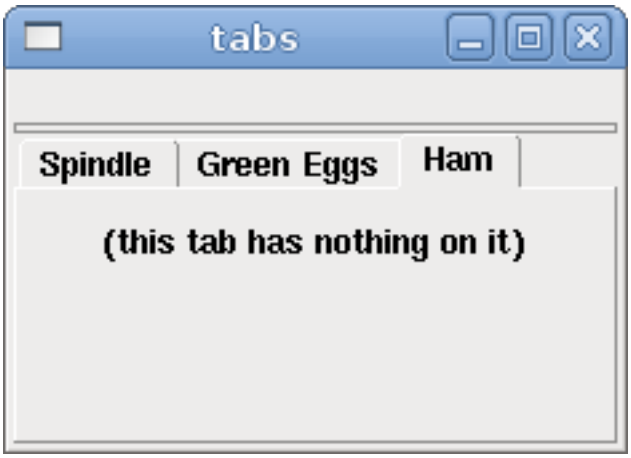


Figura 12.29: Simple Tabs Example

12.2. PyVCP Examples

12.2.1. AXIS

To create a PyVCP panel to use with the AXIS interface that is attached to the right of AXIS you need to do the following basic things.

- Create an XML file that contains your panel description and put it in your config directory.
- Add the PyVCP entry to the [DISPLAY] section of the INI file with your XML file name.
- Add the POSTGUI_HALFILE entry to the [HAL] section of the INI file with the name of your postgui HAL file name.
- Add the links to HAL pins for your panel in the postgui.hal file to *connect* your PyVCP panel to LinuxCNC.

12.2.2. Floating Panels

To create floating PyVCP panels that can be used with any interface you need to do the following basic things.

- Create an XML file that contains your panel description and put it in your config directory.
- Add a loadusr line to your HAL file to load each panel.
- Add the links to HAL pins for your panel in the postgui.hal file to *connect* your PyVCP panel to LinuxCNC.

The following is an example of a loadusr command to load two PyVCP panels and name each one so the connection names in HAL will be known.

```
loadusr -Wn btnpanel pyvcp -c btnpanel panel1.xml
loadusr -Wn sppanel pyvcp -c sppanel panel2.xml
```

The -Wn makes HAL *Wait for name* to be loaded before proceeding.

The pyvcp -c makes PyVCP name the panel.

The HAL pins from panel1.xml will be named btnpanel.<_pin name_>.

The HAL pins from panel2.xml will be named sppanel.<_pin name_>.

Make sure the loadusr line is before any nets that make use of the PyVCP pins.

12.2.3. Jog Buttons Example

In this example we will create a PyVCP panel with jog buttons for X, Y, and Z. This configuration will be built upon a StepConf Wizard generated configuration. First we run the StepConf Wizard and configure our machine, on the Advanced Configuration Options page we then make a couple of selections to add a blank PyVCP panel as shown in the following figure. For this example we named the configuration *pyvcp_xyz* on the Basic Machine Information page of the StepConf Wizard.

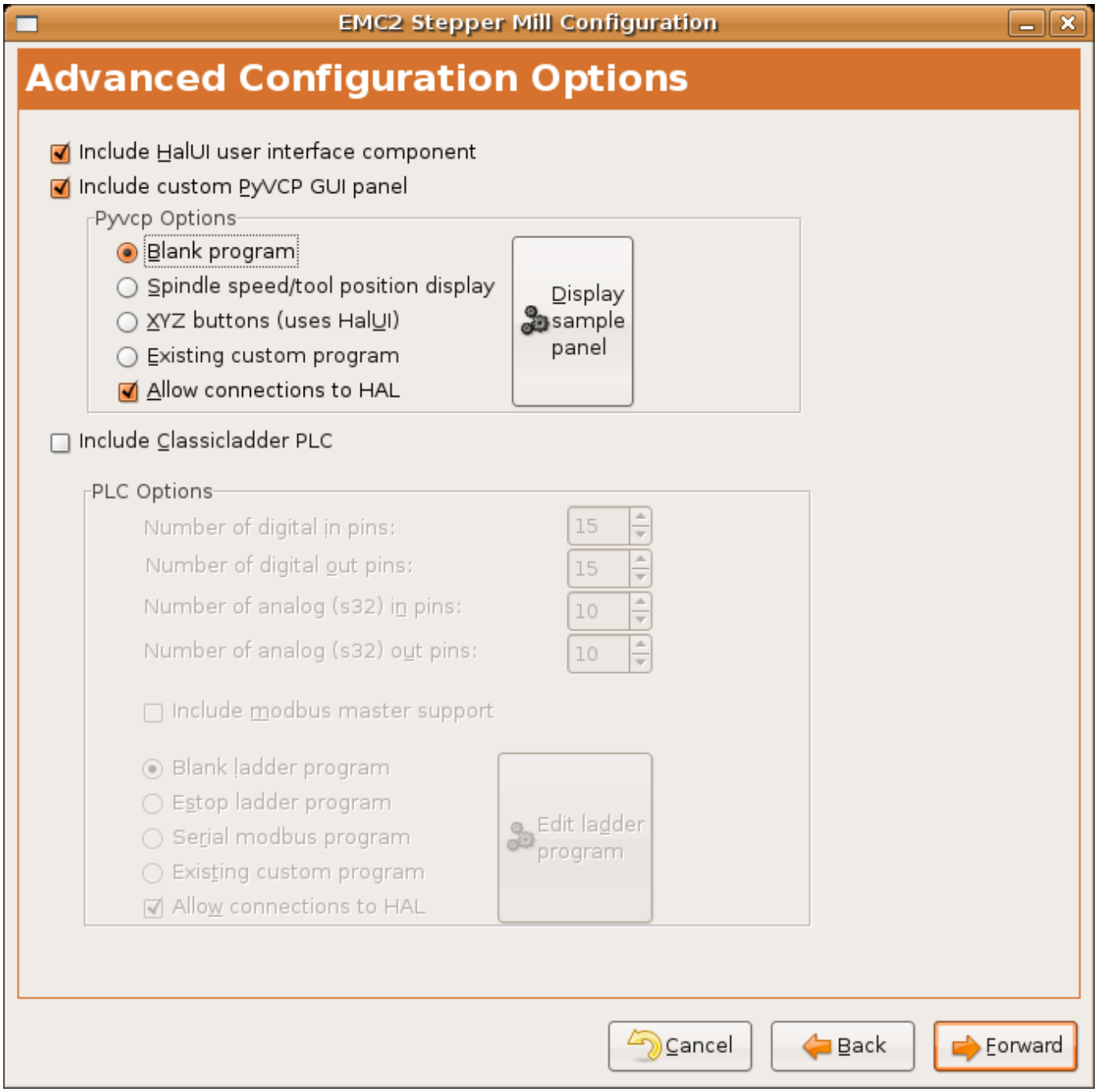


Figura 12.30: XYZ Wizard Configuration

The StepConf Wizard will create several files and place them in the `linuxcnc/configs/pyvcp_xyz` directory. If you left the create link checked you will have a link to those files on your desktop.

12.2.3.1. Create the Widgets

Open up the custompanel.xml file by right clicking on it and selecting *open with text editor*. Between the <pyvcp></pyvcp> tags we will add the widgets for our panel.

Look in the PyVCP Widgets Reference section of the manual for more detailed information on each widget documentation des widgets.

In your custompanel.xml file we will add the description of the widgets.

```
<pyvcp>
  <labelframe text="Jog Buttons">
    <font>("Helvetica",16)</font>

      <!-- the X jog buttons -->
      <hbox>
      <relief>RAISED</relief>
      <bd>3</bd>
      <button>
        <font>("Helvetica",20)</font>
        <width>3</width>
        <halpin>"x-plus"</halpin>
        <text>"X+"</text>
      </button>
      <button>
        <font>("Helvetica",20)</font>
        <width>3</width>
        <halpin>"x-minus"</halpin>
        <text>"X-"</text>
      </button>
      </hbox>

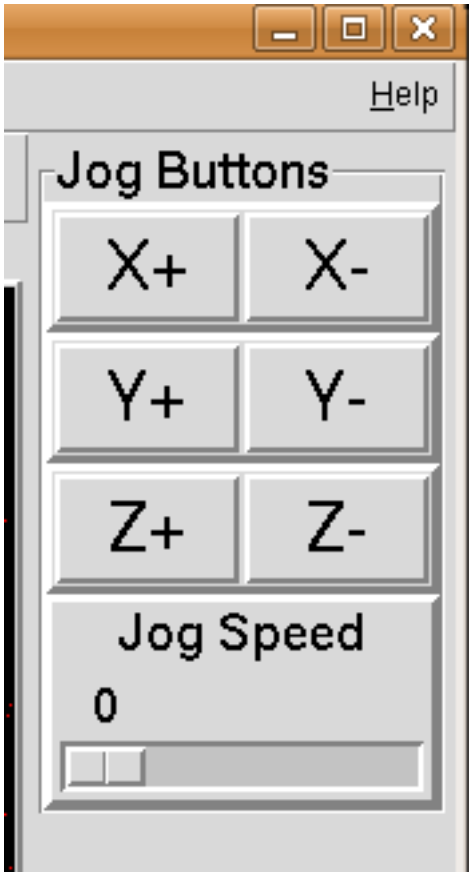
      <!-- the Y jog buttons -->
      <hbox>
      <relief>RAISED</relief>
      <bd>3</bd>
      <button>
        <font>("Helvetica",20)</font>
        <width>3</width>
        <halpin>"y-plus"</halpin>
        <text>"Y+"</text>
      </button>
      <button>
        <font>("Helvetica",20)</font>
        <width>3</width>
        <halpin>"y-minus"</halpin>
        <text>"Y-"</text>
      </button>
      </hbox>

      <!-- the Z jog buttons -->
      <hbox>
      <relief>RAISED</relief>
      <bd>3</bd>
      <button>
        <font>("Helvetica",20)</font>
        <width>3</width>
        <halpin>"z-plus"</halpin>
        <text>"Z+"</text>
      </button>
      <button>
        <font>("Helvetica",20)</font>
        <width>3</width>
        <halpin>"z-minus"</halpin>
        <text>"Z-"</text>
      </button>
      </hbox>

      <!-- the jog speed slider -->
      <vbox>
```

```
      <relief>RAISED</relief>
      <bd>3</bd>
      <label>
        <text>”Jog Speed”</text>
        <font>(”Helvetica”,16)</font>
      </label>
      <scale>
        <font>(”Helvetica”,14)</font>
        <halpin>”jog-speed”</halpin>
        <resolution>1</resolution>
        <orient>HORIZONTAL</orient>
        <min_>0</min_>
        <max_>80</max_>
      </scale>
      </vbox>
  </labelframe>
</pyvcp>
```

After adding the above you now will have a PyVCP panel that looks like the following attached to the right side of AXIS. It looks nice but it does not do anything until you *connect* the buttons to halui. If you get an error when you try and run scroll down to the bottom of the pop up window and usually the error is a spelling or syntax error and it will be there.



12.2.3.2. Hacer conexiones

To make the connections needed open up your custom_postgui.hal file and add the following.

```
# connect the X PyVCP buttons
net my-jogxminus halui.axis.x.minus <= pyvcp.x-minus
```

```
net my-jogxplus halui.axis.x.plus <= pyvcp.x-plus

# connect the Y PyVCP buttons
net my-jogyminus halui.axis.y.minus <= pyvcp.y-minus
net my-jogyplus halui.axis.y.plus <= pyvcp.y-plus

# connect the Z PyVCP buttons
net my-jogzminus halui.axis.z.minus <= pyvcp.z-minus
net my-jogzplus halui.axis.z.plus <= pyvcp.z-plus

# connect the PyVCP jog speed slider
net my-jogspeed halui.axis.jog-speed <= pyvcp.jog-speed-f
```

After resetting the E-Stop and putting it into jog mode and moving the jog speed slider in the PyVCP panel to a value greater than zero the PyVCP jog buttons should work. You can not jog when running a G-code file or while paused or while the MDI tab is selected.

12.2.4. Port Tester

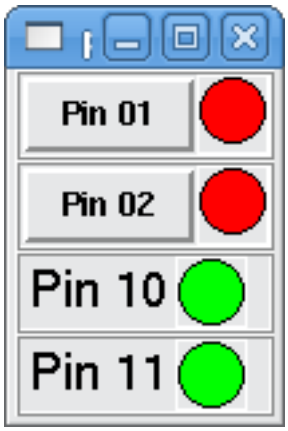
This example shows you how to make a simple parallel port tester using PyVCP and HAL.

First create the ptest.xml file with the following code to create the panel description.

```
<!-- Test panel for the parallel port cfg for out -->
<pyvcp>
  <hbox>
    <relief>RIDGE</relief>
    <bd>2</bd>
    <button>
      <halpin>”btn01”</halpin>
      <text>”Pin 01”</text>
    </button>
    <led>
      <halpin>”led-01”</halpin>
      <size>25</size>
      <on_color>”green”</on_color>
      <off_color>”red”</off_color>
    </led>
  </hbox>
  <hbox>
    <relief>RIDGE</relief>
    <bd>2</bd>
    <button>
      <halpin>”btn02”</halpin>
      <text>”Pin 02”</text>
    </button>
    <led>
      <halpin>”led-02”</halpin>
      <size>25</size>
      <on_color>”green”</on_color>
      <off_color>”red”</off_color>
    </led>
  </hbox>
  <hbox>
    <relief>RIDGE</relief>
    <bd>2</bd>
    <label>
      <text>”Pin 10”</text>
      <font>(”Helvetica”,14)</font>
    </label>
```

```
    <led>
      <halpin>”led-10”</halpin>
      <size>25</size>
      <on_color>”green”</on_color>
      <off_color>”red”</off_color>
    </led>
  </hbox>
  <hbox>
    <relief>RIDGE</relief>
    <bd>2</bd>
    <label>
      <text>”Pin 11”</text>
      <font>(”Helvetica”,14)</font>
    </label>
    <led>
      <halpin>”led-11”</halpin>
      <size>25</size>
      <on_color>”green”</on_color>
      <off_color>”red”</off_color>
    </led>
  </hbox>
</pyvcp>
```

This will create the following floating panel which contains a couple of in pins and a couple of out pins.



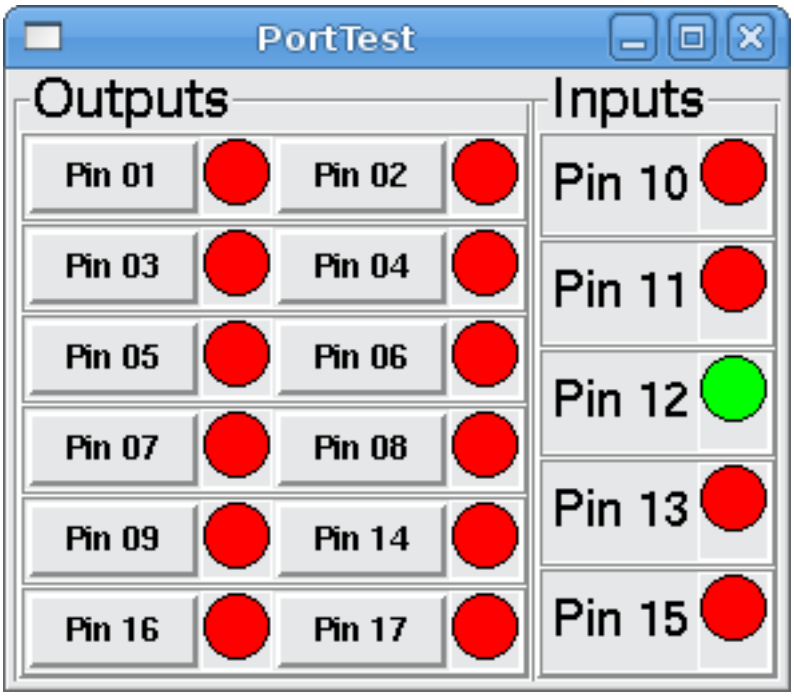
To run the HAL commands that we need to get everything up and running we put the following in our ptest.hal file.

```
loadrt hal_parport cfg=”0x378 out”
loadusr -Wn ptest pyvcp -c ptest ptest.xml
loadrt threads name1=porttest period1=1000000
addf parport.0.read porttest
addf parport.0.write porttest
net pin01 ptest.btn01 parport.0.pin-01-out ptest.led-01
net pin02 ptest.btn02 parport.0.pin-02-out ptest.led-02
net pin10 parport.0.pin-10-in ptest.led-10
net pin11 parport.0.pin-11-in ptest.led-11
start
```

To run the HAL file we use the following command from a terminal window.

```
~$ halrun -I -f ptest.hal
```

The following figure shows what a complete panel might look like.



To add the rest of the parallel port pins just modify the XML and HAL files.

To show the pins after running the HAL script use the following command at the halcmd prompt:

```
halcmd: show pin
Component Pins:
Owner Type  Dir Value  Name
    2 bit   IN  FALSE  parport.0.pin-01-out <== pin01
    2 bit   IN  FALSE  parport.0.pin-02-out <== pin02
    2 bit   IN  FALSE  parport.0.pin-03-out
    2 bit   IN  FALSE  parport.0.pin-04-out
    2 bit   IN  FALSE  parport.0.pin-05-out
    2 bit   IN  FALSE  parport.0.pin-06-out
    2 bit   IN  FALSE  parport.0.pin-07-out
    2 bit   IN  FALSE  parport.0.pin-08-out
    2 bit   IN  FALSE  parport.0.pin-09-out
    2 bit   OUT TRUE   parport.0.pin-10-in ==> pin10
    2 bit   OUT FALSE  parport.0.pin-10-in-not
    2 bit   OUT TRUE   parport.0.pin-11-in ==> pin11
    2 bit   OUT FALSE  parport.0.pin-11-in-not
    2 bit   OUT TRUE   parport.0.pin-12-in
    2 bit   OUT FALSE  parport.0.pin-12-in-not
    2 bit   OUT TRUE   parport.0.pin-13-in
    2 bit   OUT FALSE  parport.0.pin-13-in-not
    2 bit   IN  FALSE  parport.0.pin-14-out
    2 bit   OUT TRUE   parport.0.pin-15-in
    2 bit   OUT FALSE  parport.0.pin-15-in-not
    2 bit   IN  FALSE  parport.0.pin-16-out
    2 bit   IN  FALSE  parport.0.pin-17-out
    4 bit   OUT FALSE  ptest.btn01 ==> pin01
    4 bit   OUT FALSE  ptest.btn02 ==> pin02
    4 bit   IN  FALSE  ptest.led-01 <== pin01
    4 bit   IN  FALSE  ptest.led-02 <== pin02
    4 bit   IN  TRUE   ptest.led-10 <== pin10
    4 bit   IN  TRUE   ptest.led-11 <== pin11
```

This will show you what pins are IN and what pins are OUT as well as any connections.

12.2.5. GS2 RPM Meter

The following example uses the Automation Direct GS2 VDF driver and displays the RPM and other info in a PyVCP panel. This example is based on the GS2 example in the Hardware Examples section this manual.

12.2.5.1. The Panel

Para crear el panel agregamos lo siguiente al archivo XML.

```
<pyvcp>

  <!-- the RPM meter -->
  <hbox>
    <relief>RAISED</relief>
    <bd>3</bd>
    <meter>
      <halpin>”spindle_rpm”</halpin>
      <text>”Spindle”</text>
      <subtext>”RPM”</subtext>
      <size>200</size>
      <min_>0</min_>
      <max_>3000</max_>
      <majorscale>500</majorscale>
      <minorscale>100</minorscale>
      <region1>0,10,”yellow”</region1>
    </meter>
  </hbox>

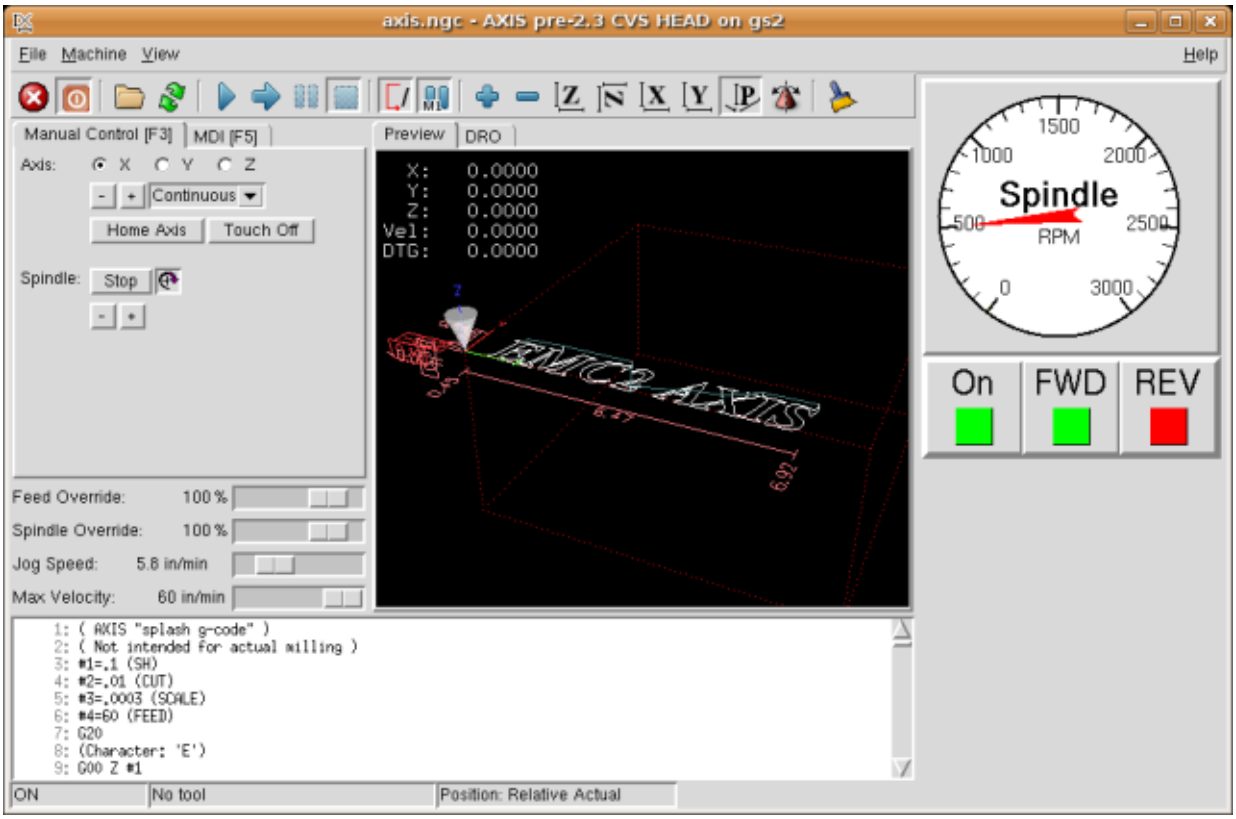
  <!-- the On Led -->
  <hbox>
  <relief>RAISED</relief>
  <bd>3</bd>
  <vbox>
  <relief>RAISED</relief>
  <bd>2</bd>
  <label>
  <text>”On”</text>
  <font>(”Helvetica”,18)</font>
  </label>
  <width>5</width>
   <hbox>
  <label width=”2”/> <!-- used to center the led -->
  <rectled>
  <halpin>”on-led”</halpin>
  <height>”30”</height>
  <width>”30”</width>
  <on_color>”green”</on_color>
  <off_color>”red”</off_color>
  </rectled>
  </hbox>
  </vbox>

  <!-- the FWD Led -->
  <vbox>
    <relief>RAISED</relief>
    <bd>2</bd>
    <label>
      <text>”FWD”</text>
      <font>(”Helvetica”,18)</font>
```

```
      <width>5</width>
    </label>
   <label width="2"/>
    <rectled>
      <halpin>"fwd-led"</halpin>
      <height>"30"</height>
      <width>"30"</width>
      <on_color>"green"</on_color>
      <off_color>"red"</off_color>
    </rectled>
  </vbox>

  <!-- the REV Led -->
  <vbox>
  <relief>RAISED</relief>
  <bd>2</bd>
    <label>
      <text>"REV"</text>
      <font>("Helvetica",18)</font>
       <width>5</width>
    </label>
    <label width="2"/>
    <rectled>
      <halpin>"rev-led"</halpin>
      <height>"30"</height>
      <width>"30"</width>
      <on_color>"red"</on_color>
      <off_color>"green"</off_color>
    </rectled>
  </vbox>
  </hbox>
</pyvcp>
```

The above gives us a PyVCP panel that looks like the following.



12.2.5.2. The Connections

To make it work we add the following code to the custom_postgui.hal file.

```
# display the rpm based on freq * rpm per hz
loadrt mult2
addf mult2.0 servo-thread
setp mult2.0.in1 28.75
net cypher_speed mult2.0.in0 <= spindle-vfd.frequency-out
net speed_out pyvcp.spindle_rpm <= mult2.0.out

# run led
net gs2-run => pyvcp.on-led

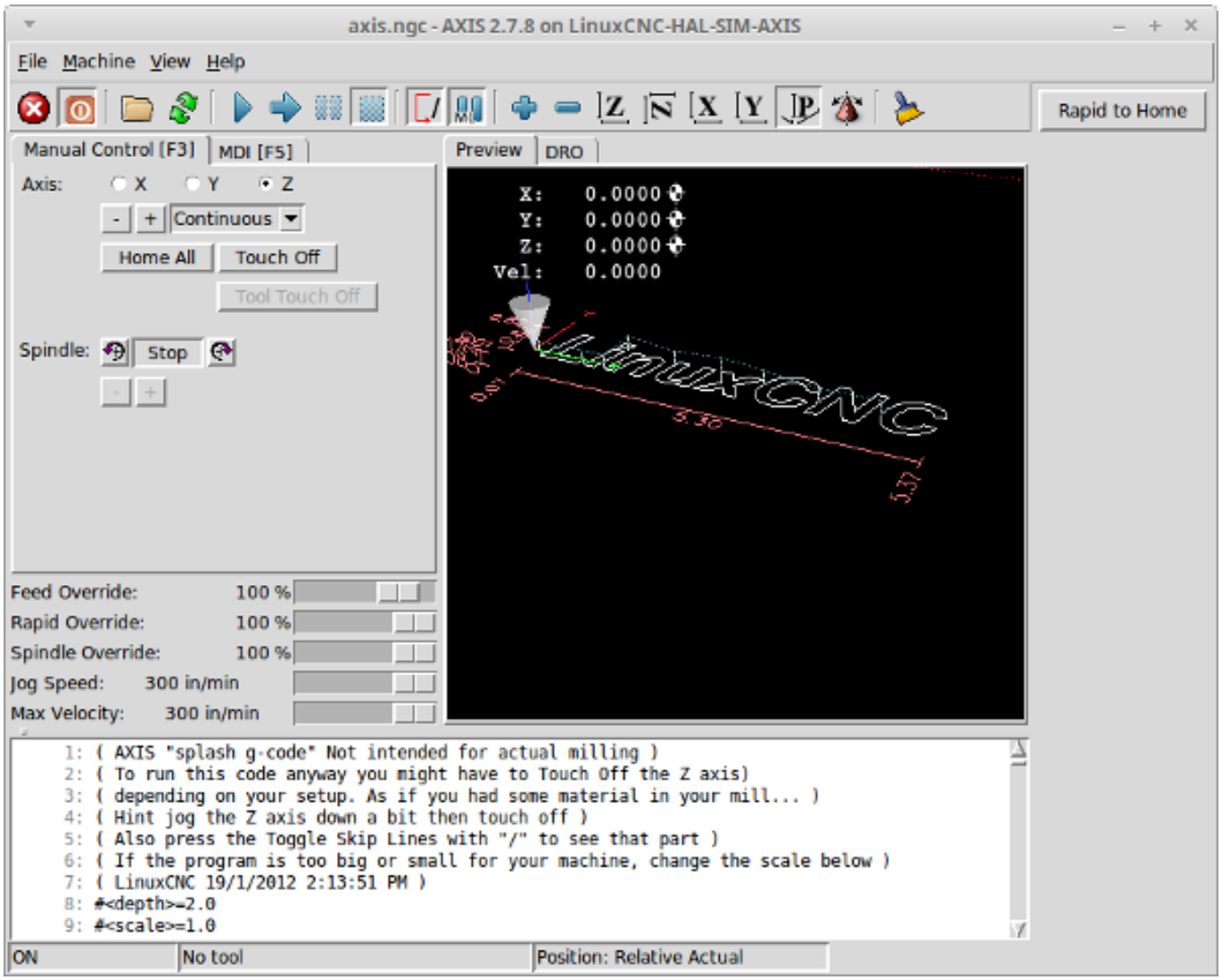
# fwd led
net gs2-fwd => pyvcp.fwd-led

# rev led
net running-rev spindle-vfd.spindle-rev => pyvcp.rev-led
```

Some of the lines might need some explanations. The fwd led line uses the signal created in the custom.hal file whereas the rev led needs to use the spindle-rev bit. You can’t link the spindle-fwd bit twice so you use the signal that it was linked to.

12.2.6. Rapid to Home Button

This example creates a button on the PyVCP side panel when pressed will send all the axis back to the home position. This example assumes you don’t have a PyVCP panel.



In your configuration directory create the XML file. In this example it's named *rth.xml*. In the *rth.xml* file add the following code to create the button.

```
<pyvcp>
<!-- rapid to home button example -->
<button>
<halpin>”rth-button”</halpin>
<text>”Rápido a Inicio”</text>
</button>
</pyvcp>
```

Open your INI file with a text editor and in the [DISPLAY] section add the following line. This is what loads the PyVCP panel.

```
PYVCP = rth.xml
```

If you don't have a [HALUI] section in the INI file create it and add the following MDI command.

```
MDI_COMMAND = G53 G0 X0 Y0 Z0
```

nota
Information about G53 and G0 G-codes.

In the [HAL] section if you don't have a post gui file add the following and create a file called *post-gui.hal*.

```
POSTGUI_HALFILE = postgui.hal
```

In the *postgui.hal* file add the following code to link the PyVCP button to the MDI command.

```
net rth halui.mdi-command-00 <= pyvcp.rth-button
```

nota
Information about the net command

12.3. GladeVCP: Glade Virtual Control Panel

12.3.1. ¿Qué es GladeVCP?

GladeVCP is a LinuxCNC component which adds the ability to add a new user interface panel to LinuxCNC user interfaces like:

- AXIS
- Touchy
- Gscreen
- GMOCCAPY

Unlike PyVCP, GladeVCP is not limited to displaying and setting HAL pins, as arbitrary actions can be executed in Python code - in fact, a complete LinuxCNC user interface could be built with GladeVCP and Python.

GladeVCP uses the Glade WYSIWYG user interface editor, which makes it easy to create visually pleasing panels. It relies on the PyGObject bindings to the rich GTK3 widget set, and in fact all of these widgets may be used in a GladeVCP application - not just the specialized widgets for interacting with HAL and LinuxCNC, which are documented here.

12.3.1.1. PyVCP versus GladeVCP at a glance

Both support the creation of panels with *HAL widgets* - user interface elements like LED's, buttons, sliders etc whose values are linked to a HAL pin, which in turn interfaces to the rest of LinuxCNC.

PyVCP:

- Widget set: uses TkInter widgets.
- User interface creation: "edit XML file / run result / evaluate looks" cycle.
- No support for embedding user-defined event handling.
- No LinuxCNC interaction beyond HAL pin I/O supported.

GladeVCP:

- Widget set: relies on the GTK3 widget set.

- User interface creation: uses the Glade WYSIWYG user interface editor.
- Any HAL pin change may be directed to call back into a user-defined Python event handler.
- Any GTK signal (key/button press, window, I/O, timer, network events) may be associated with user-defined handlers in Python.
- Direct LinuxCNC interaction: arbitrary command execution, like initiating MDI commands to call a G-code subroutine, plus support for status change operations through Action Widgets.
- Several independent GladeVCP panels may be run in different tabs.
- Separation of user interface appearance and functionality: change appearance without touching any code.

12.3.2. A Quick Tour with the Example Panel

GladeVCP panel windows may be run in three different setups:

- always visible integrated into AXIS at the right side, exactly like PyVCP panels,
- as a tab in AXIS,Touchy, Gscreen, or GMOCCAPY; in AXIS this would create a third tab besides the Preview and DRO tabs which must be raised explicitly,
- as a standalone toplevel window, which can be iconifyed/deiconified independent of the main window.

Installed LinuxCNC If you're using an installed version of LinuxCNC the examples shown below are in the configuration picker in the *Sample Configurations* > *apps* > *GladeVCP* branch.

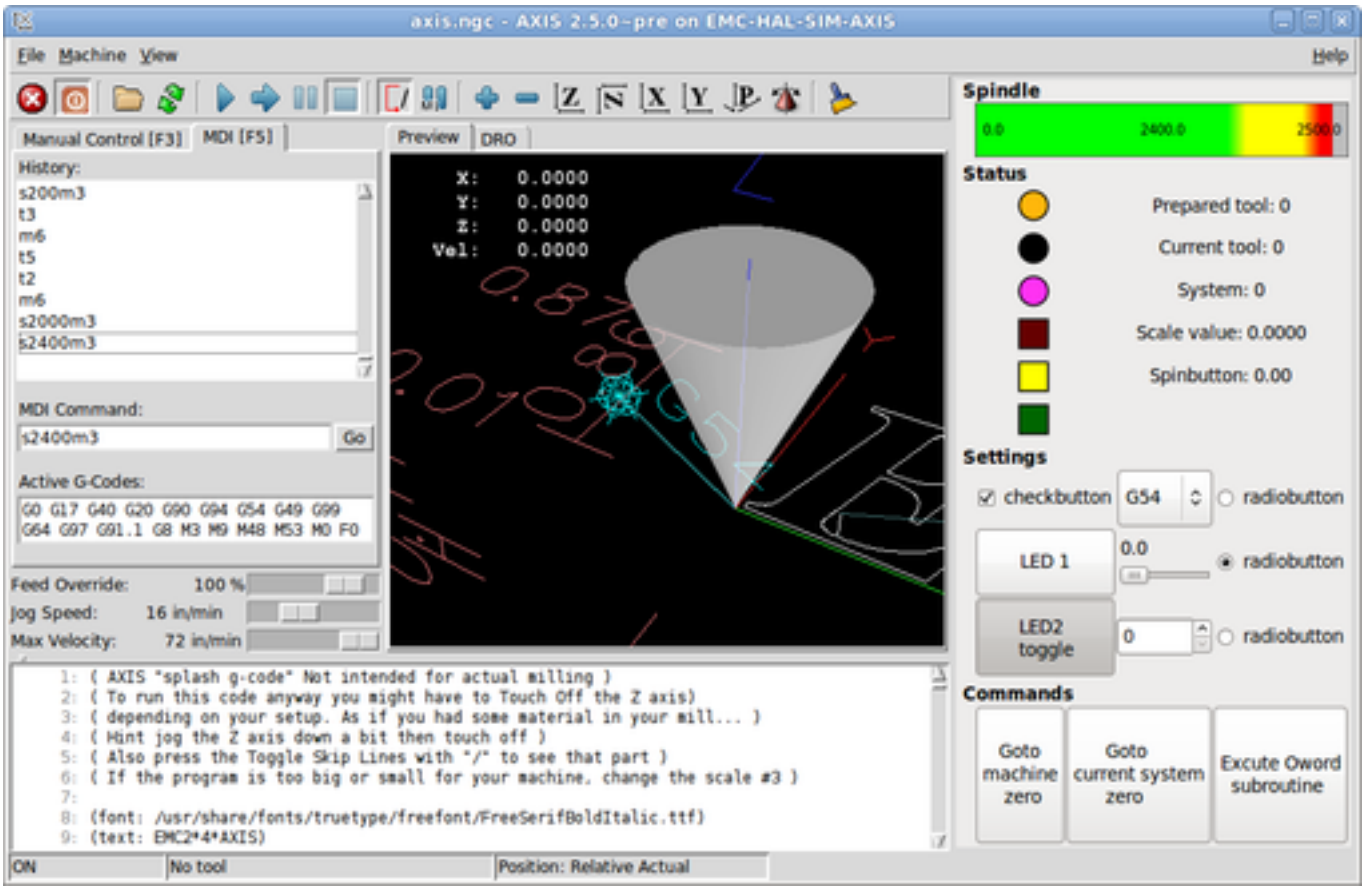
Git Checkout The following instructions only apply if you're using a git checkout. Open a terminal and change to the directory created by git then issue the commands as shown.

---

nota
For the following commands to work on your git checkout you must first run *make* then run *sudo make setuid* then run *. ./scripts/rip-environment*. More information about a git checkout is on the LinuxCNC wiki page.

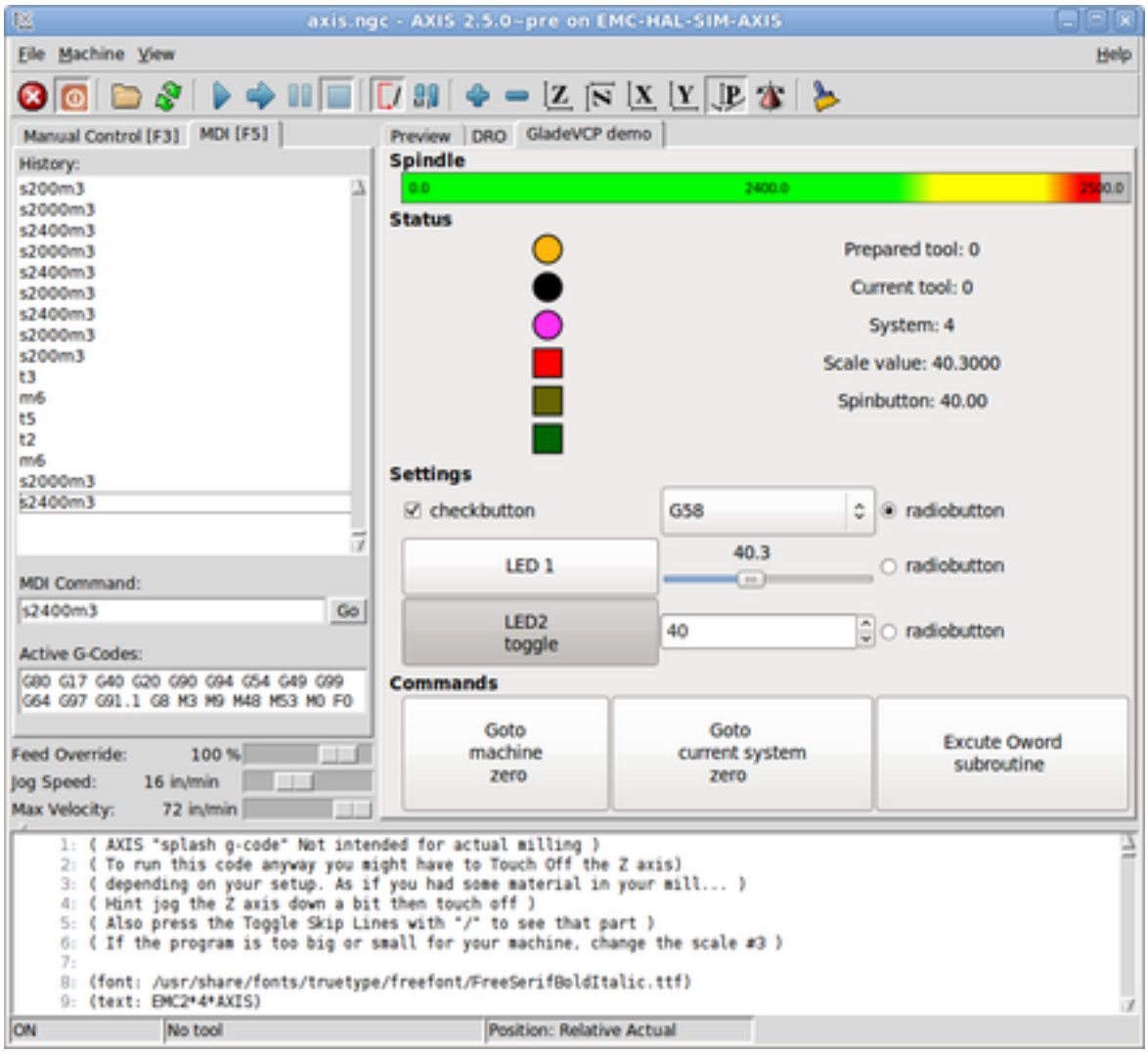
Run the sample GladeVCP panel integrated into AXIS like PyVCP as follows:

```
$ cd configs/sim/axis/gladevcp
$ linuxcnc gladevcp_panel.ini
```



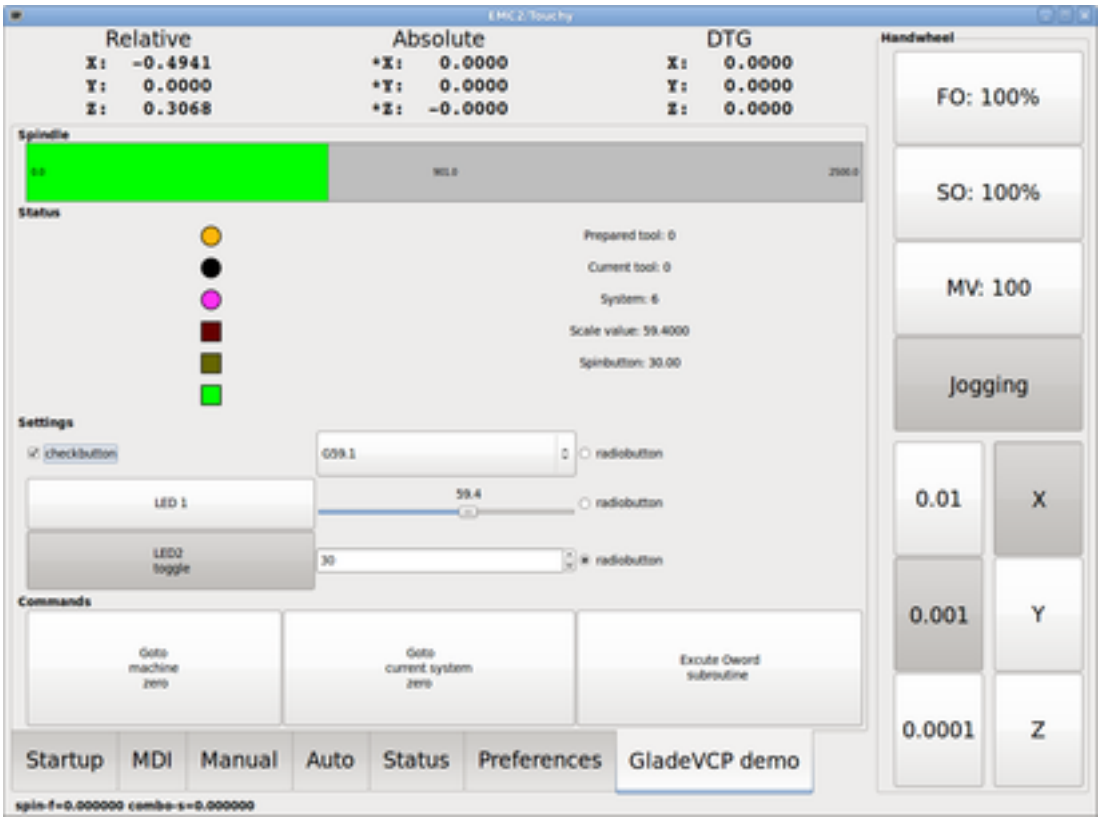
Run the same panel, but as a tab inside AXIS:

```
$ cd configs/sim/axis/gladevcp
$ linuxcnc gladevcp_tab.ini
```



To run this panel inside *Touchy*:

```
$ cd configs/sim/touchy/gladevcp
$ linuxcnc gladevcp_touchy.ini
```



Functionally these setups are identical - they only differ in screen real estate requirements and visibility. Since it is possible to run several GladeVCP components in parallel (with different HAL component names), mixed setups are possible as well - for instance a panel on the right hand side, and one or more tabs for less-frequently used parts of the interface.

12.3.2.1. Exploring the example panel

While running configs/sim/axis/gladevcp_panel.ini or configs/sim/axis/gladevcp_tab.ini, explore *Show HAL Configuration* - you will find the `gladevcp` HAL component and may observe their pin values while interacting with the widgets in the panel. The HAL setup can be found in *configs/axis/gladevcp/manual-example.hal*.

The example panel has two frames at the bottom. The panel is configured so that resetting ESTOP activates the Settings frame and turning the machine on enables the Commands frame at the bottom. The HAL widgets in the Settings frame are linked to LEDs and labels in the *Status* frame, and to the current and prepared tool number - play with them to see the effect. Executing the *T<toolnumber>* and *M6* commands in the MDI window will change the current and prepared tool number fields.

The buttons in the *Commands* frame are *MDI Action widgets* - pressing them will execute an MDI command in the interpreter. The third button *Execute Oword subroutine* is an advanced example - it takes several HAL pin values from the *Settings* frame, and passes them as parameters to the Oword subroutine. The actual parameters received by the routine are displayed by *(DEBUG,)* commands - see *../../nc_files/oword.ngc* for the subroutine body.

To see how the panel is integrated into AXIS, see the *[DISPLAY]GLADEVCP* statement in configs/sim/axis/gladevcp/gladevcp_panel.ini, the *[DISPLAY]EMBED** statement in configs/sim/axis/gladevcp/gladevcp_tab.ini and *[HAL]POSTGUI_HALFILE* statements in both configs/sim/axis/gladevcp/gladevcp_tab.ini and configs/sim/axis/gladevcp/gladevcp_panel.ini.

12.3.2.2. Exploring the User Interface description

The user interface is created with the Glade UI editor - to explore it, you need to have Glade installed. To edit the user interface, run the command

```
$ glade configs/axis/gladevcp/manual-example.ui
```

The required glade program may be named glade-gtk2 on more recent systems.

The center window shows the appearance of the UI. All user interface objects and support objects are found in the right top window, where you can select a specific widget (or by clicking on it in the center window). The properties of the selected widget are displayed, and can be changed, in the right bottom window.

To see how MDI commands are passed from the MDI Action widgets, explore the widgets listed under *Actions* in the top right window, and in the right bottom window, under the *General* tab, the *MDI command* property.

12.3.2.3. Exploring the Python callback

See how a Python callback is integrated into the example:

- In Glade, see the `hits` label widget (a plain GTK+ widget).
- In the `button1` widget, look at the *Signals* tab, and find the signal *pressed* associated with the handler *on_button_press*.
- In hitcounter.py, see the method *on_button_press* and see how it sets the label property in the *hits* object.

The is just touching upon the concept - the callback mechanism will be handled in more detail in the GladeVCP Programming section.

12.3.3. Creating and Integrating a Glade user interface

12.3.3.1. Prerequisite: Glade installation

To view or modify Glade UI files, you need Glade 3.38.2 or later installed - it is not needed just to run a GladeVCP panel. If the *glade* command is missing, install it with the command:

```
$ sudo apt install glade
```

Then verify installed version, which must be equal or superior to 3.6.7:

```
$ glade --version
```

Glade contains an internal Python interpreter, and only Python 3 is supported. This is true for Debian Bullseye, Ubuntu 21 and Mint 21 or later. Older versions will not work, you will get a Python error.

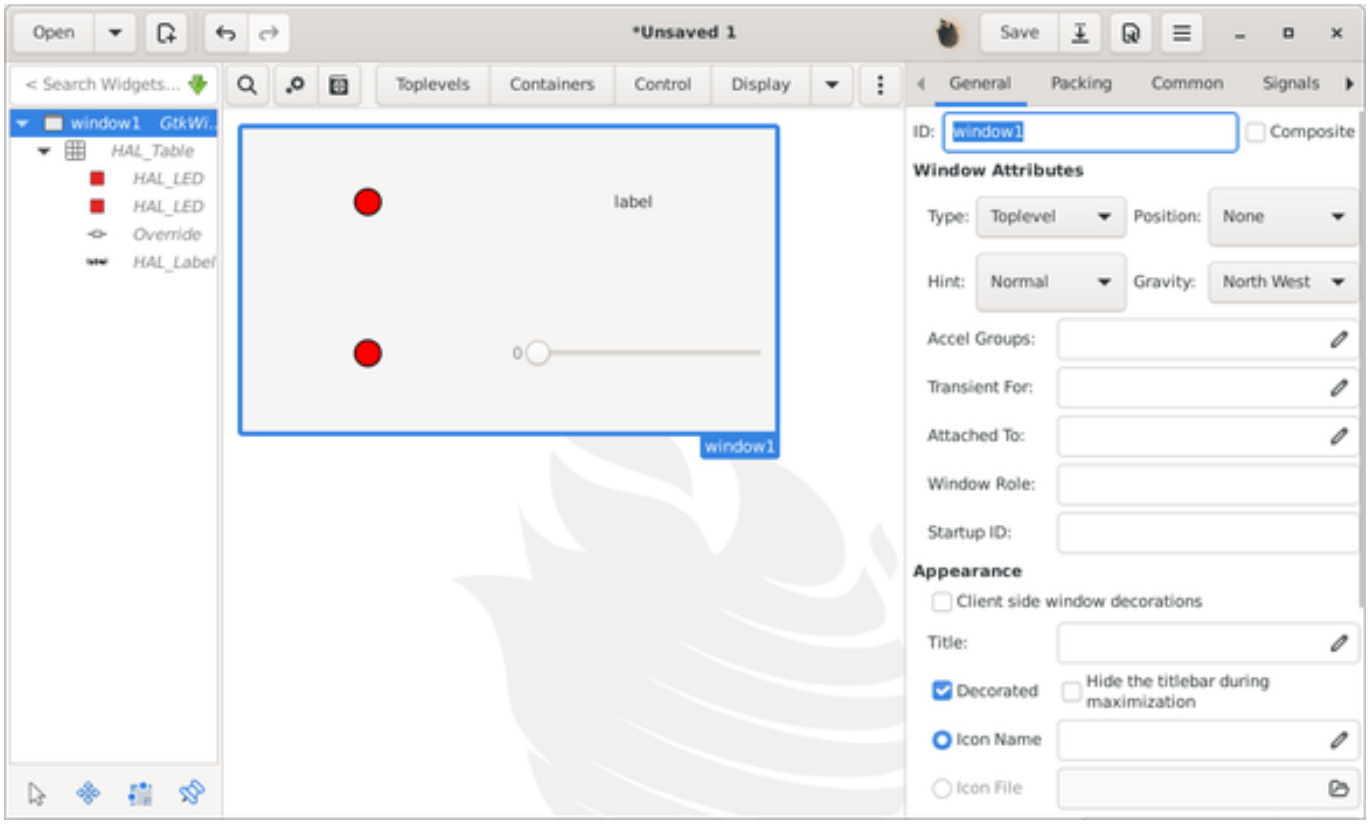
12.3.3.2. Running Glade to create a new user interface

This section just outlines the initial LinuxCNC-specific steps. For more information and a tutorial on Glade, see http://glade.gnome.org. Some Glade tips & tricks may also be found on YouTube.

Either modify an existing UI component by running `glade <file>.ui` or start a new one by just running the `glade` command from the shell.

- If LinuxCNC was not installed from a package, the LinuxCNC shell environment needs to be set up with `source <linuxcncdir>/scripts/rip-environment`, otherwise Glade won't find the LinuxCNC-specific widgets.
- When asked for unsaved preferences, just accept the defaults and hit *Close*.
- From *Toplevels* (toolbar), pick *GtkWindow* (first entry) as top level window. Set *window1* as ID in the right pane under the tab *General*. This naming is important because GladeVCP relies on it.
- From the button with the three dots you can find the LinuxCNC specific widgets.
- Add a container like a HAL_Box or a HAL_Table from *HAL Python* to the frame.
- Pick and place some elements like LED, button, etc. within a container.

This will look like this:



Glade tends to write a lot of messages to the shell window, which mostly can be ignored. Select *File→Save as*, give it a name like *myui.ui* and make sure it is saved as *GtkBuilder* file (radio button left bottom corner in Save dialog). GladeVCP will also process the older *libglade* format correctly but there is no point in using it. The convention for GtkBuilder file extension is *.ui*.

12.3.3.3. Testing a panel

You're now ready to give it a try (while LinuxCNC, e.g. AXIS is running) it with:

```
gladevcp myui.ui
```

GladeVCP creates a HAL component named like the basename of the UI file - *myui* in this case - unless overridden by the `-c <component name>` option. If running AXIS, just try *Show HAL configuration* and inspect its pins.

You might wonder why widgets contained a *HAL_Hbox* or *HAL_Table* appear greyed out (inactive). HAL containers have an associated HAL pin which is off by default, which causes all contained widgets to render inactive. A common use case would be to associate these container HAL pins with `halui.machine.is-on` or one of the `halui.mode` signals, to assure some widgets appear active only in a certain state.

To just activate a container, execute the HAL command `setp gladevcp.<container-name> 1`.

12.3.3.4. Preparing the HAL command file

The suggested way of linking HAL pins in a GladeVCP panel is to collect them in a separate file with extension *.hal*. This file is passed via the `POSTGUI_HALFILE=` option in the `HAL` section of your INI file.



atención
Do not add the GladeVCP HAL command file to the AXIS `[HAL]HALFILE=` ini section, this will not have the desired effect - see the following sections.

12.3.3.5. Integrating into AXIS, like PyVCP

Place the GladeVCP panel in the righthand side panel by specifying the following in the INI file:

```
[DISPLAY]
# add GladeVCP panel where PyVCP used to live:
GLADEVCP= -u ./hitcounter.py ./manual-example.ui

[HAL]
# HAL commands for GladeVCP components in a tab must be executed via POSTGUI_HALFILE
POSTGUI_HALFILE =  ./manual-example.hal

[RS274NGC]
# gladevcp Demo specific Oword subs live here
SUBROUTINE_PATH = ../../nc_files/gladevcp_lib
```

The default HAL component name of a GladeVCP application started with the GLADEVCP option is: `gladevcp`.

The command line actually run by AXIS in the above configuration is as follows:

```
halcmd loadusr -Wn gladevcp gladevcp -c gladevcp -x {XID} -u ./hitcounter.py ./manual- ↩
    example.ui
```

You may add arbitrary `gladevcp` options here, as long as they dont collide with the above command line options.

It is possible to create a custom HAL component name by adding the `-c` option:

```
[DISPLAY]
# add GladeVCP panel where PyVCP used to live:
GLADEVCP= -c example -u ./hitcounter.py ./manual-example.ui
```

The command line actually run by AXIS for the above is:

```
halcmd loadusr -Wn example gladevcp -c example -x {XID} -u ./hitcounter.py ./manual-example ↩
    .ui
```

nota
The file specifiers like ./hitcounter.py, ./manual-example.ui, etc. indicate that the files are located in the same directory as the INI file. You might have to copy them to you directory (alternatively, specify a correct absolute or relative path to the file(s)).

nota
The [RS274NGC]SUBROUTINE_PATH= option is only set so the example panel will find the Oword subroutine (oword.ngc) for the MDI Command widget. It might not be needed in your setup. The relative path specifier ../../nc_files/gladevcp_lib is constructed to work with directories copied by the configuration picker and when using a run-in-place setup.

12.3.3.6. Embedding as a Tab

To do so, edit your INI file and add to the DISPLAY and HAL sections of INI file as follows:

```
[DISPLAY]
# agrega panel GladeVCP como una pestaña después de Preview/DRO:
EMBED_TAB_NAME=GladeVCP demo
EMBED_TAB_COMMAND=halcmd loadusr -Wn gladevcp gladevcp -c gladevcp -x {XID} -u ./gladevcp/ ↩
    hitcounter.py ./gladevcp/manual-example.ui

[HAL]
# HAL commands for GladeVCP components in a tab must be executed via POSTGUI_HALFILE
POSTGUI_HALFILE =  ./gladevcp/manual-example.hal

[RS274NGC]
# gladevcp Demo specific Oword subs live here
SUBROUTINE_PATH = ../../nc_files/gladevcp_lib
```

Note the `halcmd loadusr` way of starting the tab command - this assures that *POSTGUI_HALFILE* will only be run after the HAL component is ready. In rare cases you might run a command here which uses a tab but does not have an associated HAL component. Such a command can be started without *halcmd loadusr*, and this signifies to AXIS that it does not have to wait for a HAL component since there is none.

When changing the component name in the above example, note that the names used in `-Wn <component>` and `-c <component>` must be identical.

Try it out by running AXIS - there should be a new tab called *GladeVCP demo* near the DRO tab. Select that tab, you should see the example panel nicely fit within AXIS.

nota
Make sure the UI file is the last option passed to GladeVCP in both the GLADEVCP= and EMBED_TAB_COMMAND= statements.

12.3.3.7. Integrating into Touchy

To do add a GladeVCP tab to *Touchy*, edit your INI file as follows:

```
[DISPLAY]
# Agrega el panel GladeVCP como pestaña
EMBED_TAB_NAME=GladeVCP demo
EMBED_TAB_COMMAND=gladevcp -c gladevcp -x {XID} -u ./hitcounter.py -H ./gladevcp-touchy.hal ↩
      ./manual-example.ui
```

```
[RS274NGC]
# gladevcp Demo specific Oword subs live here
SUBROUTINE_PATH = ../../nc_files/gladevcp_lib
```

nota
The file specifiers like ./hitcounter.py, ./manual-example.ui, etc. indicate that the files are located in the same directory as the INI file. You might have to copy them to you directory (alternatively, specify a correct absolute or relative path to the file(s)).

Note the following differences to the AXIS tab setup:

- The HAL command file is slightly modified since *Touchy* does not use the *halui* components so its signals are not available and some shortcuts have been taken.
- There is no *POSTGUI_HALFILE=* INI option, but passing the HAL command file on the *EMBED_TAB_COMM* line is ok.
- The *halcmd loaduser -Wn …* incantation is not needed.

12.3.3.8. Loading Builtin Panels

There are builtin panels available on the system, you load them in a slightly different way.
You do not add any filename extension to the panel name.
If the panel requires a user file (-u option) it will automatically be loaded.
Builtin panel names can be shown by running the gladevcp command alone in a terminal.

Loading the builtin verser probe panel:

```
gladevcp gtk_verser_probe
```

Embedding is the same, no filename extension, but other options are fine:

```
[DISPLAY]
# add GladeVCP panel as a tab next to Preview/DRO:
EMBED_TAB_NAME=GladeVCP demo
EMBED_TAB_COMMAND=halcmd loadusr -Wn gladevcp gladevcp -c gladevcp -x {XID}  ↩
    gtk_verser_probe
```

12.3.4. GladeVCP command line options

These are the GladeVCP command line options:
(See also `man gladevcp`.)
If you enter gladevcp in a terminal this is what you will see:

```
Usage: gladevcp [options] myfile.ui
usage: gladevcp [options] built_in_panel_name

Options:
  -h, --help      show this help message and exit
  -c NAME         Set component name to NAME. Default is basename of UI file
  -d              Enable debug output
  -g GEOMETRY     Set geometry WIDTHxHEIGHT+XOFFSET+YOFFSET. Values are in
                  pixel units, XOFFSET/YOFFSET is referenced from top left of
```

```
                  screen use -g WIDTHxHEIGHT for just setting size or -g
                  +XOFFSET+YOFFSET for just position
  -H FILE         execute hal statements from FILE with halcmd after the
                  component is set up and ready
  -i              Enable info output
  -m MAXIMUM      Force panel window to maxumize
  -q              Enable only error debug output
  -r GTK_RC       read custom GTK rc file to set widget style
  -t THEME        Set gtk theme. Default is system theme
  -x XID          Reparent gladevcp into an existing window XID instead of
                  creating a new top level window
  --xid           reparent window into a plug add push the plug xid number to
                  standardout
  -u FILE         Use FILEs as additional user defined modules with handlers
  -U USEROPT      pass USEROPTs to Python modules
  -v              Enable verbose debug output
  --always_above  Request the window To always be above other windows
  --ini=INI_PATH  ini path

[GladeVCP-][CRITICAL]  Available built-in VCP Panels: (gladevcp:205)
['gtk_verser_probe', 'gtk_little_probe']
```

12.3.5. Understanding the GladeVCP startup process

The integration steps outlined above look a bit tricky, and they are. It does therefore help to understand the startup process of LinuxCNC and how this relates to GladeVCP.

The normal LinuxCNC startup process does the following:

- The realtime environment is started.
- Todos los componentes HAL han sido cargados.
- The HAL components are linked together through the .hal cmd scripts.
- `task`, `iocontrol` and eventually the user interface is started.
- Pre-GladeVCP the assumption was: by the time the UI starts, all of HAL is loaded, plumbed and ready to go.

The introduction of GladeVCP brought the following issue:

- GladeVCP panels need to be embedded in a master GUI window setup.
- GladeVCP panels need to be embedded in a master GUI window setup, e.g., AXIS, or Touchy, Gscreen, or GMOCCAPY (embedded window or as an embedded tab).
- This requires the master GUI to run before the GladeVCP window can be hooked into the master GUI.
- However, GladeVCP is also a HAL component, and creates HAL pins of its own.
- As a consequence, all HAL plumbing involving GladeVCP HAL pins as source or destination must be run **after** the GUI has been set up.

This is the purpose of the `POSTGUI_HALFILE`. This INI option is inspected by the GUIs. If a GUI detects this option, it runs the corresponding HAL file after any embedded GladeVCP panel is set up. However, it does not check whether a GladeVCP panel is actually used, in which case the HAL cmd file is just run normally. So if you do NOT start GladeVCP through `GLADEVCP` or `EMBED_TAB` etc, but later in a separate

shell window or some other mechanism, a HAL command file in `POSTGUI_HALFILE` will be executed too early. Assuming GladeVCP pins are referenced herein, this will fail with an error message indicating that the GladeVCP HAL component is not available.

So, in case you run GladeVCP from a separate shell window (i.e., not started by the GUI in an embedded fashion):

- You cannot rely on the `POSTGUI_HALFILE` INI option causing the HAL commands being run *at the right point in time*, so comment that out in the INI file.
- Explicitly pass the HAL command file which refers to GladeVCP pins to GladeVCP with the `-H <halcmd file>` option (see previous section).

12.3.6. HAL Widget reference

GladeVCP includes a collection of Gtk widgets with attached HAL pins called HAL Widgets, intended to control, display or otherwise interact with the LinuxCNC HAL layer. They are intended to be used with the Glade user interface editor. With proper installation, the HAL Widgets should show up in Glade's *HAL Python* widget group. Many HAL specific fields in the Glade *General* section have an associated mouse-over tool tip.

HAL signals come in two variants, bits and numbers. Bits are off/on signals. Numbers can be "float", "s32" or "u32". For more information on HAL data types see the HAL manual. The GladeVCP widgets can either display the value of the signal with an indicator widget, or modify the signal value with a control widget. Thus there are four classes of GladeVCP widgets that you can connect to a HAL signal. Another class of helper widgets allow you to organize and label your panel.

- Widgets for indicating "bit" signals: HAL_LED
- Widgets for controlling "bit" signals: HAL_Button HAL_RadioButton HAL_CheckButton
- Widgets for indicating "number" signals: HAL_Label, HAL_ProgressBar, HAL_HBar and HAL_VBar, HAL_Meter
- Widgets for controlling "number" signals: HAL_SpinButton, HAL_HScale and HAL_VScale, Jog Wheel, Speed Control
- Sensitive control widgets: State_Sensitive_Table HAL_Table and HAL_HBox
- Tool Path preview: HAL_Gremlin
- Widgets to show axis positions: DRO Widget, Combi DRO Widget
- Widgets for file handling: IconView File Selection
- Widgets for display/edit of all axes offsets: OffsetPage
- Widgets for display/edit of all tool offsets: Tooloffset editor
- Widget for G-code display and edit: HAL_Sourceview
- Widget for MDI input and history display: MDI History

12.3.6.1. Widget and HAL pin naming

Most HAL widgets have a single associated HAL pin with the same HAL name as the widget (glade: General→Name).

Exceptions to this rule currently are:

- *HAL_Spinbutton* and *HAL_ComboBox*, which have two pins: a `<widgetname>-f` (float) and a `<widgetname>` (s32) pin
- *HAL_ProgressBar*, which has a `<widgetname>-value` input pin, and a `<widgetname>-scale` input pin.

12.3.6.2. Python attributes and methods of HAL Widgets

HAL widgets are instances of GtKWidgets and hence inherit the methods, properties and signals of the applicable GtkWidget class. For instance, to figure out which GtkWidget-related methods, properties and signals a *HAL_Button* has, lookup the description of GtkButton in the PyGObject API Reference.

An easy way to find out the inheritance relationship of a given HAL widget is as follows: Run glade, place the widget in a window, and select it; then choose the *Signals* tab in the *Properties* window. For example, selecting a *HAL_LED* widget, this will show that a *HAL_LED* is derived from a *GtkWidget*, which in turn is derived from a *GtkObject*, and eventually a *GObject*.

Full class hierarchy can be seen by invoking the GtkInspector while in the Glade GUI by selecting a widget then pressing Control-Shift-I. If the Inspector doesn't open then it can be enabled from a terminal by entering:

```
gsettings set org.gtk.Settings.Debug enable-inspector-keybinding true
```

The Inspector is also handy for testing css style changes "on the fly" as well as determining all the properties and signals available for a widget.

HAL Widgets also have a few HAL-specific Python attributes:

hal_pin
: The underlying HAL pin Python object in case the widget has a single pin type

hal_pin_s, hal_pin_f
: The s32 and float pins of the *HAL_Spinbutton* and *HAL_ComboBox* widgets - note these widgets do not have a *hal_pin* attribute!

hal_pin_scale
: The float input pin of *HAL_ProgressBar* widget representing the maximum absolute value of input.

The are several HAL-specific methods of HAL Widgets, but the only relevant method is:

<pin-hal>.get()
: Retrieve the value of the current HAL pin, where <*halpin*> is the applicable HAL pin name listed above.

12.3.6.3. Setting pin and widget values

As a general rule, if you need to set a HAL output widget's value from Python code, do so by calling the underlying Gtk *setter* (e.g., `set_active()`, `set_value()`). Do not try to set the associated pin's value by `halcomp[pinname] = value` directly because the widget will not take notice of the change!

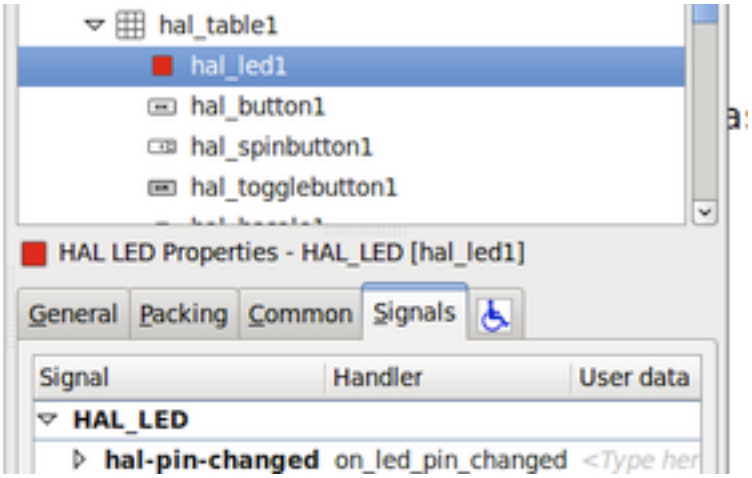
It might be tempting to *set HAL widget input pins* programmatically. Note this defeats the purpose of an input pin in the first place - it should be linked to, and react to signals generated by other HAL components. While there is currently no write protection on writing to input pins in HAL Python, this doesn't make sense. You might use `setp _pinname_ _value_` in the associated HAL file for testing though.

It is perfectly OK to set an output HAL pin's value with `halcomp[pinname] = value` provided this HAL pin is not associated with a widget, that is, has been created by the `hal_glib.GPin(halcomp.newpin(<name>` method (see GladeVCP Programming for an example).

12.3.6.4. The hal-pin-changed signal

Event-driven programming means that the UI tells your code when ”something happens” - through a callback, like when a button was pressed. The output HAL widgets (those which display a HAL pin’s value) like LED, Bar, VBar, Meter, etc., support the `hal-pin-changed` signal, which may cause a callback into your Python code when - well, a HAL pin changes its value. This means there’s no more need for permanent polling of HAL pin changes in your code, the widgets do that in the background and let you know.

Here is an example how to set a `hal-pin-changed` signal for a HAL_LED in the Glade UI editor:



The example in `configs/apps/gladevcp/complex` shows how this is handled in Python.

12.3.6.5. Buttons

This group of widgets are derived from various Gtk buttons and consists of HAL_Button, HAL_ToggleButton, HAL_RadioButton and CheckButton widgets. All of them have a single output BIT pin named identical to the widget. Buttons have no additional properties compared to their base Gtk classes.

- HAL_Button: instantaneous action, does not retain state. Important signal: `pressed`
- HAL_ToggleButton, HAL_CheckButton: retains on/off state. Important signal: `toggled`
- HAL_RadioButton: a one-of-many group. Important signal: `toggled` (per button).
- Important common methods: `set_active()`, `get_active()`
- Important properties: `label`, `image`

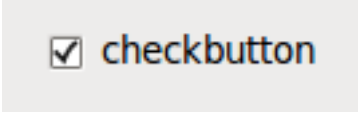


Figura 12.31: Check button

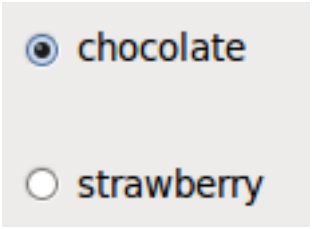


Figura 12.32: Radio buttons

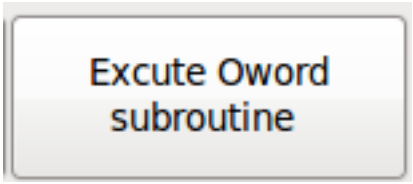


Figura 12.33: Toggle button

---

sugerencia
Defining radio button groups in Glade:

- Decide on default active button.
- In the other button's *General→Group* select the default active button's name in the *Choose a Radio Button in this project* dialog.

See `configs/apps/gladevcp/by-widget/` for a GladeVCP applications and UI file for working with radio buttons.

12.3.6.6. Scales

HAL_HScale and HAL_VScale are derived from the GtkHScale and GtkVScale respectively.

<widgetname>
out FLOAT pin

<widgetname>-s
out s32 pin

To make a scale useful in Glade, add an *Adjustment* (General → Adjustment → New or existing adjustment) and edit the adjustment object. It defines the default/min/max/increment values. Also, set adjustment *Page size* and *Page increment* to zero to avoid warnings.

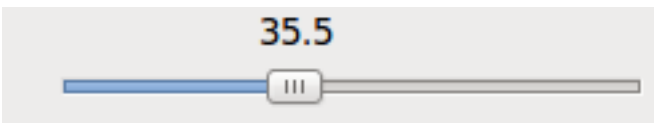


Figura 12.34: Ejemplo de HAL_HScale

12.3.6.7. SpinButton

HAL SpinButton is derived from GtkSpinButton and holds two pins:

<widgetname>-f
out FLOAT pin

<widgetname>-s
out s32 pin

To be useful, Spinbuttons need an adjustment value like scales, see above.

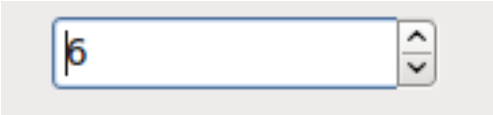


Figura 12.35: Example SpinButton

12.3.6.8. Hal_Dial

The hal_dial widget simulates a jogwheel or adjustment dial.
It can be operated with the mouse. You can just use the mouse wheel, while the mouse cursor is over the Hal_Dial widget, or you hold the left mouse button and move the cursor in circular direction to increase or degrease the counts.
By double clicking the left or right button the scale factor can be increased or decreased.

- Counterclockwise = reduce counts
- Clockwise = increase counts
- Wheel up = increase counts
- Wheel down = reduce counts
- left Double Click = x10 scale
- Right Double Click = /10 scale

`hal_dial` exports its count value as HAL pins:

<widgetname>
out s32 pin

<widgetname>-scaled
out FLOAT pin

<widgetname>-delta-scaled
out FLOAT pin

`hal_dial` has the following properties:

cpr
Sets the Counts per Revolution, allowed values are in the range from 25 to 360
default = 100

show_counts
Set this to False, if you want to hide the counts display in the middle of the widget.
default = True

label
Set the content of the label which may be shown over the counts value.
If the label given is longer than 15 Characters, it will be cut to 15 Characters.
default = blank

center_color
This allows one to change the color of the wheel. It uses a GDK color string.
default = #bdefbdefbdef (gray)

count_type_shown
There are three counts available 0) Raw CPR counts 1) Scaled counts 2) Delta scaled counts.
default = 1

- count is based on the CPR selected - it will count positive and negative. It is available as a s32 pin.
- Scaled-count is CPR count times the scale - it can be positive and negative.
  If you change the scale the output will immediately reflect the change. It is available as a FLOAT pin.
- Delta-scaled-count is cpr count CHANGE, times scale.
  If you change the scale, only the counts after that change will be scaled and then added to the current value.
  It is available as a FLOAT pin.

scale_adjustable
Set this to False if you want to disallow scale changes by double clicking the widget.
If this is false the scale factor will not show on the widget.
default = True

scale
Set this to scale the counts.
default = 1.0

There are ways to directly control the widget using Python.

Using goobject to set the above listed properties:

```
[widget name].set_property(”cpr”,int(value))
[widget name].set_property(”show_counts, True)
[widget name].set_property(”center_color”,gtk.gdk.Color(’#bdefbdefbdef’))
[widget name].set_property(’label’, ’Test Dial 12345’)
[widget name].set_property(’scale_adjustable’, True)
[widget name].set_property(’scale’, 10.5)
[widget name].set_property(’count_type_shown’, 0)
```

There are Python methods:

- `[widget name].get_value()`
  Will return the counts value as a s32 integer
- `[widget name].get_scaled_value()`
  Will return the counts value as a float
- `[widget name].get_delta_scaled_value()`
  Will return the counts value as a float
- `[widget name].set_label(”string”)`
  Sets the label content with ”string”

There are two GObject signals emitted:

- `count_changed`
  Emitted when the widget's count changes eg. from being wheel scrolled.
- `scale_changed`
  Emitted when the widget's scale changes eg. from double clicking.

Connect to these like so:

```
[widget name].connect('count_changed', [count function name])
[widget name].connect('scale_changed', [scale function name])
```

The callback functions would use this pattern:

```
def [count function name](widget, count,scale,delta_scale):
```

This will return: the widget, the current count, scale and delta scale of that widget.

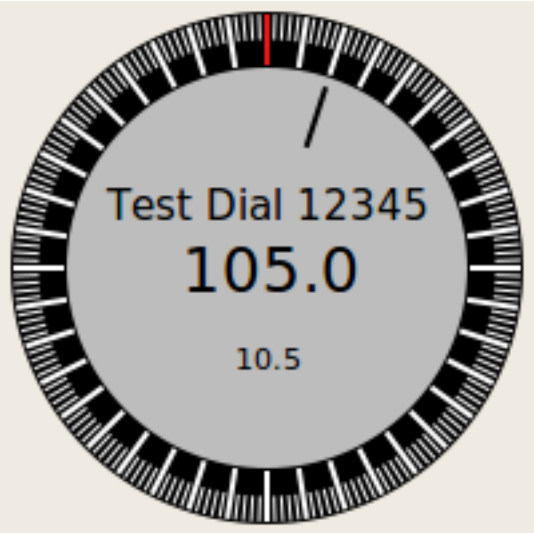


Figura 12.36: Ejemplo de Hal_Dial

12.3.6.9. Jog Wheel

The `jogwheel` widget simulates a real jogwheel. It can be operated with the mouse. You can just use the mouse wheel, while the mouse cursor is over the JogWheel widget, or you push the left mouse button and move the cursor in circular direction to increase or degrease the counts.

- Counterclockwise = reduce counts
- Clockwise = increase counts
- Wheel up = increase counts
- Wheel down = reduce counts

As moving the mouse the drag and drop way may be faster than the widget can update itself, you may loose counts turning to fast. It is recommended to use the mouse wheel, and only for very rough movements the drag and drop way.

`jogwheel` exports its count value as HAL pin:

<widgetname>-s
out s32 pin

`jogwheel` has the following properties:

size
Sets the size in pixel of the widget, allowed values are in the range of 100 to 500 default = 200

cpr
Sets the Counts per Revolution, allowed values are in the range from 25 to 100 default = 40

show_counts
Set this to False, if you want to hide the counts display in the middle of the widget.

label
Set the content of the label which may be shown over the counts value. The purpose is to give the user an idea about the usage of that jogwheel. If the label given is longer than 12 Characters, it will be cut to 12 Characters.

There a couple ways to directly control the widget using Python.

Using GObject to set the above listed properties:

```
[widget name].set_property(”size”,int(value))
[widget name].set_property(”cpr”,int(value))
[widget name].set_property(”show_counts, True)
```

There are two Python methods:

- `[widget name].get_value()`
  Will return the counts value as integer
- `[widget name].set_label(”string”)`
  Sets the label content with ”string”

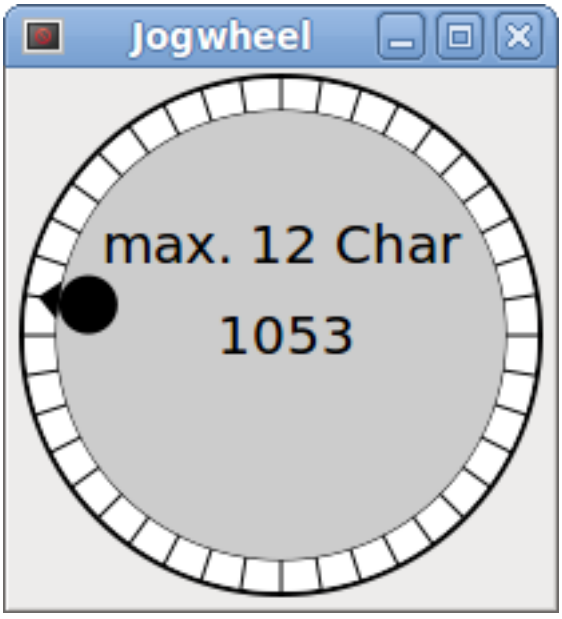


Figura 12.37: Example JogWheel

12.3.6.10. Speed Control

`speedcontrol` is a widget specially made to control an adjustment with a touch screen. It is a replacement to the normal scale widget which is difficult to slide on a touch screen.

The value is controlled with two button to increase or decrease the value. The increment will change as long a button is pressed. The value of each increment as well as the time between two changes can be set using the widget properties.

`speedcontrol` offers some HAL pin:

<widgetname>-value
: out float pin
The shown value of the widget.

<widgetname>-scaled-value
: out float pin
The shown value divided by the scale value, this is very useful, if the velocity is shown in units / min, but LinuxCNC expects it to be in units / second.

<widgetname>-scale
: in float pin
The scale to apply.
Default is 60.

<widgetname>-increase
: in bit pin
As long as the pin is true, the value will increase.
Very handy with connected momentary switch.

<widgetname>-decrease
: in bit pin
As long as the pin is true, the value will decrease.
Very handy with connected momentary switch.

`speedcontrol` has the following properties:

height
: Integer
The height of the widget in pixel.
Allowed values are 24 to 96.
Default is 36.

valor
: Float
The start value to set.
Allowed values are in the range from 0.001 to 99999.0.
Default is 10.0.

min
: Float
The min allowed value.
Allowed values are 0.0 to 99999.0.
Default is 0.0.
If you change this value, the increment will be reset to default, so it might be necessary to set afterwards a new increment.

max
Float
The max allowed value.
Allowed values are 0.001 to 99999.0.
Default is 100.0.
If you change this value, the increment will be reset to default, so it might be necessary to set afterwards a new increment.

increment
Float
Sets the applied increment per mouse click.
Allowed values are 0.001 to 99999.0 and -1.
Default is -1, resulting in 100 increments from min to max.

inc_speed
Integer
Sets the timer delay for the increment speed holding pressed the buttons.
Allowed values are 20 to 300.
Default is 100.

unit
String
Sets the unit to be shown in the bar after the value.
Any string is allowed.
Default is ”“.

color
Color
Sets the color of the bar.
Any hex color is allowed.
Default is ”#FF8116”.

template
String
Text template to display the value. Python formatting is used.
Any allowed format.
Default is ” %.1f”.

do_hide_button
Boolean
Whether to show or hide the increment an decrement button.
True or False.
Default = False.

There a couple ways to directly control the widget using Python.

Using GObject to set the above listed properties:

```
[widget name].set_property(”do_hide_button”,bool(value))
[widget name].set_property(”color”,”#FF00FF”)
[widget name].set_property(”unit”, ”mm/min”)
etc.
```

There are also Python methods to modify the widget:

```
[widget name].set_adjustment(gtk-adjustment)
```

You can assign a existing adjustment to the control, that way it is easy to replace existing sliders without many code changes. Be aware, that after changing the adjustment you may need to set a new increment, as it will be reset to its default (100 steps from MIN to MAX):

- `[widget name].get_value()`
  Will return the counts value as float
- `[widget name].set_value(float(value))`
  Sets the widget to the commanded value
- `[widget name].set_digits(int(value))`
  Sets the digits of the value to be used
- `[widget name].hide_button(bool(value))`
  Hide or show the button

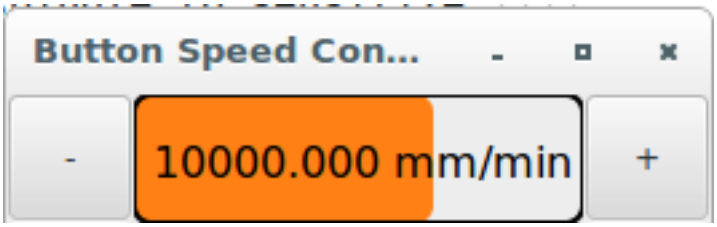


Figura 12.38: Ejemplo de Speedcontrol

12.3.6.11. Label

`hal_label` is a simple widget based on GtkLabel which represents a HAL pin value in a user-defined format.

label_pin_type
The pin's HAL type (0:s32, 1:float, 2:u32), see also the tooltip on *General→HAL pin type* (note this is different from PyVCP which has three label widgets, one for each type).

text_template
Determines the text displayed - a Python format string to convert the pin value to text. Defaults to `%s` (values are converted by the str() function) but may contain any legit as an argument to Pythons format() method.
Example: `Distance: %.03f` will display the text and the pin value with 3 fractional digits padded with zeros for a FLOAT pin.

12.3.6.12. Containers

- HAL_HideTable
- HAL_Table
- State_Sensitive_Table
- HAL_HBox (deprecated)

These containers are meant to be used to insensitize (grey out) or hide their children.
Insensitized children will not respond to input.

HAL_HideTable
Has one HAL BIT input pin which controls if its child widgets are hidden or not.

Pin: , <Panel_basename>.<widgetname>
in bit pin
If the pin is low then child widgets are visible which is the default state.

HAL_Table and HAL_Hbox
Have one HAL BIT input pin which controls if their child widgets are sensitive or not.

Pin: , <Panel_basename>.<widgetname>
in bit pin
If the pin is low then child widgets are inactive which is the default state.

State_Sensitive_Table
Responds to the state to LinuxCNC's interpreter.
Optionally selectable to respond to *must-be-all-homed*, *must-be-on* and *must-be-idle*.
You can combine them. It will always be insensitive at Estop.
(Has no pin).



aviso
HAL_Hbox is deprecated - use HAL_Table.
If current panels use it, it won't fail. You just won't find it in the GLADE editor anymore.
Future versions of GladeVCP may remove this widget completely and then you will need to update the panel.

sugerencia
If you find some part of your GladeVCP application is *grayed out* (insensitive), see whether a HAL_Table pin is unset or unconnected.

12.3.6.13. LED

The `hal_led` simulates a real indicator LED.
It has a single input BIT pin which controls its state: ON or OFF.

LEDs have several properties which control their look and feel:

on_color
String defining ON color of LED.
May be any valid gdk.Color name.
Not working on Ubuntu 8.04.

off_color
String defining OFF color of LED.
May be any valid gdk.Color name or special value `dark`. `dark` means that OFF color will be set to 0.4 value of ON color.
Not working on Ubuntu 8.04.

pick_color_on, pick_color_off
Colors for ON and OFF states.
These may be represented as `#RRRRGGGGBBBB` strings and are optional properties which have precedence over `on_color` and `off_color`.

led_size
LED radius (for square - half of LED's side)

led_shape
LED Shape.
Valid values are 0 for round, 1 for oval and 2 for square shapes.

led_blink_rate
If set and LED is ON then it is blinking.
Blink period is equal to ”led_blink_rate” specified in milliseconds.

create_hal_pin
Select/deselect creation of a HAL pin to control the LED.
With no HAL pin created LED can be controlled with a Python function.

As an input widget, LED also supports the `hal-pin-changed` signal. If you want to get a notification in your code when the LED’s HAL pin was changed, then connect this signal to a handler, for example `on_led_pin_changed` and provide the handler as follows:

```
def on_led_pin_changed(self,hal_led,data=None):
    print(”on_led_pin_changed() - HAL pin value:”,hal_led.hal_pin.get())
```

This will be called at any edge of the signal and also during program start up to report the current value.

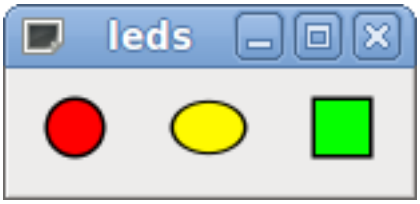


Figura 12.39: Example LEDs

12.3.6.14. ProgressBar

nota
This widget might go away.
Use the HAL_HBar and HAL_VBar widgets instead.

The `HAL_ProgressBar` is derived from gtk.ProgressBar and has two float HAL input pins:

<widgetname>
the current value to be displayed

<widgetname>-scale
the maximum absolute value of input

`HAL_ProgressBar` has the following properties:

scale
Value scale.
Sets the maximum absolute value of input. Same as setting the <widgetname>.scale pin.
A float, range from -2^{24} to $+2^{24}$.

green_limit
Green zone lower limit

yellow_limit
Yellow zone lower limit

red_limit
Red zone lower limit

text_template
Text template to display the current value of the `<widgetname>` pin.
Python formatting may be used for dict `{"value":value}`.



Figura 12.40: Example HAL_ProgressBar

12.3.6.15. ComboBox

`HAL_ComboBox` is derived from gtk.ComboBox. It enables choice of a value from a dropdown list.

`HAL_ComboBox` exports two HAL pins:

<widgetname>-f
Current value, type FLOAT

<widgetname>-s
Current value, type s32

`HAL_ComboBox` has the following property which can be set in Glade:

column
The column index.
Type s32.
Valid range from -1..100.
Defaults value -1.

In default mode this widgets sets the pins to the index of the chosen list entry. So if your widget has three labels, it may only assume values 0,1 and 2.

In column mode (column > -1), the value reported is chosen from the ListStore array as defined in Glade. So typically your widget definition would have two columns in the ListStore, one with text displayed in the dropdown, and an int or float value to use for that choice.

There's an example in `configs/apps/by-widget/combobox.{py,ui}` which uses column mode to pick a float value from the ListStore.

If you're confused like me about how to edit ComboBox ListStores and CellRenderer, see https://youtu.be/-Z5_F-rW2cL8.

12.3.6.16. Bars

`HAL_Bar` and `HAL_VBar` widgets for horizontal and vertical bars representing float values.

`HAL_Bar` and `HAL_VBar` each have one input FLOAT HAL pin.

`HAL_Bar` and `HAL_VBar` both bars have the following properties:

invert
Swap min and max direction.
An inverted HBar grows from right to left, an inverted VBar from top to bottom.

min, max
Minimum and maximum value of desired range. It is not an error condition if the current value is outside this range.

show limits
Used to select/deselect the limits text on bar.

zero
Zero point of range.
If it is inside of min/max range then the bar will grow from that value and not from the left (or right) side of the widget.
Useful to represent values that may be both positive or negative.

force_width, force_height
Forced width or height of widget.
If not set then size will be deduced from packing or from fixed widget size and bar will fill whole area.

text_template
Like in Label, sets text format for min/max/current values.
Can be used to turn off value display.

valor
Sets the bar display to the value entered.
Used only for testing in GLADE editor.
The value will be set from a HAL pin.

target value
Sets the target line to the value entered.
Used only for testing in GLADE editor.
The value will can be set in a Python function.

target_width
Width of the line that marks the target value.

bg_color
Background (inactive) color of bar.

target_color
Color of the the target line.

z0_color, z1_color, z2_color
Colors of different value zones.
Defaults are `green`, `yellow` and `red`.
For description of zones see `z*_border` properties.

z0_border, z1_border
Define up bounds of color zones.
By default only one zone is enabled. If you want more then one zone set `z0_border` and `z1_border` to desired values so zone 0 will fill from 0 to first border, zone 1 will fill from first to second border and zone 2 from last border to 1.
Borders are set as fractions.
Valid values range from 0 to 1.

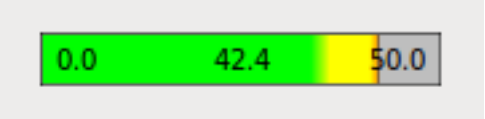


Figura 12.41: Horizontal bar

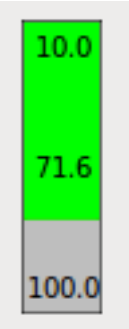


Figura 12.42: Vertical bar

12.3.6.17. Meter

`HAL_Meter` is a widget similar to PyVCP meter - it represents a float value.

`HAL_Meter` has one input FLOAT HAL pin.

HAL Meter has the following properties:

min, max
Minimum and maximum value of desired range.
It is not an error condition if the current value is outside this range.

force_size
Forced diameter of widget.
If not set then size will be deduced from packing or from fixed widget size, and meter will fill all available space with respect to aspect ratio.

text_template
Like in Label, sets text format for current value.
Can be used to turn off value display.

label
Large label above center of meter.

sublabel
Small label below center of meter.

bg_color
Background color of meter.

z0_color, z1_color, z2_color
Colors of different value zones.
Defaults are `green`, `yellow` and `red`.
For description of zones see `z*_border` properties.

z0_border, z1_border
Define up bounds of color zones.
By default only one zone is enabled. If you want more then one zone set `z0_border` and `z1_border` to desired values so zone 0 will fill from min to first border, zone 1 will fill from first to second border and zone 2 from last border to max.
Borders are set as values in range min-max.

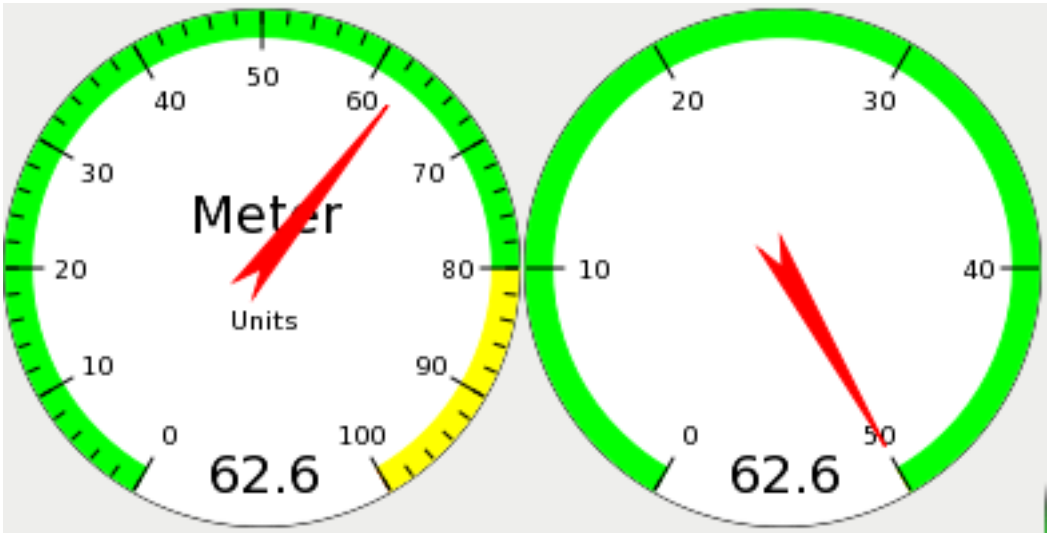


Figura 12.43: Example HAL Meters

12.3.6.18. HAL_Graph

This widget is for plotting values over time.

12.3.6.19. Gremlin tool path preview for NGC files

Gremlin is a plot preview widget similar to the AXIS preview window. It assumes a running LinuxCNC environment like AXIS or Touchy. To connect to it, inspects the INI_FILE_NAME environment variable. Gremlin displays the current NGC file - it does monitor for changes and reloads the ngc file if the file name in AXIS/Touchy changes. If you run it in a GladeVCP application when LinuxCNC is not running, you might get a traceback because the Gremlin widget can't find LinuxCNC status, like the current file name.

Gremlin does not export any HAL pins.

Gremlin has the following properties:

enable_dro
This displays the dro on the graphics.
Default = true.

show_velocity
This displays the tool speed.
Default = true.

use_commanded
This selects the DRO to use: commanded or actual values.
Default = true.

metric_units
This selects the DRO to use: metric or imperial units.
Default = true.

show_rapids
This tells the plotter to show the rapid moves.
Default = true.

show_dtg_
This selects the DRO to display the distance-to-go value.
Default = true.

use_relative
This selects the DRO to show values relative to user system or machine coordinates.
Default = true.

show_live_plot
This tells the plotter to draw or not.
Default = true.

show_limits
This tells the plotter to show the machine's limits.
Default = true.

show_lathe_radius
This selects the DRO to display the X axis in radius or diameter, if in lathe mode (selectable in the INI file with LATHE = 1).
Default = true.

show_extents_option
This tells the plotter to show the machine's extents.
Default = true.

show_tool
This tells the plotter to draw the tool.
Default = true.

show_program
Shows the G-code program.
Default = True

use_joints_mode
Used in non trivialkins machines (e.g., robots).
Default = false.

grid_size
Sets the size of the grid (only visible in the X, Y and Z views).
Defaults to 0

use_default_controls
This disables the default mouse controls.
This is most useful when using a touchscreen as the default controls do not work well. You can programmatically add controls using Python and the handler file technique.
Default = true.

view
May be any of `x`, `y`, `y2` , `z`, `z2` , `p` (perspective).
Defaults to `z` view.

enable_dro
Type = boolean.
Whether to draw a DRO on the plot or not.
Default = true.

mouse_btn_mode
Type = integer.
Mouse button handling: leads to different functions of the button:

- 0 = default: left rotate, middle move, right zoom
- 1 = left zoom, middle move, right rotate
- 2 = left move, middle rotate, right zoom
- 3 = left zoom, middle rotate, right move
- 4 = left move, middle zoom, right rotate
- 5 = left rotate, middle zoom, right move
- 6 = left move, middle zoom, right zoom

Mode 6 is recommended for plasmas and lathes, as rotation is not needed for such machines.

There a couple ways to directly control the widget using Python.

Using GObject to set the above listed properties:

```
[widget name].set_property('view','P')
[widget name].set_property('metric_units',False)
[widget name].set_property('use_default_controls',False)
[widget name].set_property('enable_dro' False)
[widget name].set_property('show_program', False)
[widget name].set_property('show_limits', False)
[widget name].set_property('show_extents_option', False)
[widget name].set_property('show_live_plot', False)
[widget name].set_property('show_tool', False)
[widget name].set_property('show_lathe_radius',True)
[widget name].set_property('show_dtg',True)
[widget name].set_property('show_velocity',False)
[widget name].set_property('mouse_btn_mode', 4)
```

There are Python methods:

```
[widget name].show_offsets = True
[widget name].grid_size = .75
[widget name].select_fire(event.x,event.y)
[widget name].select_prime(event.x,event.y)
[widget name].start_continuous_zoom(event.y)
[widget name].set_mouse_start(0,0)
[widget name].gremlin.zoom_in()
[widget name].gremlin.zoom_out()
[widget name].get_zoom_distance()
[widget name].set_zoom_distance(dist)
[widget name].clear_live_plotter()
[widget name].rotate_view(x,y)
[widget name].pan(x,y)
```

Hints

- If you set all the plotting options false but show_offsets true you get an offsets page instead of a graphics plot.
- If you get the zoom distance before changing the view then reset the zoom distance, it is much more user friendly.
- Si selecciona un elemento en la vista previa, el elemento seleccionado será tomado como punto central de rotación

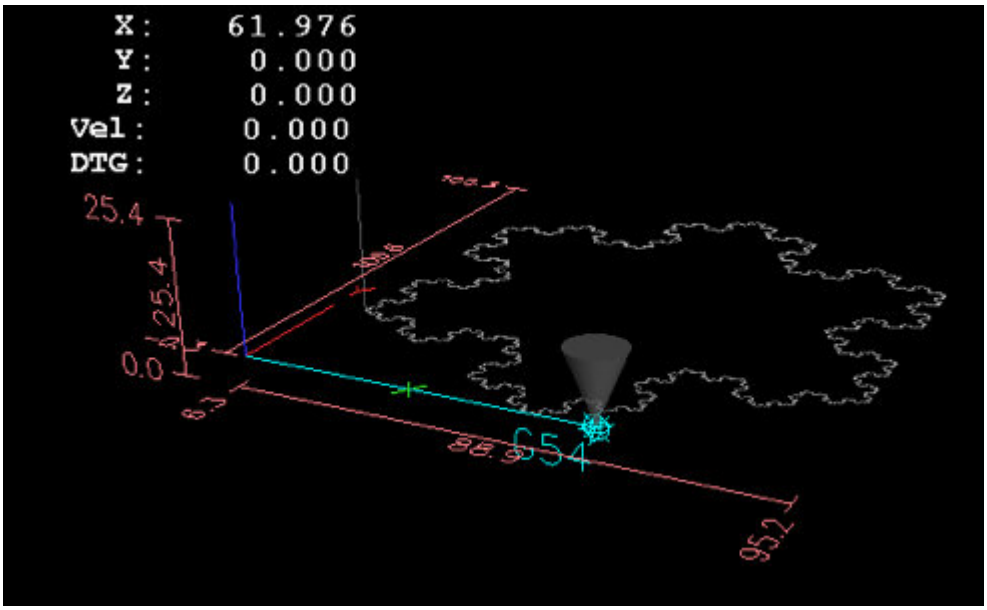


Figura 12.44: Gremlin Example

12.3.6.20. HAL_Offset

The `HAL_Offset` widget is used to display the offset of a single axis.

`HAL_Offset` has the following properties:

display_units_mm
Mostrar en unidades métricas.

joint_number
Used to select which axis (technically which joint) is displayed.
On a trivialkins machine (mill, lathe, router) axis vs. joint number are:

```
0:X 1:Y 2:Z 3:A 4:B 5:C 6:U 7:V 8:W
```

mm_text_template
You can use Python formatting to display the position with different precision.

imperial_text_template
You can use Python formatting to display the position with different precision.

reference_type

```
0:G5x 1:tool 2:G92 3:Rotation around Z
```

12.3.6.21. DRO widget

The DRO widget is used to display the current axis position.

It has the following properties:

display_units_mm
Used to toggle the display units between metric and imperial. Default is False.

actual
Select actual (feedback) position or commanded position. Default is True.

diámetro
Display diameter for a lathe. Default is False.

mm_text_template
You can use Python formatting to display the position with different precision. Default is ” %10.3f”.

imperial_text_template
You can use Python formatting to display the position with different precision. Default is ” %9.4f”.

joint_number
Used to select which axis (technically which joint) is displayed. Default is 0.
On a trivialkins machine (mill, lathe, router) axis vs. joint number are:

```
0:X 1:Y 2:Z 3:A 4:B 5:C 6:U 7:V 8:W +
```

reference_type

- 0 = `absolute` (machine origin).
- 1 = `relative` (to current user coordinate origin - G5x).
- 2 = `distance-to-go` (relative to current user coordinate origin). El valor predeterminado es 0.

font_family
Specify the font family e.g. mono. Defaults to sans. If the font does not exist then the current system font will be used. Default is sans.

font_size
Specify the size of the font between 8 and 96. Default is 26.

font_weight
Specify the weight of the font. Select from lighter, normal, bold, or bolder. Default is bold.

unhomed_color
The text color when unhomed specified as a Gdk.RGBA color. Default is red, Gdk.RGBA(red=1.000000, green=0.000000, blue=0.000000, alpha=1.000000)

homed_color
The text color when homed specified as a Gdk.RGBA color. Default is green, Gdk.RGBA(red=0.000000, green=0.501961, blue=0.000000, alpha=1.000000)

Hints

- If you want the display to be right justified, set the Horizontal Alignment to `End`.
- The background of the widget is actually see through, so if you place it over an image, the DRO numbers will show on top of it with no background. There is a special technique to do this. See the animated function diagrams below.
- The DRO widget is a modified gtk label widget. As such, much of what can be done to a gtk label can be done to the DRO widget.
- The font properties may also be set from a css stylesheet which has the highest priority and will override values set by GObject properties.

There a couple ways to directly control the widget using Python.

Using GObject to set the above listed properties

```
[widget name].set_property(”display_units_mm”, True)
[widget name].set_property(”actual”, True)
[widget name].set_property(”diameter”, True)
[widget name].set_property(”mm_text_template”, ”%10.3f”)
[widget name].set_property(”imperial_text_template”, ”%9.4f”)
[widget name].set_property(”joint_number”, 3)
[widget name].set_property(”reference_type”, 3)
[widget name].set_property(”font_family”, ”mono”)
[widget name].set_property(”font_size”, 30)
[widget name].set_property(”font_weight”, ”bold”)
```

```
# it is easier to read colors by calling a function:
def str_to_rgba(color):
  c = Gdk.RGBA()
  c.parse(color)
  return c

[widget name].set_property(”unhomed_color”, str_to_rgba(”magenta”))
[widget name].set_property(”homed_color”, str_to_rgba(”cyan”))
```

Using a CSS stylesheet to set font properties

Colors may be specified in one of several formats, these would all specify the same color, red, *#ff0000, *rgb(255,0,0), or rgba(255,0,0,255).

Colors may be referenced either collectively:

```
.dro_unhomed {color: magenta}
.dro_homed {color: cyan}
```

or individually by widget name:

```
#[widget name].dro_unhomed {color: magenta}
#[widget name].dro_homed {color: cyan}
```

The other style properties need to be referenced by widget name:

```
#[widget name], #[widget name], #[widget name] {
    font-family: mono;
    font-size: 60px;
    font-weight: lighter;
}
```

Hay dos métodos de Python

```
[widget name].set_dro_inch()
[widget name].set_dro_metric()
```

12.3.6.22. Combi_DRO widget

The `Combi_DRO` widget is used to display the current, the relative axis position and the distance to go in one DRO.
By clicking on the DRO the Order of the DRO will toggle around.
In Relative Mode the actual coordinate system will be displayed.

`Combi_DRO` has the following properties:

joint_number
Used to select which axis (technically which joint) is displayed.
On a trivialkins machine (mill, lathe, router) axis/joint numbers are:

```
0:X 1:Y 2:Z etc.
```

actual
Select actual (feedback) or commanded position.

metric_units
Used to toggle the display units between metric and imperial.

auto_units
Units will toggle between metric and imperial according to the active G-code being G20 or G21.
Default is TRUE.

diámetro
Whether to display position as diameter or radius.
In diameter mode the DRO will display the joint value multiplied by 2.

mm_text_template
You can use Python formatting to display the position with different precision.
Default is ”%10.3f”.

imperial_text_template
You can use Python formatting to display the position with different precision.
Default is ”%9.4f”.

homed_color
El color de primer plano de los números DRO si la articulación regresa a su posición inicial.
El predeterminado es verde.

unhomed_color
El color de primer plano de los números DRO si la articulación no esta en su posición inicial.
El predeterminado es rojo.

abs_color
El color de fondo del DRO, si el DRO principal muestra coordenadas absolutas.
El predeterminado es azul.

rel_color
El color de fondo del DRO, si el DRO principal muestra coordenadas relativas.
El predeterminado es negro.

dtg_color
El color de fondo del DRO, si el DRO principal muestra distancia por recorrer.
El predeterminado es amarillo.

font_size
The font size of the big numbers, the small ones will be 2.5 times smaller.
The value must be an integer in the range of 8 to 96.
Default is 25.

toggle_readout
A left mouse click will toggle the DRO readout through the different modes [”Rel”, ”Abs”, ”DTG”]. By unchecking the box you can disable that behavior. The toggling can still be done with `[widget name].toggle_readout()`.
Value must be boolean.
Default is TRUE.

cycle_time
The time the DRO waits between two polls.
This setting should only be changed if you use more than 5 DRO at the same time, i.e. on a 6 axis config, to avoid that the DRO slows down the main application too much.
The value must be an integer in the range of 100 to 1000. FIXME unit=ms ?
Default is 150.

Using GObject to set the above listed properties:

```
[widget name].set_property(property, value)
```

There are several Python methods to control the widget:

- `[widget name].set_to_inch(state)`
  Sets the DRO to show imperial units.
  `state` = boolean (True or False)
  Default is FIXME.
- `[widget name].set_auto_units(state)`
  If True the DRO will change units according to active G-code (G20 / G21).
  `state` = boolean (True or False)
  Default is True.
- `[widget name].set_to_diameter(state)`
  If True the DRO will show the diameter not the radius, i.e., the axis value multiplied by 2 (specially needed for lathes).
  `state` = boolean (True or False)
  Default is False.
- `[widget name].toggle_readout()`
  Toggles the order of the DRO in the widget.
- `[widget name].change_axisletter(letter)`
  Changes the automatically given axis letter.
  Very useful to change an lathe DRO from *X* to *R* or *D*.
  `letter` = string
- `[widget name].get_order()`
  Returns the order of the DRO in the widget mainly used to maintain them consistent.
  The order will also be transmitted with the clicked signal.
  Returns a list containing the order.
- `[widget name].set_order(order)`
  Sets the order of the DRO, mainly used to maintain them consistent.
  `order` = list object, must be one of:
  - `[”Rel”, ”Abs”, ”DTG”]` (default)
  - `[”DTG”, ”Rel”, ”Abs”]`
  - `[”Abs”, ”DTG”, ”Rel”]`

- `[widget name].get_position()`
  Returns the position of the DRO as a list of floats.
  The order is independent of the order shown on the DRO and will be given as `[Absolute , relative , DTG]`.

  - `Absolute` = the machine coordinates, depends on the actual property will give actual or commanded position.
  - `Relative` = será las coordenadas del sistema de coordenadas real.
  - `DTG` = the distance to go.
    Will mostly be 0, as this function should not be used while the machine is moving, because of time delays.

The widget will emit the following signals:

- `clicked`
  This signal is emitted, when the user has clicked on the Combi_DRO widget.
  It will send the following data:
  - `widget` = widget object
    The widget object that sends the signal.
  - `joint_number` = integer
    The joint number of the DRO, where *0:X 1:Y 2:Z etc*.
  - `order` = list object
    The order of the DRO in that widget.
    The order may be used to set other Combi_DRO widgets to the same order with `[widget name].set_ord`
- `units_changed`
  This signal is emitted if the DRO units are changed.
  It will send the following data:
  - `widget` = widget object
    The widget object that sends the signal.
  - `metric_units` = boolean
    True if the DRO does display metric units, False in case of imperial display.
- `system_changed`
  This signal is emitted if the DRO units are changed.
  It will send the following data:
  - `widget` = widget object
    The widget object that sends the signal.
  - `system` = string
    The actual coordinate system. Will be one of G54 G55 G56 G57 G58 G59 G59.1 G59.2 G59.3 or Rel if none has been selected at all, what will only happen in Glade with no LinuxCNC running.

There are some information you can get through commands, which may be of interest for you:

- `[widget name].system`
  The actual system, as mentioned in the system_changed signal.
- `[widget name].homed`
  True if the joint is homed.

- `[widget name].machine_units`
  0 if Imperial, 1 if Metric.



Figura 12.45: Example: Three Combi_DRO in a window

```
X = Relative Mode
Y = Absolute Mode
Z = DTG Mode
```

12.3.6.23. IconView (File Select)

This is a touch screen friendly widget to select a file and to change directories.

`IconView` widget has the following properties:

icon_size
Sets the size of the displayed icon.
Allowed values are integers in the range from 12 to 96.
Default is 48.

start_dir
Sets the directory to start in when the widget is shown first time.
Must be a string, containing a valid directory path.
Default is ”/”.

jump_to_dir
Sets the ”jump to” directory, which is selected by the corresponding button in the bottom button list (the 5th button counting from the left).
Must be a string, containing a valid directory path.
Default is ”\~”.

filetypes
Sets the file filter for the objects to be shown.
Must be a string containing a comma separated list of extensions to be shown.
Default is ”ngc,py”.

sortorder
Sets the sorting order of the displayed icon.
Must be an integer value from 0 to 3, where:

- 0 = ASCENDING (sorted according to file names)
- 1 = DESCENDING (sorted according to file names)
- 2 = FOLDERFIRST (show the folders first, then the files), default
- 3 = FILEFIRST (show the files first, then the folders)

Using GObject to set the above listed properties:

```
[widget name].set_property(property,Value)
```

There are Python methods to control the widget:

- `[widget name].show_buttonbox(state)`
  If False the bottom button box will be hidden.
  This is helpful in custom screens, with special buttons layouts to not alter the layout of the GUI.
  Good example for that is GMOCCAPY.
  `state` = boolean (True or False).
  Default is True.
- `[widget name].show_filelabel(state)`
  If True the file label (between the IconView window and the bottom button box) will be shown.
  Hiding this label may save place, but showing it is very useful for debugging reasons.
  `state` = boolean (True or False).
  Default is True.
- `[widget name].set_icon_size(iconsize)`
  Sets the icon size.
  Must be an integer in the range from 12 to 96.
  Default = 48.
- `[widget name].set_directory(directory)`
  Allows to set an directory to be shown.
  `directory` = string (a valid file path).
- `[widget name].set_filetypes(filetypes)`
  Sets the file filter to be used.
  Only files with the given extensions will be shown.
  `filetypes` = string containing a comma separated list of extensions.
  Default = ”ngc,py”.
- `[widget name].get_selected()`
  Returns the path of the selected file, or `None` if a directory has been selected.
- `[widget name].refresh_filelist()`
  Refreshes the filelist.
  Needed if you add a file without changing the directory.

If the button box has been hidden, you can reach the functions of this button through its clicked signals like so:

```
[widget name].btn_home.emit(”clicked”)
[widget name].btn_jump_to.emit(”clicked”)
[widget name].btn_sel_prev.emit(”clicked”)
[widget name].btn_sel_next.emit(”clicked”)
[widget name].btn_get_selected.emit(”clicked”)
[widget name].btn_dir_up.emit(”clicked”)
[widget name].btn_exit.emit(”clicked”)
```

The widget will emit the following signals:

- `selected`
  This signal is emitted when the user selects an icon.
  It will return a string containing a file path if a file has been selected, or `None` if a directory has been selected.
- `sensitive`
  This signal is emitted when the buttons change their state from sensitive to not sensitive or vice versa.
  This signal is useful to maintain surrounding GUI synchronized with the button of the widget. See GMOCCAPY as example.
  It will return the **buttonname** and the new **state**:
  - `buttonname` is one of `btn_home`, `btn_dir_up`, `btn_sel_prev`, `btn_sel_next`, `btn_jump_to` or `btn_select`
  - `state` is a boolean and will be True or False.
- `exit`
  This signal is emitted when the exit button has been pressed to close the IconView.
  Mostly needed if the application is started as stand alone.

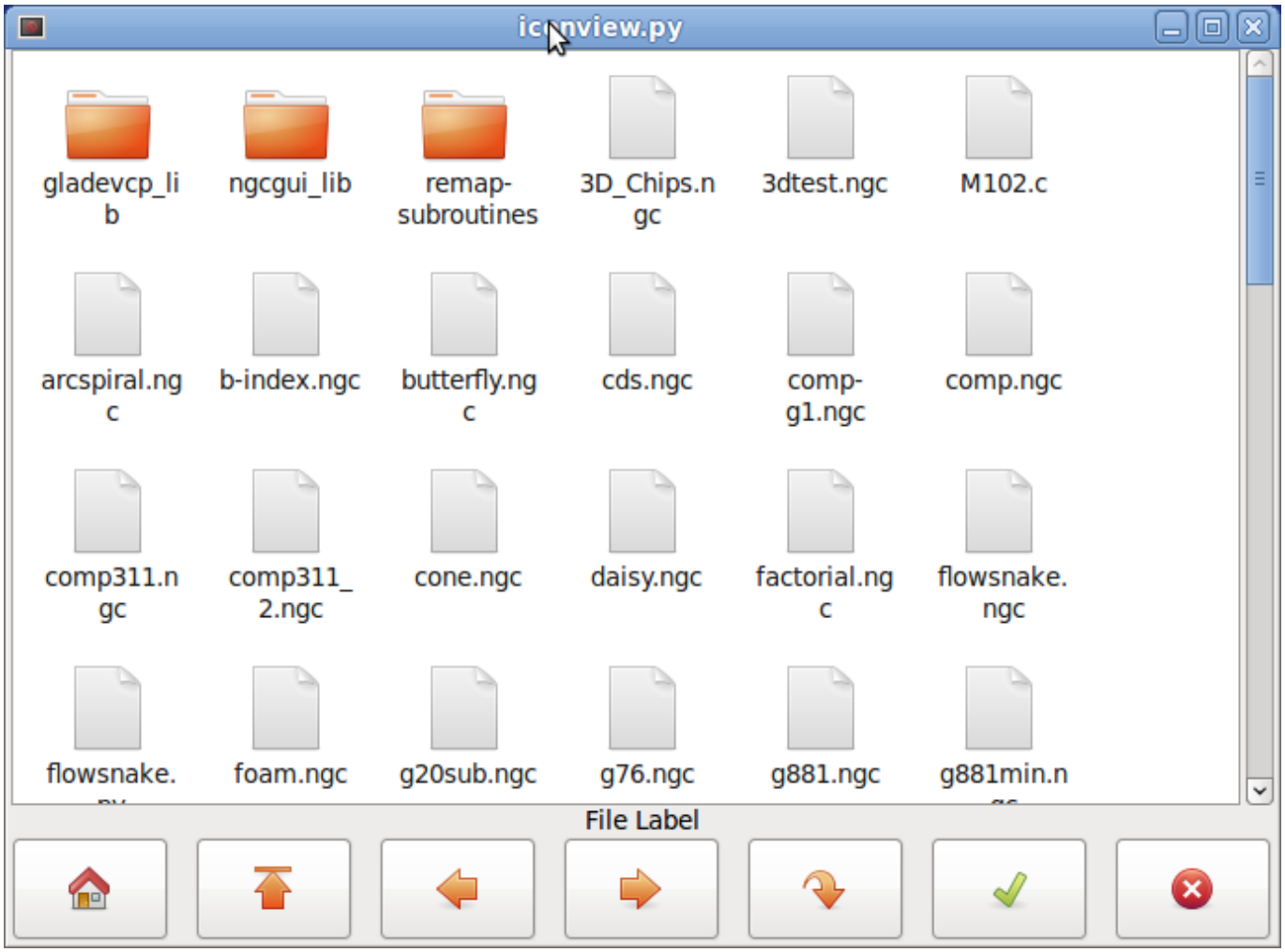


Figura 12.46: Iconview Example

12.3.6.24. Calculator widget

This is a simple calculator widget, that can be used for numerical input.
You can preset the display and retrieve the result or that preset value.

`calculator` has the following properties:

is_editable
: This allows the entry display to be typed into from a keyboard.

font
: This allows you to set the font of the display.

There a couple ways to directly control the widget using Python.

Using goobject to set the above listed properties:

```
[widget name].set_property(”is_editable”,True)
[widget name].set_property(”font”,”sans 25”)
```

There are Python methods:

- `[widget name].set_value(2.5)`
  This presets the display and is recorded.
- `[widget name].set_font(”sans 25”)`
- `[widget name].set_editable(True)`
- `[widget name].get_value()`
  Returns the calculated value - a float.
- `[widget name].set_editable(True)`
- `[widget name].get_preset_value()`
  Returns the recorded value: a float.

12.3.6.25. Tooleditor widget

This is a `tooleditor` widget for displaying and modifying a tool file.
If in lathe mode, it will display wear offsets and tool offsets separately.
Wear offsets are designated by tool number above 10000 (Fanuc style).
It checks the current file once a second to see if LinuxCNC updated it.

---

nota
LinuxCNC requires remapping of tool calls to actually use wear offsets.

`tooleditor` has the following properties:

font
: Display font to use

hide_columns
: This will hide the given columns.
The columns are designated (in order) as such: `s,t,p,x,y,z,a,b,c,u,v,w,d,i,j,q`.
You can hide any number of columns including the select and comments.

lathe_display_type
Show lathe format

There a couple ways to directly control the widget using Python.

Using goobject to set the above listed properties:

```
[widget name].set_properties('hide_columns','uvwijq')
```

This would hide the uvwij and q columns and show all others.

There are Python methods:

- `[widget name].set_visible("ijq",False)`
  Would hide ij and Q columns and leave the rest as they were.
- `[widget name].set_filename(path_to_file)`
  Sets and loads the tool file.
- `[widget name].reload(None)`
  Reloads the current toolfile.
- `[widget name].set_font('sans 16,tab='1')`
  Sets the (Pango) font on the Tab, column title, and tool data.
  The `all_offsets`, `wear_offsets`, `tool_offsets` can be set at the same time by adding 1, 2 and/or 3 to the tab string.
  Default is all the tabs set.
- `[widget name].set_title_font('sans 16,tab='1')`
  Sets the (Pango) font on the column titles only.
  The `all_offsets`, `wear_offsets`, `tool_offsets` can be set at the same time by adding 1, 2 and/or 3 to the tab string.
  Default is all the tabs set.
- `[widget name].set_tab_font('sans 16,tab='1')`
  Sets the (Pango) font on the tabs only.
  The `all_offsets`, `wear_offsets`, `tool_offsets` can be set at the same time by adding 1, 2 and/or 3 to the tab string.
  Default is all the tabs set.
- `[widget name].set_col_visible("abcUVW", False, tab='1')`
  This would hide (False) the abcuvw columns on tab 1 (all_offsets)
- `[widget name].set_lathe_display(value)`
  Hides or shows the wear and tool offset tabs used for lathes
- `[widget name].get_toolinfo(toolnum)`
  Returns the tool information array of the requested toolnumber or current tool if no tool number is specified.
  Returns None if tool not found in table or if there is no current tool.
- `[widget name].hide_buttonbox(self, True)`
  Convenience method to hide buttons.
  You must call this after show_all().
- `[widget name].get_selected_tool()`
  Return the user selected (highlighted) tool number.
- `[widget name].set_selected_tool(toolnumber)`
  Selects (highlights) the requested tool.

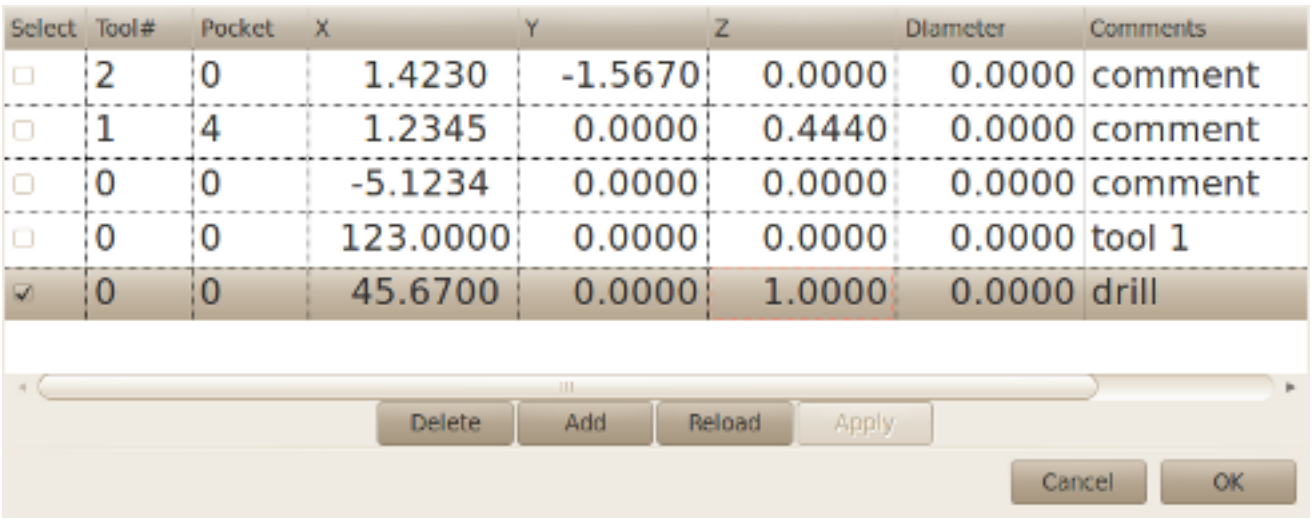


Figura 12.47: Tooleditor Example

12.3.6.26. Offsetpage

The `Offsetpage` widget is used to display/edit the offsets of all the axes.
It has convenience buttons for zeroing G92 and Rotation-Around-Z offsets.
It will only allow you to select the edit mode when the machine is on and idle.
You can directly edit the offsets in the table at this time. Unselect the edit button to allow the `OffsetPage` to reflect changes.

It has the following properties:

display_units_mm
Display in metrice units

hide_columns
A no-space list of columns to hide. The columns are designated (in order) as such: `xyzabcuvwt`.
You can hide any of the columns.

hide_rows
A no-space list of rows to hide.
The rows are designated (in order) as such: `0123456789abc`.
You can hide any of the rows.

font
Sets text font type and size.

highlight_color
When editing this is the highlight color.

foreground_color
When `OffsetPage` detects an active user coordinate system it will use this color for the text.

mm_text_template
You can use Python formatting to display the position with different precision.

imperial_text_template
You can use Python formatting to display the position with different precision.

There a couple ways to directly control the widget using Python.

Using goobject to set the above listed properties:

```
[widget name].set_property(”highlight_color”,gdk.Color(’blue’))
[widget name].set_property(”foreground_color”,gdk.Color(’black’))
[widget name].set_property(”hide_columns”,”xyzabcuvwt”)
[widget name].set_property(”hide_rows”,”123456789abc”)
[widget name].set_property(”font”,”sans 25”)
```

There are Python methods to control the widget:

- `[widget name].set_filename(”../../../configs/sim/gscreen/gscreen_custom/sim.var”)`
- `[widget name].set_col_visible(”Yabuvw”,False)`
- `[widget name].set_row_visible(”456789abc”,False)`
- `[widget name].set_to_mm()`
- `[widget name].set_to_inch()`
- `[widget name].hide_button_box(True)`
- `[widget name].set_font(”sans 20”)`
- `[widget name].set_highlight_color(”violet”)`
- `[widget name].set_foreground_color(”yellow”)`
- `[widget name].mark_active(”G55”)`
  Allows you to directly set a row to highlight, e.g., in case you wish to use your own navigation controls. See the chapter on GMOCCAPY.
- `[widget name].selection_mask = (”Tool”,”Rot”,”G5x”)`
  These rows are NOT selectable in edit mode.
- `[widget name].set_names([[’G54’,’Default’],[”G55”,”Vice1”],[’Rot’,’Rotational’]])`
  This allows you to set the text of the *T* column of each/any row.
  This is a list of a list of offset-name/user-name pairs.
  The default text is the same as the offset name.
- `[widget name].get_names()`
  This returns a list of a list of row-keyword/user-name pairs.
  The user name column is editable, so saving this list is user friendly.
  See `set_names` above.

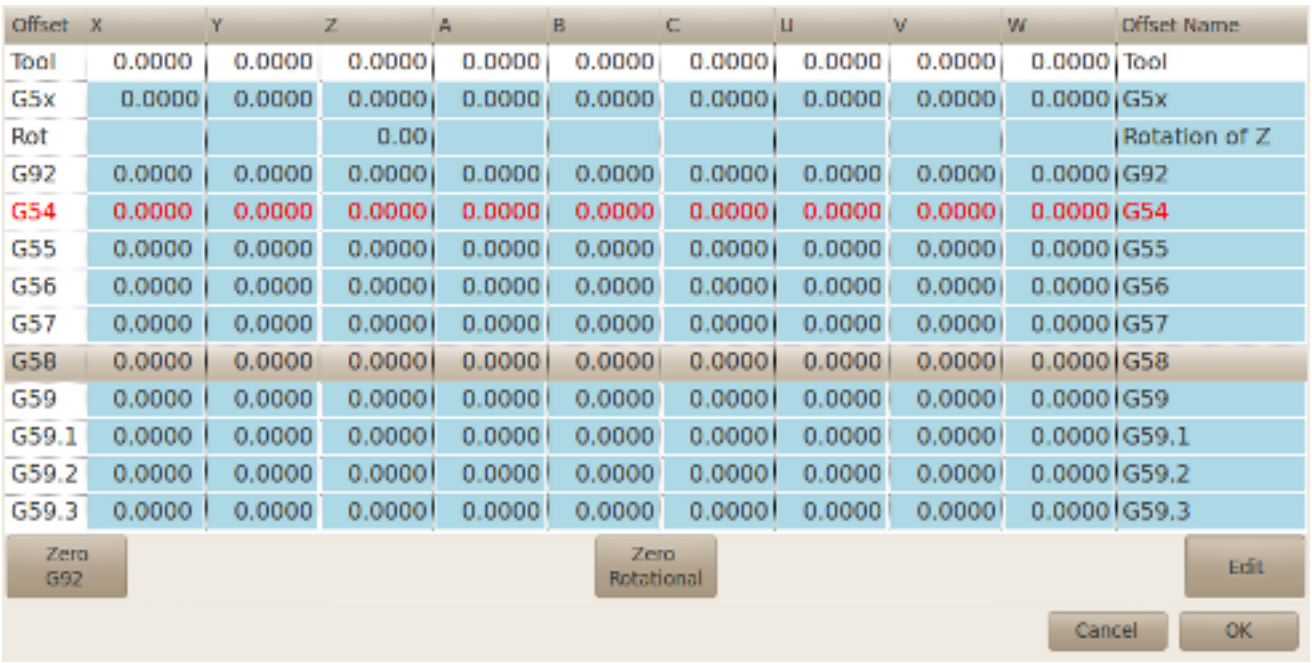


Figura 12.48: Offsetpage Example

12.3.6.27. HAL_sourceview widget

This is for displaying and simple editing of G-code. It looks for `.ngc` highlighting specs in `~/share/gtksource`
The current running line will be highlighted.

With external Python glue code it can:

- Search for text, undo and redo changes.
- Be used for program line selection.

There are Python methods to control the widget:

- `[widget name].redo()`
  Redo one level of changes.
- `[widget name].undo()`
  Undo one level of changes
- `[widget name].text_search(direction=True,mixed_case=True,text='G92')`
  Searches forward (direction = True) or backward,
  Searches with mixed case (mixed_case = True) or exact match
- `[widget name].set_line_number(linenumber)`
  Sets the line to highlight.
  Uses the sourceview line numbers.
- `[widget name].get_line_number()`
  Returns the currently highlighted line.
- `[widget name].line_up()`
  Moves the highlighted line up one line.

- `[widget name].line_down()`
  Moves the highlighted line down one line.
- `[widget name].load_file('filename')`
  Loads a file.
  Using None (not a filename string) will reload the same program.
- `[widget name].get_filename()`
  FIXME description

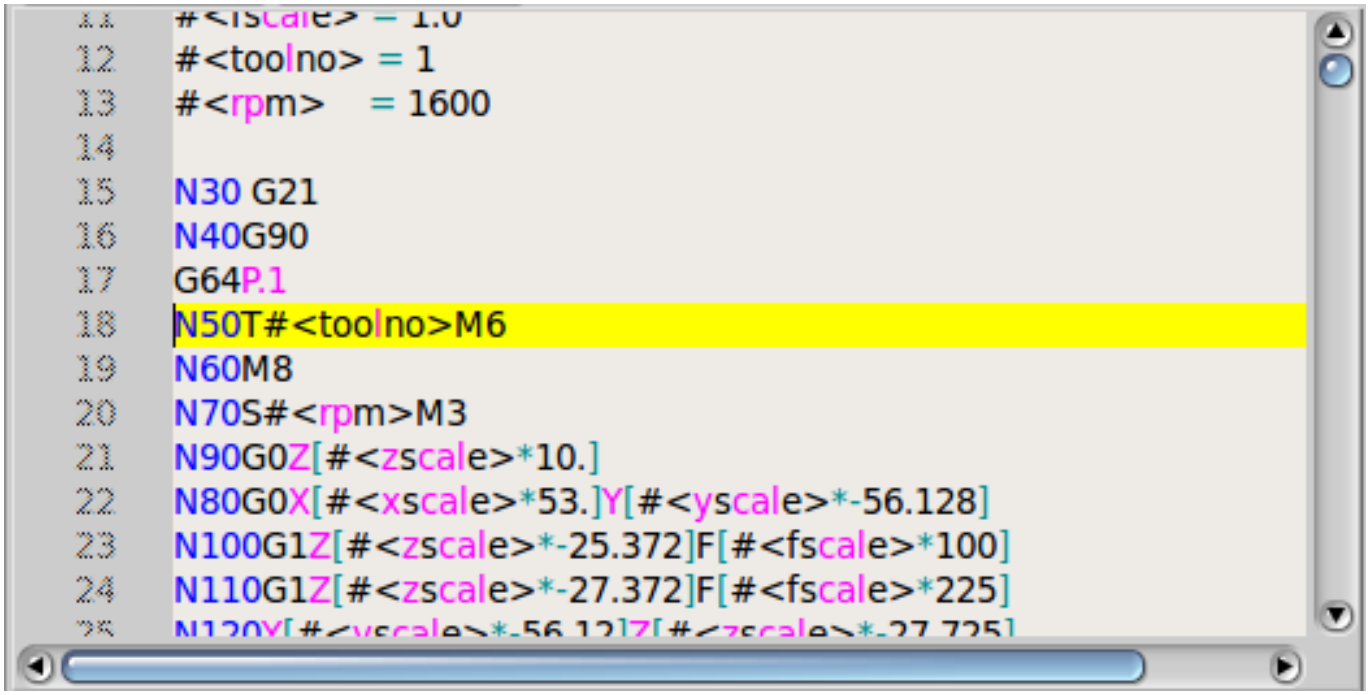


Figura 12.49: Sourceview Example

12.3.6.28. MDI history

This is for displaying and entering MDI codes.
It will be automatically grayed out when MDI is not available, e.g., during E-stop and program running.

font_size_tree
Integer value between 8 and 96.
Will modify the default font size of the treeview to the selected value.

font_size_entry
Integer value between 8 and 96.
Will modify the default font size of the entry to the selected value.

use_double_click
Boolean, True enables the mouse double click feature and a double click on an entry will submit that command.
It is not recommended to use this feature on real machines, as a double click on a wrong entry may cause dangerous situations.

Using goobject to set the above listed properties:

```
[widget name].set_property("font_size_tree",10)
[widget name].set_property("font_size_entry",20)
[widget name].set_property("use_double_click",False)
```

12.3.6.29. Animated function diagrams: HAL widgets in a bitmap

For some applications it might be desirable to have a background image - like a functional diagram - and position widgets at appropriate places in that image. A good combination is setting a bitmap background image, like from a .png file, making the GladeVCP window fixed-size, and use the Glade Fixed widget to position widgets on this image. The code for the below example can be found in `configs/apps/gladevcp/animated-backdrop`:

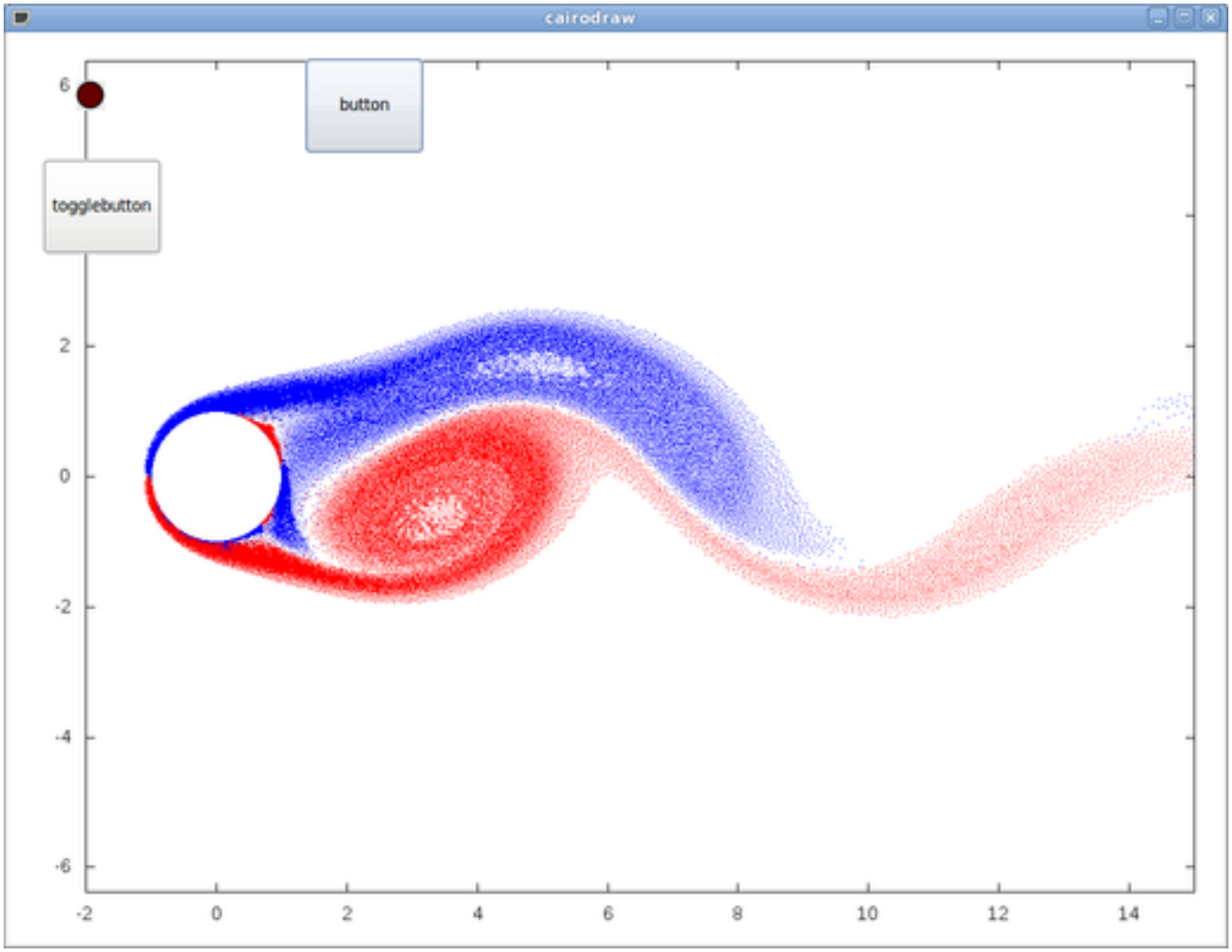


Figura 12.50: HAL widgets in a bitmap Example

12.3.7. Action Widgets Reference

GladeVCP includes a collection of "canned actions" called **VCP Action Widgets** for the Glade user interface editor.

nota
Other than HAL widgets, which interact with HAL pins, VCP Actions interact with LinuxCNC and the G-code interpreter.

VCP Action Widgets are derived from the `Gtk.Action` widget.

The Action widget in a nutshell:

- It is an object available in Glade
- It has no visual appearance by itself
- Its purpose: Associate a visible, sensitive UI component like menu, toolbutton, button with a command. See these widget's *General→Related→Action* property.
- The "canned action" will be executed when the associated UI component is triggered (button press, menu click..).
- It provides an easy way to execute commands without resorting to Python programming.

The appearance of VCP Actions in Glade is roughly as follows:

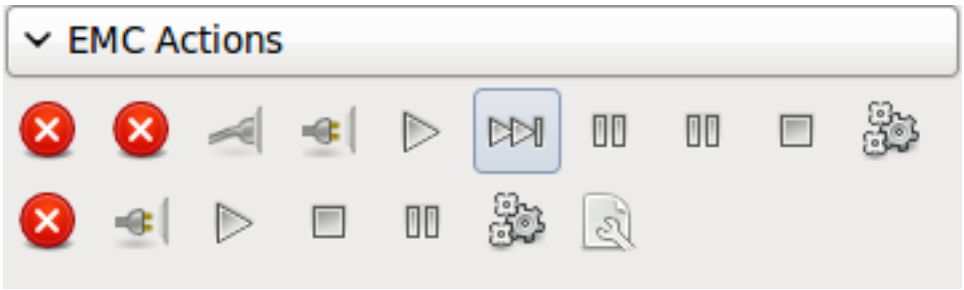


Figura 12.51: Action Widgets

Tooltip hovers provide a description.

12.3.7.1. VCP Action Widgets

VCP Action widgets are one-shot type widgets. They implement a single action and are for use in simple buttons, menu entries or radio/check groups.

12.3.7.2. VCP Action Python

This widget is used to execute small arbitrary Python code.

The command string may use special keywords to access important functions.

- *ACTION* for access to the ACTION command library.
- `GSTAT` for access to the Gstat status message library.
- *INFO* for access to collected data from the INI file.
- *HAL* para acceso al módulo HAL linuxcnc Python.
- `STAT` para acceso al estado bruto de LinuxCNC por medio del módulo LinuxCNC Python.

- `CMD` para acceso a los comandos de LinuxCNC por medio del módulo LinuxCNC Python.
- `EXT` para acceder a las funciones del archivo manejador si esta disponible.
- `linuxcnc` para acceso al módulo LinuxCNC Python.
- `self` for access to the widget instance.
- *dir* for access to the handlers list of attributes.

Hay opciones para

- select when the widget will be active,
- asigna el modo antes de ejecutar el comando.

Example command to just print a message to the terminal:

```
print('action activated')
```

Example command to set the machine to off state:

```
CMD.state(linuxcnc.STATE_OFF)
```

Example command to call a handler function that passes data:

```
EXT.on_button_press(self, 100)
```

You can use a semicolon to separate multiple commands;

```
print('Set Machine Off');CMD.state(linuxcnc.STATE_OFF)
```

Se puede encontrar mas información sobre INFO y ACTION aquí: Módulos de librerías GladeVCP.

Se puede encontrar mas información sobre GStat aquí: [?].

12.3.7.3. VCP ToggleAction widgets

These are **bi-modal** widgets. They implement two actions or use a second (usually `pressed`) state to indicate that currently an action is running. Toggle actions are aimed for use in `ToggleButtons`, `ToggleToolButtons` or toggling menu items. A simplex example is the `ESTOP` toggle button.

Currently the following widgets are available:

- The `ESTOP` toggle sends `ESTOP` or `ESTOP_RESET` commands to LinuxCNC depending on its state.
- The `ON/OFF` toggle sends `STATE_ON` and `STATE_OFF` commands.
- `Pause/Resume` sends `AUTO_PAUSE` or `AUTO_RESUME` commands.

The following toggle actions have only one associated command and use the `pressed` state to indicate that the requested operation is running:

- The `Run` toggle sends an `AUTO_RUN` command and waits in the `pressed` state until the interpreter is idle again.
- The `Stop` toggle is inactive until the interpreter enters the active state (is running G-code) and then allows user to send `AUTO_ABORT` command.
- The `MDI` toggle sends given MDI command and waits for its completion in `pressed` inactive state.

12.3.7.4. The Action_MDI Toggle and Action_MDI widgets

These widgets provide a means to execute arbitrary MDI commands.
The `Action_MDI` widget does not wait for command completion as the `Action_MDI` Toggle does, which remains disabled until command complete.

12.3.7.5. A simple example: Execute MDI command on button press

`configs/apps/gladevcp/mdi-command-example/whoareyou.ui` is a Glade UI file which conveys the basics:

1. Open it in Glade and study how it is done.
2. Start AXIS, and then start this from a terminal window with `gladevcp whoareyou.ui`.
3. See the `hal_action_mdi1` action and its `MDI command` property - this just executes `(MSG, "Hi, I'm an VCP_Action_MDI")` so there should be a message popup in AXIS like so:

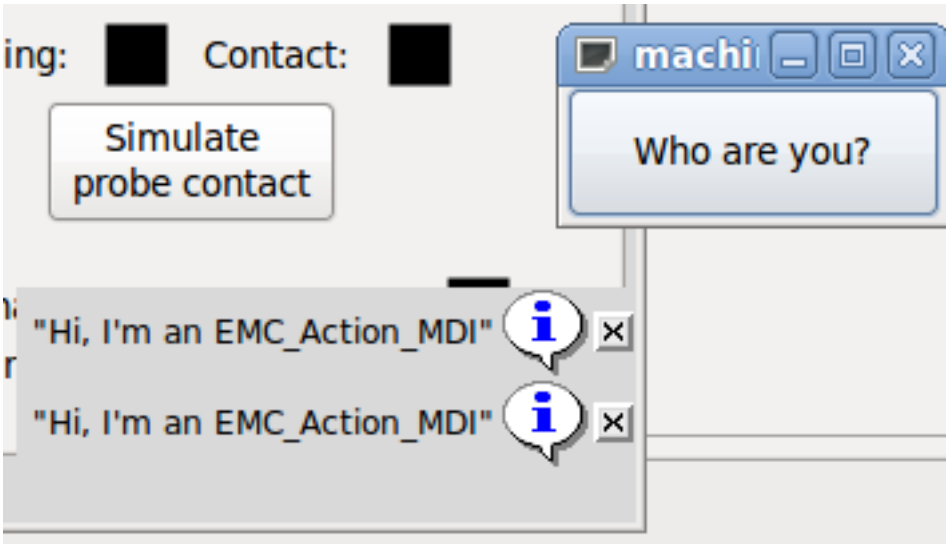


Figura 12.52: Action_MDI Simple Example

You'll notice that the button associated with the Action_MDI action is grayed out if the machine is off, in E-Stop or if the interpreter is running. It will automatically become active when the machine is turned on and out of E-Stop, and the program is idle.

12.3.7.6. Parameter passing with Action_MDI and ToggleAction_MDI widgets

Optionally, `MDI command` strings may have parameters substituted before they are passed to the interpreter. Parameters currently may be names of HAL pins in the GladeVCP component. This is how it works:

- assume you have a *HAL SpinBox* named `speed`, and you want to pass its current value as a parameter in an MDI command.
- The HAL SpinBox will have a float-type HAL pin named speed-f (see HalWidgets description).
- To substitute this value in the MDI command, insert the HAL pin name enclosed like so: `${pin-name}`

- for the above HAL SpinBox, we could use `(MSG, "The speed is: ${speed-f}")` just to show what's happening.

The example UI file is `configs/apps/gladevcp/mdi-command-example/speed.ui`. Here's what you get when running it:

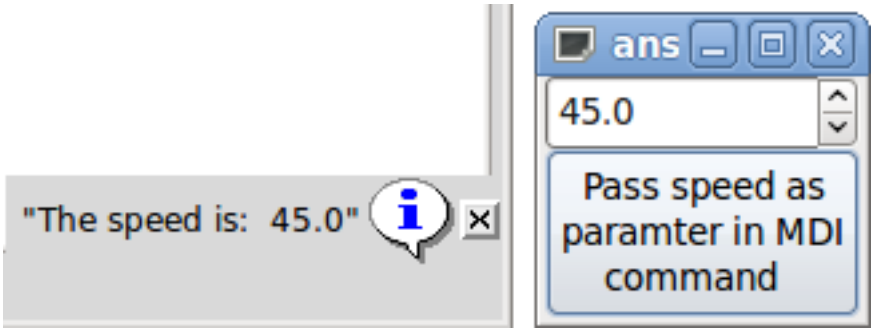


Figura 12.53: Action_MDI Parameter Passing Example

12.3.7.7. An advanced example: Feeding parameters to an O-word subroutine

It's perfectly OK to call an O-word subroutine in an MDI command, and pass HAL pin values as actual parameters. An example UI file is in `configs/apps/gladevcp/mdi-command-example/owordsub.ui`.

Place `nc_files/gladevcp_lib/oword.ngc` so AXIS can find it, and run `gladevcp owordsub.ui` from a terminal window. This looks like so:

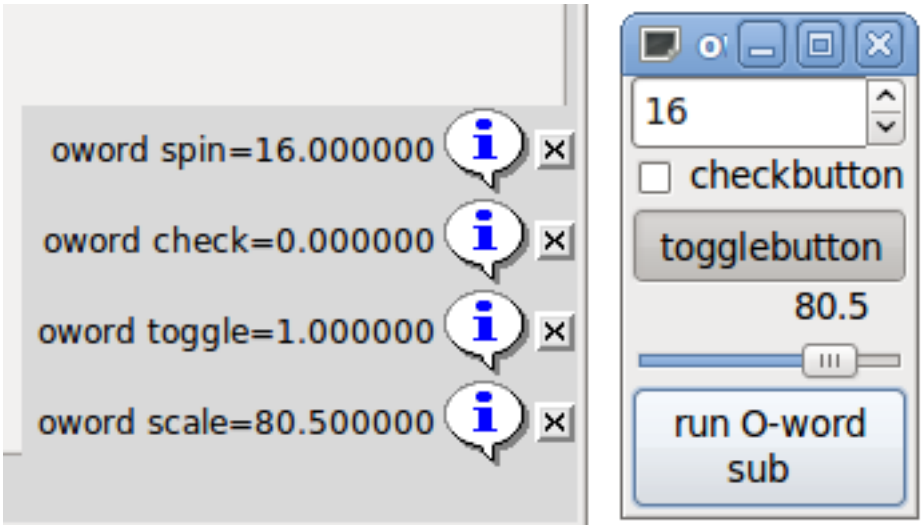


Figura 12.54: Action_MDI Advanced Example

12.3.7.8. Preparing for an MDI Action, and cleaning up afterwards

The LinuxCNC G-code interpreter has a single global set of variables, like feed, spindle speed, relative/absolute mode and others. If you use G-code commands or O-word subs, some of these variables might get changed by the command or subroutine - for example, a probing subroutine will very likely set the feed value quite low. With no further precautions, your previous feed setting will be overwritten by the probing subroutine's value.

To deal with this surprising and undesirable side effect of a given O-word subroutine or G-code statement executed with an LinuxCNC ToggleAction_MDI, you might associate pre-MDI and post-MDI handlers with a given LinuxCNC ToggleAction_MDI. These handlers are optional and provide a way to save any state before executing the MDI Action, and to restore it to previous values afterwards. The signal names are `mdi-command-start` and `mdi-command-stop`; the handler names can be set in Glade like any other handler.

Here's an example how a feed value might be saved and restored by such handlers (note that LinuxCNC command and status channels are available as `self.linuxcnc` and `self.stat` through the VCP_ActionBase class):

```
    def on_mdi_command_start(self, action, userdata=None):
        action.stat.poll()
        self.start_feed = action.stat.settings[1]

    def on_mdi_command_stop(self, action, userdata=None):
        action.linuxcnc.mdi('F%.1f' % (self.start_feed))
        while action.linuxcnc.wait_complete() == -1:
            pass
```

Only the `Action_MDI` Toggle widget supports these signals.

---

nota
In a later release of LinuxCNC, the new M-codes M70-M72 will be available. They will make saving state before a subroutine call, and restoring state on return much easier.

12.3.7.9. Using the LinuxCNC Stat object to deal with status changes

Many actions depend on LinuxCNC status - is it in manual, MDI or auto mode? Is a program running, paused or idle? You cannot start an MDI command while a G-code program is running, so this needs to be taken care of. Many LinuxCNC actions take care of this themselves, and related buttons and menu entries are deactivated when the operation is currently impossible.

When using Python event handlers - which are at a lower level than Actions - one needs to take care of dealing with status dependencies oneself. For this purpose, there's the LinuxCNC Stat widget: to associate LinuxCNC status changes with event handlers.

LinuxCNC Stat has no visible component - you just add it to your UI with Glade. Once added, you can associate handlers with its following signals:

- state-related:
  - `state-estop`: emitted when E-Stop condition occurs,
  - `state-estop-reset`: emitted when machine is reset,
  - `state-on`: emitted when machine is turned on,
  - `state-off`: emitido cuando la máquina es apagada.
- mode-related:
  - `mode-manual`: emitted when LinuxCNC enters manual mode,
  - `mode-mdi`: emitted when LinuxCNC enters MDI mode,
  - `mode-auto`: emitted when LinuxCNC enters automatic mode,
- interpreter-related: emitted when the G-code interpreter changes into that mode
  - `interp-run`

  - `interp-idle`
  - `interp-paused`
  - `interp-reading`
  - `interp-waiting`
  - `file-loaded`
  - `line-changed`
- homing-related: emitted when LinuxCNC is homed or not
  - `all-homed`
  - `not-all-homed`

12.3.8. GladeVCP Programming

12.3.8.1. User Defined Actions

Most widget sets, and their associated user interface editors, support the concept of callbacks, i.e. functions in user-written code which are executed when *something happens* in the UI - events like mouse clicks, characters typed, mouse movement, timer events, window hiding and exposure and so forth.

HAL output widgets typically map input-type events like a button press to a value change of the associated HAL pin by means of such a - predefined - callback. Within PyVCP, this is really the only type of event handling supported - doing something more complex, like executing MDI commands to call a G-code subroutine, is not supported.

Within GladeVCP, HAL pin changes are just one type of the general class of events (called signals) in GTK+. Most widgets may originate such signals, and the Glade editor supports associating such a signal with a Python method or function name.

If you decide to use user-defined actions, your job is to write a Python module whose class methods - or in the simple case, just functions - can be referred to in Glade as event handlers. GladeVCP provides a way to import your module(s) at startup and will automatically link your event handlers with the widget signals as set in the Glade UI description.

12.3.8.2. Core Library

There are three libraries of functions that can be used to help program GladeVCP.

- *Info*: collects details from the INI file.
- *Action*: A collection of functions to change LinuxCNC states.
- *Status*: Reports the state of LinuxCNC. It wraps around *Gstat*.

Importing and instantiating the libraries:

```
from gladevcp.core import Info, Action

ACTION = Action()
INFO = Info()
```

Usando la funciones de librería:

```
print(INFO.MACHINE_IS_METRIC)
ACTION.SET_ERROR_MESSAGE('Something went wrong')
```

More information can be found here: GladeVCP Libraries modules. There is a sample configuration that demonstrates using the core library with GladeVCP's action Python widgets and with a Python handler file. Try loading *sim/axis/gladevcp/gladevcp_panel_tester*.

12.3.8.3. An example: adding custom user callbacks in Python

This is just a minimal example to convey the idea - details are laid out in the rest of this section.

GladeVCP can not only manipulate or display HAL pins, you can also write regular event handlers in Python. This could be used, among others, to execute MDI commands. Here's how you do it:

Write a Python module like so and save as e.g. handlers.py:

```
nhits = 0
def on_button_press(gtkobj,data=None):
    global nhits
    nhits += 1
    gtkobj.set_label("hits: %d" % nhits)
```

In Glade, define a button or HAL button, select the *Signals* tab, and in the GtkButton properties select the *pressed* line. Enter *on_button_press* there, and save the Glade file.

Then add the option *-u handlers.py* to the GladeVCP command line. If your event handlers are spread over several files, just add multiple *-u <pyfilename>* options.

Now, pressing the button should change its label since it is set in the callback function.

What the +-u+ flag does: all Python functions in this file are collected and setup as potential callback handlers for your Gtk widgets - they can be referenced from Glade *Signals* tabs. The callback handlers are called with the particular object instance as parameter, like the GtkButton instance above, so you can apply any GtkButton method from there.

Or do some more useful stuff, like calling an MDI command!

12.3.8.4. HAL value change events

HAL input widgets, like a LED, automatically associate their HAL pin state (on/off) with the optical appearance of the widget (LED lit/dark).

Beyond this built-in functionality, one may associate a change callback with any HAL pin, including those of predefined HAL widgets. This fits nicely with the event-driven structure of a typical widget application: Every activity, be it mouse click, key, timer expired, or the change of a HAL pin's value, generates a callback and is handled by the same orthogonal mechanism.

For user-defined HAL pins not associated with a particular HAL widget, the signal name is *value-changed*. See the Adding HAL pins section below for details.

HAL widgets come with a pre-defined signal called *hal-pin-changed*. See the HAL Widgets section for details.

12.3.8.5. Programming model

The overall approach is as follows:

- Design your UI with Glade, and set signal handlers where you want actions associated with a widget.
- Write a Python module which contains callable objects (see *handler models* below).
- Pass your module's path name to GladeVCP with the *-u <module>* option.
- GladeVCP imports the module, inspects it for signal handlers and connects them to the widget tree.
- The main event loop is run.

For simple tasks it is sufficient to define functions named after the Glade signal handlers. These will be called when the corresponding event happens in the widget tree. Here's a trivial example - it assumes that the *pressed* signal of a Gtk Button or HAL Button is linked to a callback called *on_button_press*:

```
nhits = 0
def on_button_press(gtkobj,data=None):
    global nhits
    nhits += 1
    gtkobj.set_label("hits: %d" % nhits)
```

Add this function to a Python file and run as follows:

```
gladevcp -u <myhandler>.py mygui.ui
```

Note communication between handlers has to go through global variables, which does not scale well and is positively un-pythonic. This is why we came up with the class-based handler model.

The idea here is: Handlers are linked to class methods. The underlying class(es) are instantiated and inspected during GladeVCP startup and linked to the widget tree as signal handlers. So the task now is to write:

- One or more several class definition(s) with one or several methods, in one module or split over several modules,
- a function *get_handlers* in each module which will return a list of class instances to GladeVCP - their method names will be linked to signal handlers.

Here is a minimum user-defined handler example module:

```
class MyCallbacks :
    def on_this_signal(self,obj,data=None):
        print("this_signal happened, obj=",obj)

def get_handlers(halcomp,builder,useropts):
    return [MyCallbacks ()]
```

Now, *on_this_signal* will be available as signal handler to your widget tree.

For GladeVCP panel which respond to HAL inputs it may be important that the handler code can tell that the GladeVCP panel is currently active and displayed. For example a panel inside the Touchy interface might well need to perform an action when the switch connected to touchy.cycle-start is operated (in the same way that the native tabs respond differently to the same button).
To make this possible, a signal is sent from the GUI (at the time of writing, only Touchy) to the embedded tab. The signal is of type "Gladevcp" and the two messages sent are "Visible" and "Hidden". (Note that the signals have a fixed length of 20 characters so only the first characters should be used in any comparison, hence the [:7] below.) A sample handler for these signals is:

```
    # This catches our messages from another program
    def event(self,w,event):
        print(event.message_type,event.data)
        if event.message_type == 'Gladevcp':
            if event.data[:7] == 'Visible':
                self.active = True
            else:
                self.active = False

    # connect to client-events from the host GUI
    def on_map_event(self, widget, data=None):
        top = widget.get_toplevel()
        print("map event")
        top.connect('client-event', self.event)
```

If during module inspection GladeVCP finds a function `get_handlers`, it calls it as follows:

```
get_handlers(halcomp,builder,useropts)
```

The arguments are:

- `halcomp` - refers to the HAL component under construction,
- `builder` - widget tree - result of reading the UI definition (either referring to a GtkBuilder or libglade-type object),
- `useropts` - a list of strings collected from the GladeVCP command line `-U <useropts>` option.

GladeVCP then inspects the list of class instances and retrieves their method names. Qualifying method names are connected to the widget tree as signal handlers. Only method names which do not begin with an _ (underscore) are considered.

Note that regardless whether you're using the libglade or the new GtkBuilder format for your Glade UI, widgets can always be referred to as `builder.get_object(<widgetname>)`. Also, the complete list of widgets is available as `builder.get_objects()` regardless of UI format.

12.3.8.6. Initialization sequence

It is important to know in which state of affairs your `get_handlers()` function is called so you know what is safe to do there and what not. First, modules are imported and initialized in command line order. After successful import, `get_handlers()` is called in the following state:

- The widget tree is created, but not yet realized (no toplevel `window.show()` has been executed yet).
- The halcomp HAL component is set up and all HAL widgets' pins have already been added to it.
- It is safe to add more HAL pins because `halcomp.ready()` has not yet been called at this point, so you may add your own pins, for instance in the class `__init__()` method.

Once all modules have been imported and method names extracted, the following steps happen:

- All qualifying method names will be connected to the widget tree with `connect_signals()/signal_autoc` (depending on the type of UI imported - GtkBuilder vs the old libglade format).
- The HAL component is finalized with halcomp.ready().
- If a window ID was passed as argument, the widget tree is re-parented to run in this window, and Glade's toplevel window1 is abandoned (see FAQ).
- If a HAL command file was passed with `-H halfile`, it is executed with halcmd.
- The Gtk main loop is run.

So when your handler class is initialized, all widgets are existent but not yet realized (displayed on screen). And the HAL component isn't ready as well, so its unsafe to access pins values in your `__init__()` method.

If you want to have a callback to execute at program start after it is safe to access HAL pins, then a connect a handler to the realize signal of the top level window1 (which might be its only real purpose). At this point GladeVCP is done with all setup tasks, the HAL file has been run, and GladeVCP is about to enter the Gtk main loop.

12.3.8.7. Multiple callbacks with the same name

Within a class, method names must be unique. However, it is OK to have multiple class instances passed to GladeVCP by get_handlers() with identically named methods. When the corresponding signal occurs, these methods will be called in definition order - module by module, and within a module, in the order class instances are returned by `get_handlers()`.

12.3.8.8. The GladeVCP `-U <useropts>` flag

Instead of extending GladeVCP for any conceivable option which could potentially be useful for a handler class, you may use the -U <useroption> flag (repeatedly if you wish). This flag collects a list of <useroption> strings. This list is passed to the get_handlers() function (useropts argument). Your code is free to interpret these strings as you see fit. An possible usage would be to pass them to the Python exec function in your `get_handlers()` as follows:

```
debug = 0
...
def get_handlers(halcomp,builder,useropts):
    ...
    global debug # assuming there's a global var
    for cmd in useropts:
        exec cmd in globals()
```

This way you can pass arbitrary Python statements to your module through the `gladevcp -U` option, for example:

```
gladevcp -U debug=42 -U "print 'debug=%d' % debug" ...
```

This should set debug to 2 and confirm that your module actually did it.

12.3.8.9. Persistent variables in GladeVCP

An annoying aspect of GladeVCP in its earlier form and PyVCP is the fact that you may change values and HAL pins through text entry, sliders, spin boxes, toggle buttons, etc., but their settings are not saved and restored at the next run of LinuxCNC - they start at the default value as set in the panel or widget definition.

GladeVCP has an easy-to-use mechanism to save and restore the state of HAL widgets, and program variables (in fact any instance attribute of type int, float, bool or string).

This mechanism uses the popular INI file format to save and reload persistent attributes.

Persistence, program versions and the signature check Imagine renaming, adding or deleting widgets in Glade: An .INI file lying around from a previous program version, or an entirely different user interface, would be not be able to restore the state properly since variables and types might have changed.

GladeVCP detects this situation by a signature which depends on all object names and types which are saved and to be restored. In the case of signature mismatch, a new INI file with default settings is generated.

12.3.8.10. Using persistent variables

If you want any of Gtk widget state, HAL widgets output pin's values and/or class attributes of your handler class to be retained across invocations, proceed as follows:

- Importa el módulo `gladevcp.persistence`.

- Decide which instance attributes, and their default values you want to have retained, if any.
- Decide which widgets should have their state retained.
- Describe these decisions in your handler class' `\__init()__` method through a nested dictionary as follows:

```
def __init__(self, halcomp,builder,useropts):
    self.halcomp = halcomp
    self.builder = builder
    self.useropts = useropts
    self.defaults = {
        # the following names will be saved/restored as method attributes
        # the save/restore mechanism is strongly typed - the variables type will be derived ↩
            from the type of the
        # initialization value. Currently supported types are: int, float, bool, string
        IniFile.vars : { 'nhits' : 0, 'a': 1.67, 'd': True ,'c' : "a string"},
        # to save/restore all widget's state which might remotely make sense, add this:
        IniFile.widgets : widget_defaults(builder.get_objects())
        # a sensible alternative might be to retain only all HAL output widgets' state:
        # IniFile.widgets: widget_defaults(select_widgets(self.builder.get_objects(),  ↩
            hal_only=True,output_only = True)),
    }
```

Entonces asocie un archivo INI con este descriptor:

```
self.ini_filename = __name__ + '.ini'
self.ini = IniFile(self.ini_filename,self.defaults,self.builder)
self.ini.restore_state(self)
```

After `restore_state()`, self will have attributes set if as running the following:

```
self.nhits = 0
self.a = 1.67
self.d = True
self.c = "a string"
```

Note that types are saved and preserved on restore. This example assumes that the INI file didn't exist or had the default values from self.defaults.

After this incantation, you can use the following IniFile methods:

ini.save_state(obj)
: Saves objs's attributes as per `IniFile.vars` dictionary and the widget state as described in `IniFile.widgets` in `self.defaults`.

ini.create_default_ini()
: Crea un archivo INI con valores predeterminados.

ini.restore_state(obj)
: Restore HAL out pins and obj's attributes as saved/initialized to default as above.

12.3.8.11. Saving the state on GladeVCP shutdown

To save the widget and/or variable state on exit, proceed as follows:

- Select some interior widget (type is not important, for instance a table).
- In the *Signals* tab, select *GtkObject*. It should show a *destroy* signal in the first column.

- Add the handler name, e.g. *on_destroy*, to the second column.
- Add a Python handler like below:

```
import gtk
...
def on_destroy(self,obj,data=None):
    self.ini.save_state(self)
```

This will save state and shutdown GladeVCP properly, regardless whether the panel is embedded in AXIS, or a standalone window.



atención
Do not use `window1` (the toplevel window) to connect a *destroy* event. Due to the way a GladeVCP panel interacts with AXIS, if a panel is embedded within AXIS, **window1 will not receive destroy events properly**. However, since on shutdown all widgets are destroyed, anyone will do. Recommended: use a second-level widget - for instance, if you have a table container in your panel, use that.

Next time you start the GladeVCP application, the widgets should come up in the state when the application was closed.



atención
The *GtkWidget* line has a similarly sounding *destroy-event* - **dont use that to connect to the *on_destroy* handler, it wont work** - make sure you use the *destroy* event from the *GtkObject* line.

12.3.8.12. Saving state when Ctrl-C is pressed

By default, the reaction of GladeVCP to a Ctrl-C event is to just exit - *without* saving state. To make sure that this case is covered, add a handler call `on_unix_signal` which will be automatically be called on Ctrl-C (actually on the SIGINT and SIGTERM signals). Example:

```
def on_unix_signal(self,signum,stack_frame):
    print("on_unix_signal(): signal %d received, saving state" % (signum))
    self.ini.save_state(self)
```

12.3.8.13. Hand-editing INI (.ini) files

You can do that, but note that the values in self.defaults override your edits if there is a syntax or type error in your edit. The error is detected, a console message will hint about that happened, and the bad inifile will be renamed to have the .BAD suffix. Subsequent bad INI files overwrite earlier .BAD files.

12.3.8.14. Adding HAL pins

If you need HAL pins which are not associated with a specific HAL widget, add them as follows:

```
import hal_glib
...
# in your handler class __init__():
self.example_trigger = hal_glib.GPin(halcomp.newpin('example-trigger', hal.HAL_BIT, hal. ↩
    HAL_IN))
```

To get a callback when this pin's value changes, associate a `value-change` callback with this pin, add:

```
self.example_trigger.connect('value-changed', self._on_example_trigger_change)
```

and define a callback method (or function, in this case leave out the `self` parameter):

```
# note '_' - this method will not be visible to the widget tree
def _on_example_trigger_change(self,pin,userdata=None):
    print("pin value changed to:" % (pin.get()))
```

12.3.8.15. Adding timers

Since GladeVCP uses Gtk widgets which rely on the PyGObject base class, the full GLib functionally is available. Here is an example for a timer callback:

```
def _on_timer_tick(self,userdata=None):
    ...
    return True # to restart the timer; return False for on-shot
...
# demonstrate a slow background timer - granularity is one second
# for a faster timer (granularity 100 ms), use this:
# GLib.timeout_add(100, self._on_timer_tick,userdata) # 10Hz
GLib.timeout_add_seconds(1, self._on_timer_tick)
```

12.3.8.16. Setting HAL widget properties programmatically

With Glade, widget properties are typically set fixed while editing. You can, however, set widget properties at runtime, for instance from INI file values, which would typically be done in the handler initialization code. Setting properties from HAL pin values is possible, too.

In the following example (assuming a HAL Meter widget called `meter`), the meter's min value is set from an INI file parameter at startup, and the max value is set via a HAL pin, which causes the widget's scale to readjust dynamically:

```
import linuxcnc
import os
import hal
import hal_glib

class HandlerClass:

    def _on_max_value_change(self,hal_pin,data=None):
        self.meter.max = float(hal_pin.get())
        self.meter.queue_draw() # force a widget redraw

    def __init__(self, halcomp,builder,useropts):
        self.builder = builder

        # HAL pin with change callback.
        # When the pin's value changes the callback is executed.
        self.max_value = hal_glib.GPin(halcomp.newpin('max-value',  hal.HAL_FLOAT, hal. ↩
            HAL_IN))
        self.max_value.connect('value-changed', self._on_max_value_change)

        inifile = linuxcnc.ini(os.getenv("INI_FILE_NAME"))
        mmin = inifile.getreal("METER", "MIN", fallback=0.0)
        self.meter = self.builder.get_object('meter')
        self.meter.min = mmin
```

```
def get_handlers(halcomp,builder,useropts):
    return [HandlerClass(halcomp,builder,useropts)]
```

12.3.8.17. Value-changed callback with hal_glib

GladeVCP usa la biblioteca hal_glib, que puede usarse para conectar una señal ”observador” en un pin de entrada HAL.
Esta señal se puede usar para registrar una función a llamar cuando el pin HAL cambia de estado.

One must import the `hal_glib` and the `hal` modules:

```
import hal_glib
import hal
```

Then make a pin and connect a *value-changed* (the watcher) signal to a function call:

```
class HandlerClass:
    def __init__(self, halcomp,builder,useropts):
        self.example_trigger = hal_glib.GPin(halcomp.newpin('example-trigger', hal.HAL_BIT, ↩
            hal.HAL_IN))
        self.example_trigger.connect('value-changed', self._on_example_trigger_change)
```

And have a function to be called:

```
    def _on_example_trigger_change(self,pin,userdata=None):
        print("pin value changed to: {}".format(pin.get()))
        print("pin name= {}".format(pin.get_name()))
        print("pin type= {}".format(pin.get_type()))

        # this can be called outside the function
        self.example_trigger.get()
```

12.3.8.18. Examples, and rolling your own GladeVCP application

Visit `linuxcnc_root_directory/configs/apps/gladevcp` for running examples and starters for your own projects.

12.3.9. FAQ

1. *I get an unexpected unmap event in my handler function right after startup. What's this?*

   This is a consequence of your Glade UI file having the window1 Visible property set to True, together with re-parenting the GladeVCP window into AXIS or touchy. The GladeVCP widget tree is created, including a top level window, and then *reparented into AXIS*, leaving that toplevel window laying around orphaned. To avoid having this useless empty window hanging around, it is unmapped (made invisible), which is the cause of the unmap signal you get. Suggested fix: set window1.visible to False, and ignore an initial unmap event.

2. *My GladeVCP program starts, but no window appears where I expect it to be?*

   The window AXIS allocates for GladeVCP will obtain the *natural size* of all its child widgets combined. It is the child widget's job to request a size (width and/or height). However, not all widgets do request a width greater than 0, for instance the Graph widget in its current form. If there's such a widget in your Glade file and it is the one which defines the layout you might want

to set its width explicitly. Note that setting the window1 width and height properties in Glade does not make sense because this window will be orphaned during re-parenting and hence its geometry will have no impact on layout (see above). The general rule is: if you manually run a UI file with *gladevcp <uifile>* and its window has reasonable geometry, it should come up in AXIS properly as well.

3. *I want a blinking LED, but it wont blink*

   I ticked the checkbutton to let it blink with 100 msec interval. It wont blink, and I get a startup warning: Warning: value "0" of type 'gint' is invalid or out of range for property 'led-blink-rate' of type 'gint'? This seems to be a Glade bug. Just type over the blink rate field, and save again - this works for me.

4. *My GladeVCP panel in AXIS doesn't save state when I close AXIS, although I defined an on_destroy handler linked to the window destroy signal*

   Very likely this handler is linked to window1, which due to reparenting isn't usable for this purpose. Please link the on_destroy handler to the destroy signal of an interior window. For instance, I have a notebook inside window1, and linked on_destroy to the notebooks destroy signal, and that works fine. It doesn't work for window1.

5. *I want to set the background color or text of a HAL_Label widget depending on its HAL pin value*

   See the example in configs/apps/gladevcp/colored-label. Setting the background color of a GtkLabel widget (and HAL_Label is derived from GtkLabel) is a bit tricky. The GtkLabel widget has no window object of its own for performance reasons, and only window objects can have a background color. The solution is to enclose the Label in an EventBox container, which has a window but is otherwise invisible - see the coloredlabel.ui file.

I defined a `hal_spinbutton` widget in Glade, and set a default `value` property in the correspondin

This is due to a bug in the old Gtk version distributed with Ubuntu 8.04 and 10.04, and is likely to be the case for all widgets using adjustment. The workaround mentioned for instance in http://osdir.com/ml/gtk-app-devel-list/2010-04/msg00129.html does not reliably set the HAL pin value, it is better to set it explicitly in an `on_realize` signal handler during widget creation. See the example in `configs/apps/gladevcp/by-widget/spinbutton.{ui,py}`.

12.3.10. Troubleshooting

- Make sure you have the development version of LinuxCNC installed. You don't need the axisrc file any more, this was mentioned in the old GladeVCP wiki page.
- Run GladeVCP or AXIS from a terminal window. If you get Python errors, check whether there's still a `/usr/lib/python2.6/dist-packages/hal.so` file lying around besides the newer `/usr/lib/python2.6` (note the underscore); if yes, remove the `hal.so` file. It has been superseded by hal.py in the same directory and confuses the import mechanism.
- If you're using run-in-place, do a *make clean* to remove any accidentally left over hal.so file, then *make*.
- If you're using *HAL_table* or *HAL_HBox* widgets, be aware they have an HAL pin associated with it which is off by default. This pin controls whether these container's children are active or not.

12.3.11. Implementation note: Key handling in AXIS

We believe key handling works OK, but since it is new code, we're telling about it you so you can watch out for problems; please let us know of errors or odd behavior. This is the story:

AXIS uses the TkInter widget set. GladeVCP applications use Gtk widgets and run in a separate process context. They are hooked into AXIS with the Xembed protocol. This allows a child application like GladeVCP to properly fit in a parent's window, and - in theory - have integrated event handling.

However, this assumes that both parent and child application properly support the Xembed protocol, which Gtk does, but TkInter does not. A consequence of this is that certain keys would not be forwarded from a GladeVCP panel to AXIS properly under all circumstances. One of these situations was the case when an Entry, or SpinButton widget had focus: In this case, for instance an Escape key would not have been forwarded to AXIS and cause an abort as it should, with potentially disastrous consequences.

Therefore, key events in GladeVCP are explicitly handled, and selectively forwarded to AXIS, to assure that such situations cannot arise. For details, see the `keyboard_forward()` function in `lib/python/gladevcp`

12.3.12. Adding Custom Widgets

The LinuxCNC Wiki has information on adding custom widgets to GladeVCP. GladeVCP Custom Widgets

12.3.13. Aplicaciones auxiliares GladeVCP

Support is provided for independently installed GladeVCP applications that conform to system directory placements as defined by the LINUXCNC_AUX_GLADEVCP and LINUXCNC_AUX_EXAMPLES items reported by the script linuxcnc_var:

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

The system directory defined by LINUXCNC_AUX_GLADEVCP (/usr/share/linuxcnc/aux_gladevcp) specifies the location for a GladeVCP-compatible Python file(s) and related subdirectories. The Python file is imported at GladeVCP startup and made available to subsequent GladeVCP applications including embedded usage in supporting GUIs.

The system directory defined by LINUXCNC_AUX_EXAMPLES (/usr/share/linuxcnc/aux_examples) specifies the location of example configuration subdirectories used for auxiliary applications. See the getting-started/running-linuxcnc section for *Adding Configuration Selection Items*.

For testing, a runtime specification of auxiliary applications may be specified using the exported environmental variable: GLADEVCP_EXTRAS. This variable should be a path list of one or more configuration directories separated by a (:). Typically, this variable would be set in a shell starting `linuxcnc` or in a user's `~/.profile` startup script. Example:

```
export GLADEVCP_EXTRAS=~/mygladevcp:/opt/othergladevcp
```

Files found in directories specified with the environmental variable GLADEVCP_EXTRAS supersede identically-named files within subdirectories of the system directory specified by LINUXNC_AUX_GLADEVCP (e.g., /usr/share/linuxcnc/aux_gladevcp). This provision allows a developer to test an application by exporting GLADEVCP_EXTRAS to specify a private application directory without removing a system-installed application directory. Messages indicating rejected duplicates are printed to stdout.

---

nota
Support for auxiliary GladeVCP applications requires a Python module named *importlib*. This module may not be available in older installations like Ubuntu-Lucid.

12.4. GladeVCP Library modules

Las bibliotecas son módulos de Python preconstruidos que le dan características adicionales a GladeVCP. De esta manera, puede seleccionar qué características desea, y no tenga que construir las comunes usted mismo.

12.4.1. Info

Info is a library to collect and filters data from the INI file.

The available data and defaults:

```
LINUXCNC_IS_RUNNING
LINUXCNC_VERSION
INIPATH
INI = linuxcnc.ini(INIPATH)
MDI_HISTORY_PATH = '~/.axis_mdi_history'
QTVCP_LOG_HISTORY_PATH = '~/qtvcp.log'
MACHINE_LOG_HISTORY_PATH = '~/.machine_log_history'
PREFERENCE_PATH = '~/.Preferences'
SUB_PATH = None
SUB_PATH_LIST = []
self.MACRO_PATH = None
MACRO_PATH_LIST = []
INI_MACROS = self.INI.findall("DISPLAY", "MACRO")

IMAGE_PATH = IMAGEDIR
LIB_PATH = os.path.join(HOME, "share","qtvcp")

PROGRAM_FILTERS = None
PARAMETER_FILE = None
MACHINE_IS_LATHE = False
MACHINE_IS_METRIC = False
MACHINE_UNIT_CONVERSION = 1
MACHINE_UNIT_CONVERSION_9 = [1]*9
TRAJ_COORDINATES =
JOINT_COUNT = self.INI.getint("KINS","JOINTS", fallback=0)
AVAILABLE_AXES = ['X','Y','Z']
AVAILABLE_JOINTS = [0,1,2]
GET_NAME_FROM_JOINT = {0:'X',1:'Y',2:'Z'}
GET_JOG_FROM_NAME = {'X':0,'Y':1,'Z':2}
NO_HOME_REQUIRED = False
HOME_ALL_FLAG
JOINT_TYPE = self.INI.getstring(section, "TYPE", fallback="LINEAR")
JOINT_SEQUENCE_LIST
JOINT_SYNC_LIST

JOG_INCREMENTS = None
ANGULAR_INCREMENTS = None
GRID_INCREMENTS

DEFAULT_LINEAR_JOG_VEL = 15 units per minute
MIN_LINEAR_JOG_VEL = 60 units per minute
MAX_LINEAR_JOG_VEL = 300 units per minute

DEFAULT_ANGULAR_JOG_VEL =
MIN_ANGULAR_JOG_VEL =
MAX_ANGULAR_JOG_VEL =

MAX_FEED_OVERRIDE =
```

```
MAX_TRAJ_VELOCITY =

AVAILABLE_SPINDLES = self.INI.getint("TRAJ", "SPINDLES", fallback=1)
DEFAULT_SPINDLE_0_SPEED = 200
MAX_SPINDLE_0_SPEED = 2500
MAX_SPINDLE_0_OVERRIDE = 100
MIN_SPINDLE_0_OVERRIDE = 50

MAX_FEED_OVERRIDE = 1.5
MAX_TRAJ_VELOCITY

# user message dialog info
USRMESS_BOLDTEXT = self.INI.findall("DISPLAY", "MESSAGE_BOLDTEXT")
USRMESS_TEXT = self.INI.findall("DISPLAY", "MESSAGE_TEXT")
USRMESS_TYPE = self.INI.findall("DISPLAY", "MESSAGE_TYPE")
USRMESS_PINNAME = self.INI.findall("DISPLAY", "MESSAGE_PINNAME")
USRMESS_DETAILS = self.INI.findall("DISPLAY", "MESSAGE_DETAILS")
USRMESS_ICON = self.INI.findall("DISPLAY", "MESSAGE_ICON")
ZIPPED_USRMESS =

self.GLADEVCP = self.INI.find("DISPLAY", "GLADEVCP")

# embedded program info
TAB_NAMES = (self.INI.findall("DISPLAY", "EMBED_TAB_NAME")) or None
TAB_LOCATION = (self.INI.findall("DISPLAY", "EMBED_TAB_LOCATION")) or []
TAB_CMD = (self.INI.findall("DISPLAY", "EMBED_TAB_COMMAND")) or None
ZIPPED_TABS =

MDI_COMMAND_LIST =      (heading: [MDI_COMMAND_LIST], title: MDI_COMMAND")
TOOL_FILE_PATH =        (heading: [EMCIO], title:TOOL_TABLE)
POSTGUI_HALFILE_PATH =  (heading: [HAL], title: POSTGUI_HALFILE)
```

Hay algunas *funciones auxiliares* - mayormente usadas para soporte de widgets

```
get_error_safe_setting(self, heading, detail, default=None)
convert_metric_to_machine(data)
convert_imperial_to_machine(data)
convert_9_metric_to_machine(data)
convert_9_imperial_to_machine(data)
convert_units(data)
convert_units_9(data)
get_filter_program(fname)
```

To import this modules add this Python code to your import section:

```
############################
# **** IMPORT SECTION **** #
############################

from gladevcp.core import Info
```

To instantiate the module so you can use it in a handler file add this Python code to your instantiate section:

```
###########################################
# **** INSTANTIATE LIBRARIES SECTION **** #
###########################################

INFO = Info()
```

To access INFO data use this general syntax:

```
home_state = INFO.NO_HOME_REQUIRED
if INFO.MACHINE_IS_METRIC is True:
    print('Metric based')
```

12.4.2. Action

This library is used to command LinuxCNC's motion controller. It tries to hide incidental details and add convenience methods for developers.

To import this modules add this Python code to your import section:

```
############################
# **** IMPORT SECTION **** #
############################

from gladevcp.core import Action
```

To instantiate the module so you can use it add this Python code to your instantiate section:

```
###########################################
# **** INSTANTIATE LIBRARIES SECTION **** #
###########################################

ACTION = Action()
```

To access Action commands use general syntax such as these:

```
ACTION.SET_ESTOP_STATE(state)
ACTION.SET_MACHINE_STATE(state)

ACTION.SET_MACHINE_HOMING(joint)
ACTION.SET_MACHINE_UNHOMED(joint)

ACTION.SET_LIMITS_OVERRIDE()

ACTION.SET_MDI_MODE()
ACTION.SET_MANUAL_MODE()
ACTION.SET_AUTO_MODE()

ACTION.SET_LIMITS_OVERRIDE()

ACTION.CALL_MDI(code)
ACTION.CALL_MDI_WAIT(code)
ACTION.CALL_INI_MDI(number)

ACTION.CALL_OWORD()

ACTION.OPEN_PROGRAM(filename)
ACTION.SAVE_PROGRAM(text_source, fname):

ACTION.SET_AXIS_ORIGIN(axis,value)
ACTION.SET_TOOL_OFFSET(axis,value,fixture = False)

ACTION.RUN()
ACTION.ABORT()
ACTION.PAUSE()

ACTION.SET_MAX_VELOCITY_RATE(rate)
ACTION.SET_RAPID_RATE(rate)
```

```
ACTION.SET_FEED_RATE(rate)
ACTION.SET_SPINDLE_RATE(rate)

ACTION.SET_JOG_RATE(rate)
ACTION.SET_JOG_INCR(incr)
ACTION.SET_JOG_RATE_ANGULAR(rate)
ACTION.SET_JOG_INCR_ANGULAR(incr, text)

ACTION.SET_SPINDLE_ROTATION(direction = 1, rpm = 100, number = 0)
ACTION.SET_SPINDLE_FASTER(number = 0)
ACTION.SET_SPINDLE_SLOWER(number = 0)
ACTION.SET_SPINDLE_STOP(number = 0)

ACTION.SET_USER_SYSTEM(system)

ACTION.ZERO_G92_OFFSET()
ACTION.ZERO_ROTATIONAL_OFFSET()
ACTION.ZERO_G5X_OFFSET(num)

ACTION.RECORD_CURRENT_MODE()
ACTION.RESTORE_RECORDED_MODE()

ACTION.SET_SELECTED_AXIS(jointnum)

ACTION.DO_JOG(jointnum, direction)
ACTION.JOG(jointnum, direction, rate, distance=0)

ACTION.TOGGLE_FLOOD()
ACTION.SET_FLOOD_ON()
ACTION.SET_FLOOD_OFF()

ACTION.TOGGLE_MIST()
ACTION.SET_MIST_ON()
ACTION.SET_MIST_OFF()

ACTION.RELOAD_TOOLTABLE()
ACTION.UPDATE_VAR_FILE()

ACTION.TOGGLE_OPTIONAL_STOP()
ACTION.SET_OPTIONAL_STOP_ON()
ACTION.SET_OPTIONAL_STOP_OFF()

ACTION.TOGGLE_BLOCK_DELETE()
ACTION.SET_BLOCK_DELETE_ON()
ACTION.SET_BLOCK_DELETE_OFF()

ACTION.RELOAD_DISPLAY()
ACTION.SET_GRAPHICS_VIEW(view)

ACTION.UPDATE_MACHINE_LOG(text, option=None):

ACTION.SET_DISPLAY_MESSAGE(string)
ACTION.SET_ERROR_MESSAGE(string)
```

There are some *helper functions* - mostly used for this library's support

```
get_jog_info (num)
jnum_check(num)
ensure_mode(modes)
open_filter_program(filename, filter)
```

12.5. QtVCP

QtVCP is an **infrastructure to build custom CNC screens or control panels for LinuxCNC**.

It displays a *`.ui` file built with Qt Designer* screen editor and combines this with *Python programming* to create a GUI screen for running a CNC machine.

QtVCP is completely *customizable*: you can add different buttons and status LEDs etc., or add Python code for even finer grain customization.

12.5.1. Showcase

Algunos ejemplos de pantallas QtVCP construidas y paneles de control virtuales:

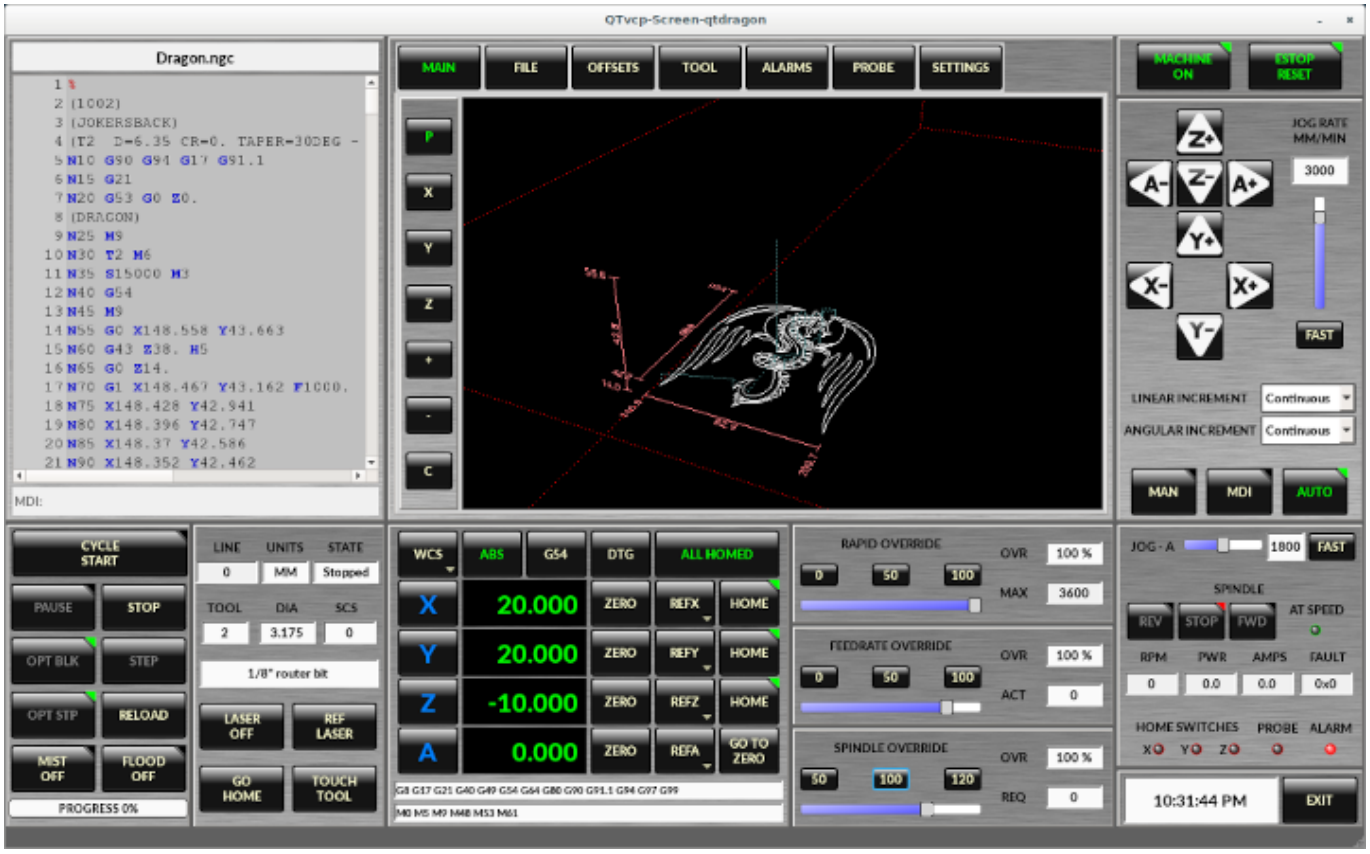


Figura 12.55: QtDragon - 3/4-Axis Sample



Figura 12.56: QtDefault - 3-Axis Sample

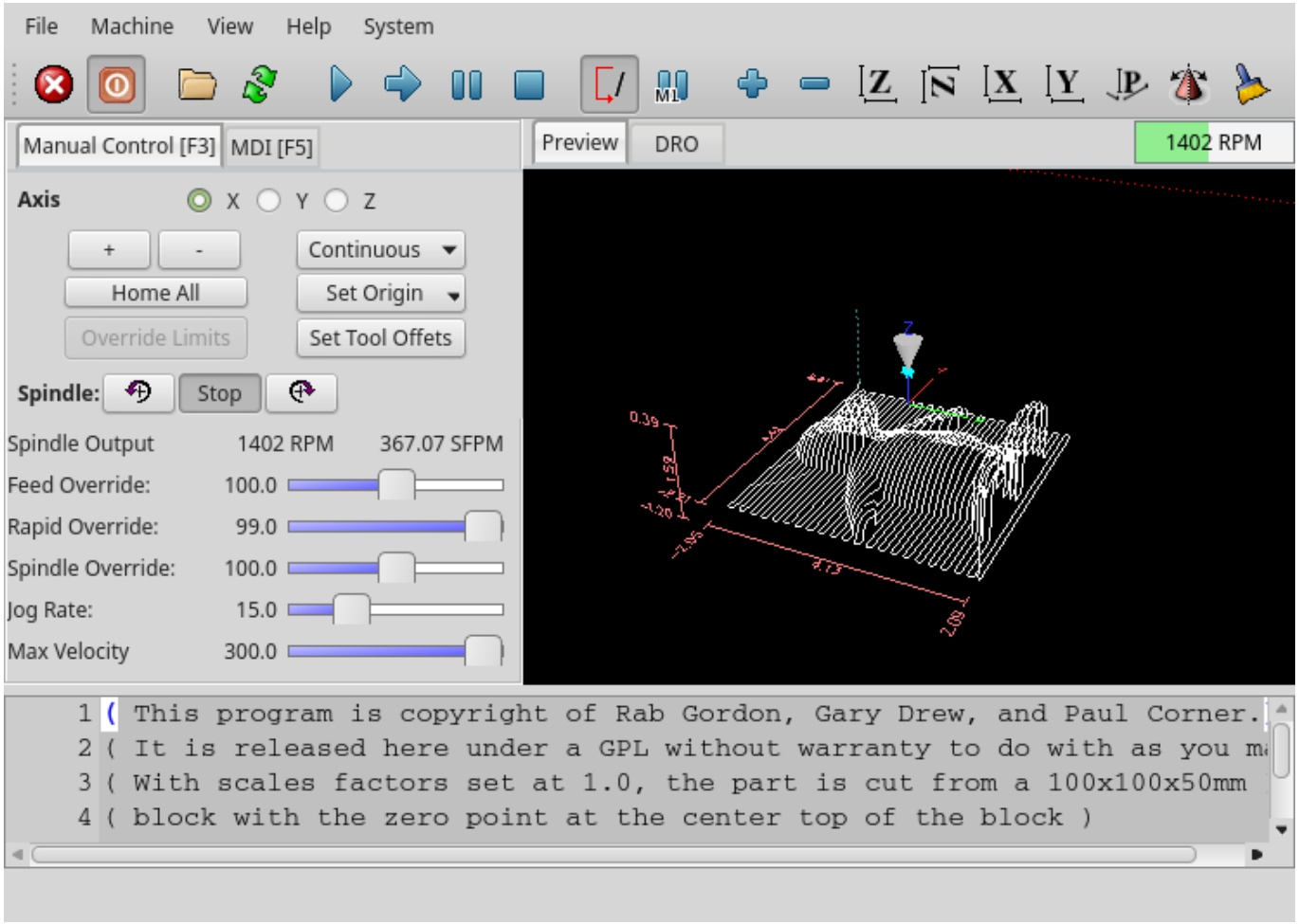


Figura 12.57: QtAxis - Self Adjusting Axis Sample

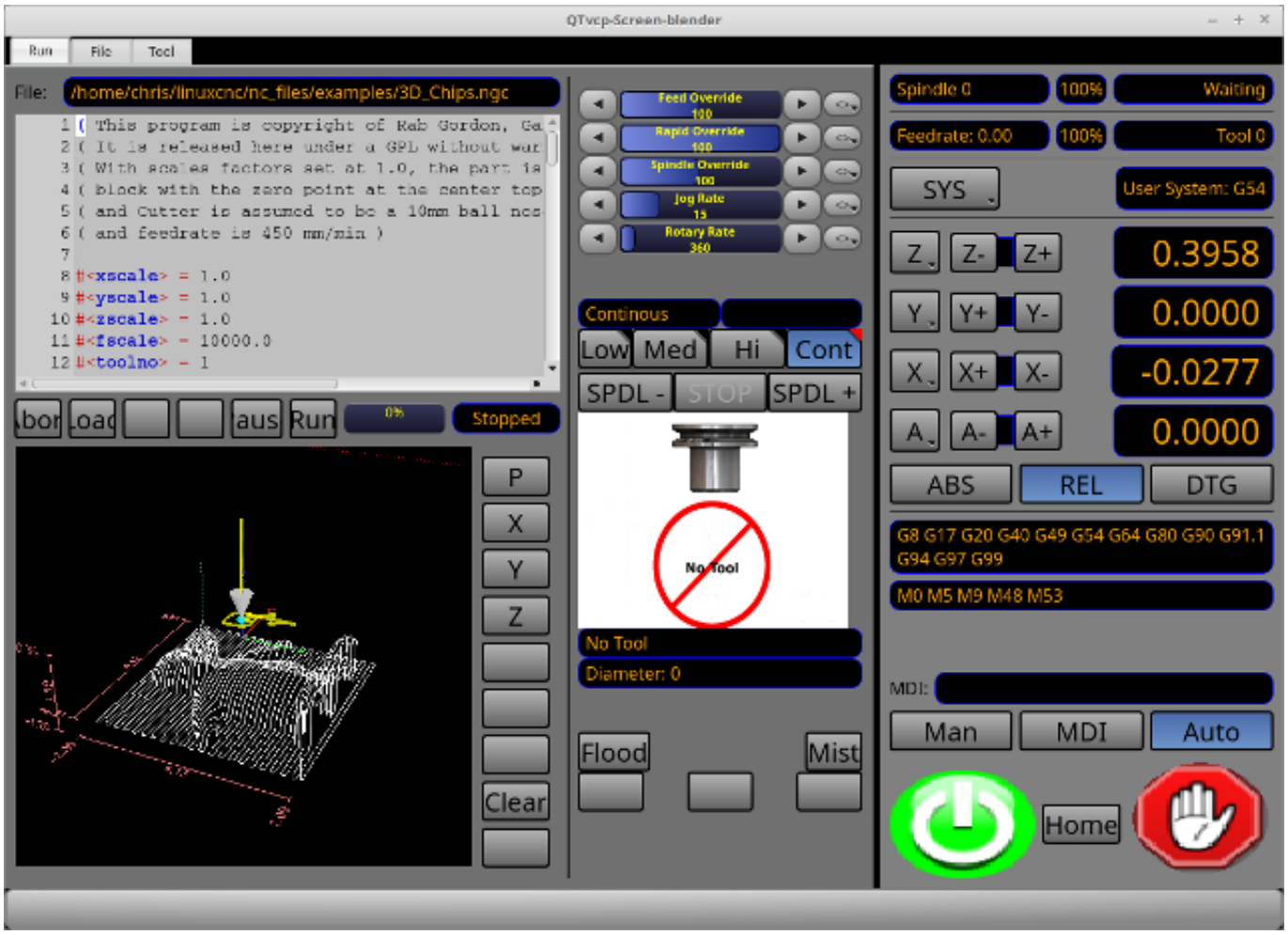


Figura 12.58: Blender - 4-Axis Sample

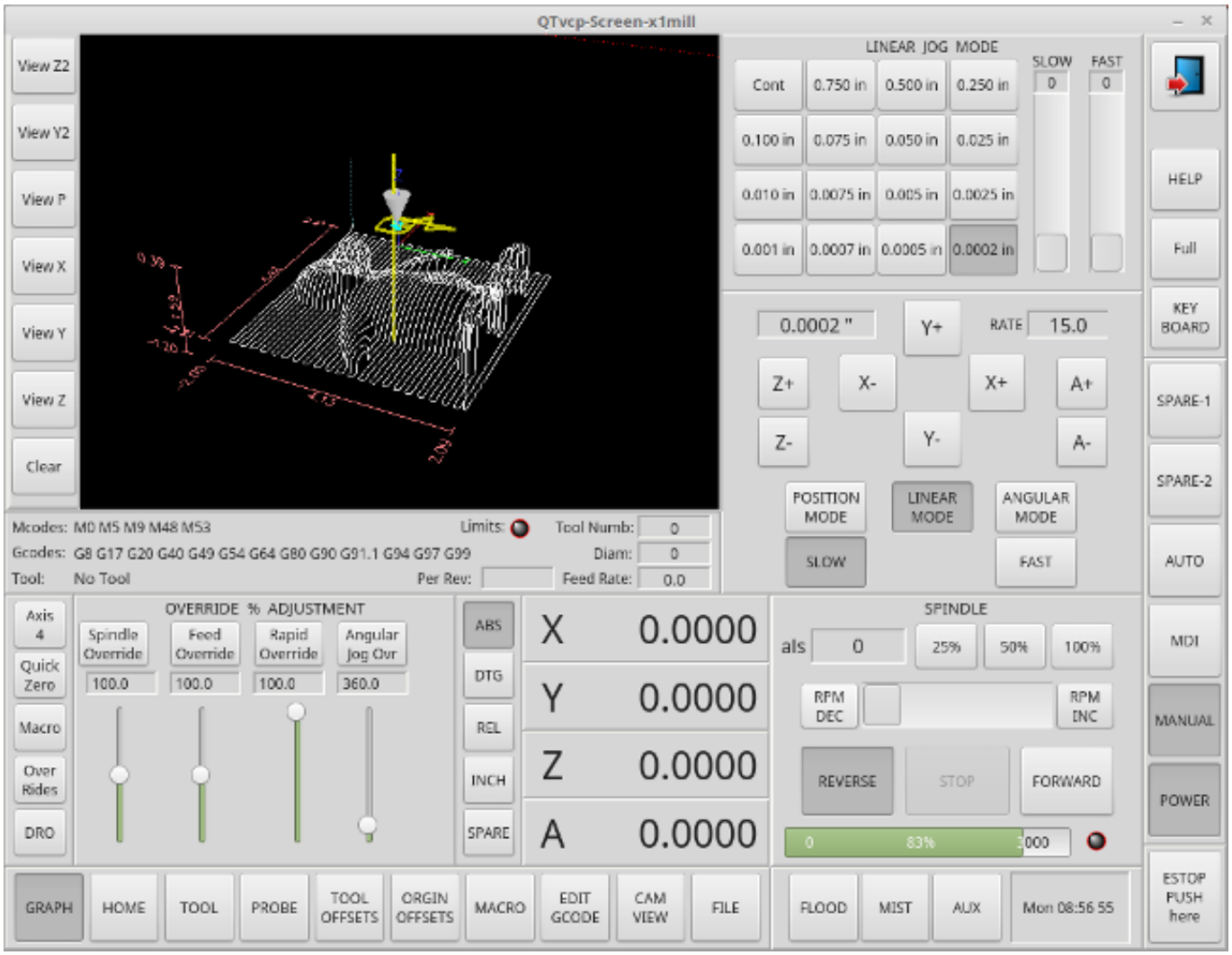


Figura 12.59: X1mill - 4-Axis Sample



Figura 12.60: cam_align - Camera Alignment VCP

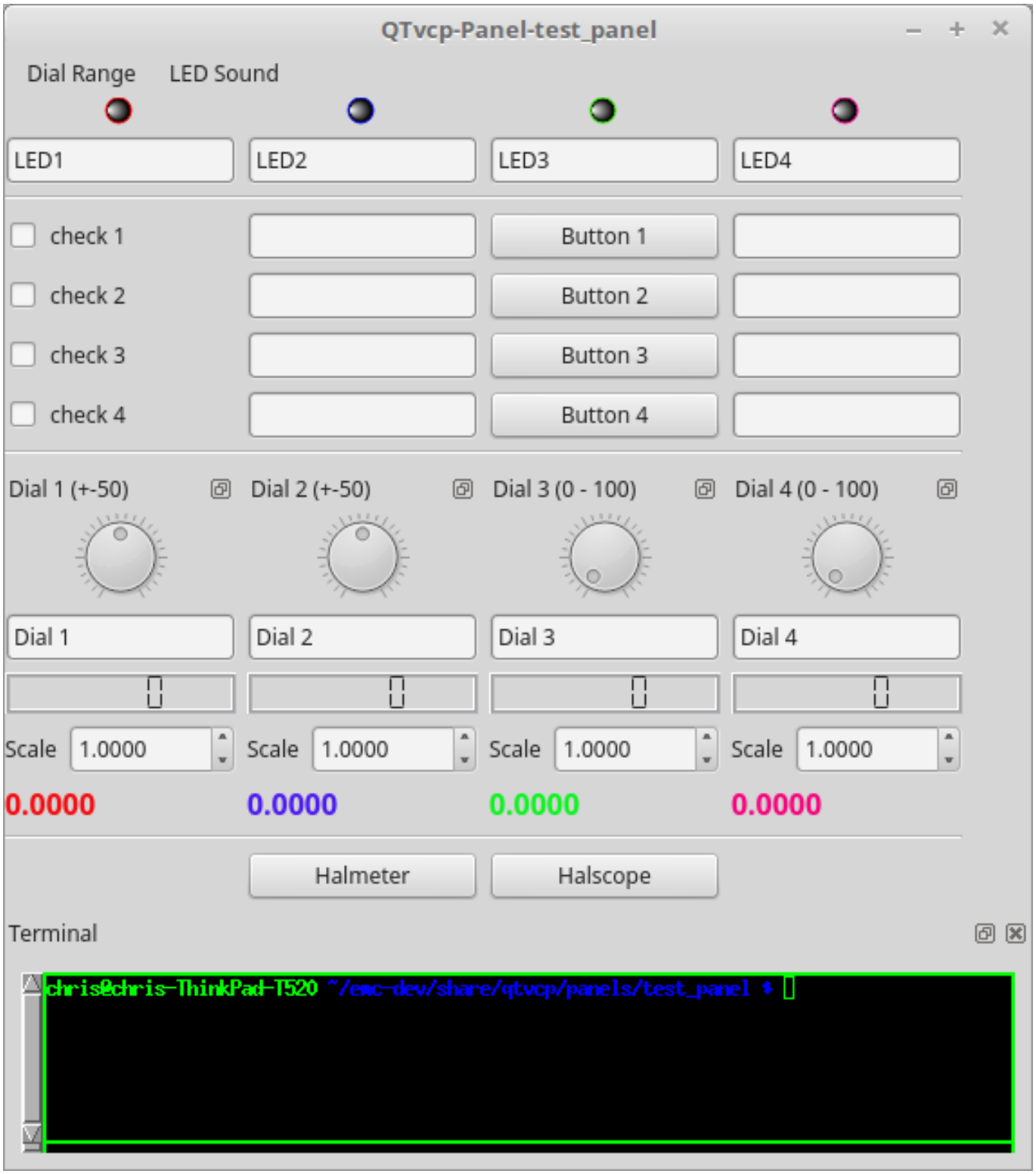


Figura 12.61: test_panel - Test Panel VCP

12.5.2. Descripción general

Two files are used, individually or in combination, to add customizations:

- A **UI file** that is a *XML* file made with *Qt Designer* graphical editor.
- A **handler file** which is a *Python* code text file.

Normally QtVCP uses the stock UI and handler file, but you can specify QtVCP to use *local* UI and handler files.
A ***local* file** is one that is in the *configuration folder* that defines the rest of the machine's requirements.

One is not restricted to adding a custom panel on the right or a custom tab as QtVCP leverages *Qt Designer* (the editor) and *PyQt5* (the widget toolkit).

QtVCP has some added **special LinuxCNC widgets and actions**.
There are special widgets to bridge third party widgets to HAL pins.
It's possible to create widget responses by connecting signals to Python code in the handler file.

12.5.2.1. QtVCP Widgets

QtVCP uses the **PyQt5 toolkit's widgets** for LinuxCNC integration.

Widget is the *general name for user interface objects* such as buttons and labels in PyQt5.

You are free to use any available **default widgets** in the Qt Designer editor.

There are also **special widgets** made for LinuxCNC that make integration easier.
These are split in three heading on the left side of the editor:

- Uno es para *HAL only widgets*;
- One is for *CNC control widgets*;
- Uno es para *Widgets del diálogo*.

You are free to mix them in any way on your panel.

A very important widget for CNC control is the **`ScreenOptions` widget**: It does not add anything visually to the screen but, allows important details to be selected rather then be coded in the handler file.

12.5.2.2. Configuración INI

If you are using QtVCP to make a CNC motion control screen (rather then a HAL based panel), in the INI file, in the `[DISPLAY]` section, add a line with the following pattern:

```
DISPLAY = qtvcp <opciones> <nombre-de-pantalla>
```

---

nota
All `<options>` must appear before `<screen_name>`.

Opciones

- `-d` Activar depuración.
- `-i` Enable info output.
- `-v` Enable verbose debug output.
- `-q` Enable only error debug output.

- `-a` Set window always on top.
- `-c NAME` HAL component name. Default is to use the UI file name.
- `-g GEOMETRY` Set geometry WIDTHxHEIGHT+XOFFSET+YOFFSET. Values are in pixel units, XOFFSET/YOFFSET is referenced from top left of screen. Use -g WIDTHxHEIGHT for just setting size or -g +XOFFSET+YOFFSET for just position. Example: `-g 200x400+0+100`
- `-H FILE` Execute hal statements from FILE with halcmd after the component is set up and ready.
- `-m` Maximize window.
- `-f` Fullscreen the window.
- `-t THEME` Default is system theme
- `-x XID` Embed into a X11 window that doesn't support embedding.
- `--push_xid` Send QtVCP's X11 window id number to standard output; for embedding.
- `-u USERMOD` File path of a substitute handler file.
- `-o USEROPTS` Pass a string to QtVCP's handler file under `self.w.USEROPTIONS_` list variable. Can be multiple -o.

<screen_name> `<nombre-de-pantalla>` es el nombre base de los archivos .ui y _handler.py. Si falta `<nombre-de-pantalla>`, se cargará la pantalla predeterminada.

QtVCP assumes the UI file and the handler file use the **same base name**. QtVCP will first search the LinuxCNC configuration directory that was launched for the files, then in the system skin folder holding standard screens.

Cycle Times

```
[DISPLAY]
CYCLE_TIME = 100
GRAPHICS_CYCLE_TIME = 100
HALPIN_CYCLE = 100
```

Adjusts the response rate of the GUI updates in milliseconds. Defaults to 100, useable range 50 - 200.

The widgets, graphics and HAL pin update can be set separately.

If the update time is not set right the screen can become unresponsive or very jerky.

12.5.2.3. Qt Designer UI File

A Qt Designer file is a text file organized in the *XML* standard that describes the **layout and widgets** of the screen.

PyQt5 uses this file to build the display and react to those widgets.

The Qt Designer editor makes it relatively easy to build and edit this file.

12.5.2.4. Handler Files

A handler file is a file containing *Python* code, which **adds to QtVCP default routines**.

A handler file allows one to *modify defaults*, or *add logic* to a QtVCP screen without having to modify QtVCP's core code. In this way you can have **custom behaviors**.

If present a handler file will be loaded. **Only one file** is allowed.

12.5.2.5. Libraries Modules

QtVCP como está construido hace poco más que mostrar la pantalla y reaccionar a los widgets. Para más **comportamientos preconstruidos** hay bibliotecas disponibles. (que se encuentran en `lib/python/qtvcp/lib` en RIP LinuxCNC install).

Las **bibliotecas** son *Módulos de Python* preconstruidos que **agregan características** a QtVCP. De esta manera, puede seleccionar qué características desea, sin embargo, no tiene que crear las comunes usted mismo.

Such libraries include:

- `audio_player`
- `aux_program_loader`
- `keybindings`
- `message`
- `preferences`
- `notify`
- `virtual_keyboard`
- `machine_log`

12.5.2.6. Themes

Themes are a way to modify the **look and feel** of the widgets on the screen.

For instance the *color* or *size* of buttons and sliders can be changed using themes.

The *Windows theme* is default for screens.
The *System theme* is default for panels.

To see available themes, they can be loaded by running the following command in a terminal:

```
qtvcp -d -t <theme_name>
```

QtVCP can also be customized with *Qt stylesheets (QSS)* using CSS.

12.5.2.7. Local Files

If present, local UI/QSS/Python files in the configuration folder will be loaded instead of the stock UI files.

Local UI/QSS/Python files allow you to use your customized designs rather than the default screens.

QtVCP will look for a folder named <screen_name> (in the launched configuration folder that holds the INI file).

In that folder, QtVCP will load any of the available following files:

- `<screen_name>.ui`,
- `<screen_name>_handler.py`, and
- `<screen_name>.qss`.

12.5.2.8. Modifying Stock Screens

There are *three ways* to customize a screen/panel.

Las hojas de estilo se pueden usar para **asignar propiedades de Qt**. Si un widget usa propiedades, usualmente pueden ser modificadas mediante hojas de estilo.

Example of a widget with accompanying style sheet settings.

```
State_LED #name_of_led{
  qproperty-color: red;
  qproperty-diameter: 20;
  qproperty-flashRate: 150;
  }
```

We can have QtVCP load a subclassed version of the standard handler file. In that file we can manipulate the original functions or add new ones.
Subclassing just means our handler file first loads the original handler file and adds our new code on top of it - effectively a patch of changes.
This is useful for changing/adding behaviour while still retaining standard handler updates from LinuxCNC repositories.

You may still need to use the handler copy dialog to copy the original handler file to decide how to patch it. See *custom handler file*.

There should be a folder in the config folder; for screens: named *<CONFIG FOLDER>/qtvcp/screens/<SCRE NAME>/*
add the handle patch file there, named like so <ORIGINAL SCREEN NAME>_handler.py,
e.g., for QtDragon the file would be called *qtdragon_handler.py*.

Here is a sample to add X axis jog pins to a screen like QtDragon:

```
import sys
import importlib
from qtvcp.core import Path, Qhal, Action
PATH = Path()
QHAL = Qhal()
ACTION = Action()

# get reference to original handler file so we can subclass it
sys.path.insert(0, PATH.SCREENDIR)
module = "{}.{}_handler".format(PATH.BASEPATH,PATH.BASEPATH)
mod = importlib.import_module(module, PATH.SCREENDIR)
sys.path.remove(PATH.SCREENDIR)
HandlerClass = mod.HandlerClass

# regresa nuestro objeto subclase manejador a QtVCP
def get_handlers(halcomp, widgets, paths):
    return [UserHandlerClass(halcomp, widgets, paths)]

# sub class HandlerClass which was imported above
class UserHandlerClass(HandlerClass):
    # add a terminal message so we know this got loaded
    print('\nCustom subclassed handler patch loaded.\n')

    def init_pins(self):
        # call original handler init_pins function
        super().init_pins()

        # add jog pins X axis
        pin = QHAL.newpin("jog.axis.jog-x-plus", QHAL.HAL_BIT, QHAL.HAL_IN)
        pin.value_changed.connect(lambda s: self.kb_jog(s, 0, 1, fast = False, linear =  ↩
            True))
```

```
        pin = QHAL.newpin("jog.axis.jog-x-minus", QHAL.HAL_BIT, QHAL.HAL_IN)
        pin.value_changed.connect(lambda s: self.kb_jog(s, 0, -1, fast = False, linear =  ↩
            True))
```

Another Python file can be used to **add commands** to the screen, after the handler file is parsed. This can be useful for minor changes while still honouring standard handler updates from LinuxCNC repositories.

---

nota
Handler patching is a better way to add changes - instance patching is black magic voodoo - this is here for legacy reasons.

En el *archivo INI* bajo el encabezado `[DISPLAY]` agregue **`USER_COMMAND_FILE = _PATH_`**
PATH puede ser cualquier ruta válida. Puede usar `~` para el directorio casa o `WORKINGFOLDER` o `CONFIGFOLDER` para representar la idea de QtVCP de esos directorios, p. ej.:

```
[DISPLAY]
USER_COMMAND_FILE = CONFIGFOLDER/<screen_name_added_commands>
```

Si no se encuentra una entrada en el *INI*, QtVCP buscará en la **ruta predeterminada**. La ruta predeterminada esta en el directorio de configuración como un archivo oculto usando el nombre base de la pantalla y rc, p. ej. **`DIRECTORIO DE CONFIGURACIÓN/.<nombre_de_pantalla>rc`**.

This file will be read and executed as Python code in the **handler file context**.

Solo funciones y atributos locales pueden referenciarse
Las librerías globales definidas en el archivo manejador de la pantalla pueden referenciarse importando el archivo manejador.
Esas se ven normalmente en mayúsculas sin tener antes self.
self referencia las funciones de clase de la ventana.
self.w referencia típicamente a los widgets.

What can be used can vary by screen and development cycle.

A simple example Reference the main window to change the title (won't show if using INI entries for title change).

```
self.w.setWindowTitle('My Title Test')
```

An advanced instance patching example This could work with the QtDragon screen's handler file. Here we show how to add new functions and override existing ones.

```
# Needed to instance patch.
# reference: https://ruivieira.dev/python-monkey-patching-for-readability.html
import types

# import the handlerfile to get reference to its libraries.
# use <screenname>_handler
import qtdragon_handler as hdlr

# This is actually an unbounded function with 'obj' as a parameter.
# You call this function without the usual preceding 'self.'.
# This is because will will not be patching it into the original handler class instance
# It will only be called from code in this file.
def test_function(obj):
    print(dir(obj))

# This is a new function we will added to the existing handler class instance.
```

```
# Notice it calls the unbounded function with 'self' as an parameter 'self' is the only global reference available.
# It references the window instance.
def on_keycall_F10(self,event,state,shift,cntrl):
    if state:
        print ('F10')
        test_function(self)

# This will be used to override an existing function in the existing handler class instance.
# Note, we also call a copy of the original function too.
# This shows how to extend an existing function to do extra functions.
def on_keycall_F11(self,event,state,shift,cntrl):
    if state:
        self.on_keycall_F11_super(event,state,shift,cntrl)
        print ('Hello')

# We are referencing the KEYBIND library that was instantiated in the original handler class instance
# by adding 'hdlr.' to it (from the imp).
# This function tells KEYBIND to call 'on_keycall_F10' when F10 is pressed.
hdlr.KEYBIND.add_call('Key_F10','on_keycall_F10')

# Here we are instance patching the original handler file to add a new function
# that calls our new function (of the same name) defined in this file.
self.on_keycall_F10 = types.MethodType(on_keycall_F10, self)

# Here we are defining a copy of the original 'on_keycall_F11' function,
# so we can call it later. We can use any valid, unused function name.
# We need to do this before overriding the original function.
self.on_keycall_F11_super = self.on_keycall_F11

# Here we are instance patching the original handler file to override an existing function
# to point to our new function (of the same name) defined in this file.
self.on_keycall_F11 = types.MethodType(on_keycall_F11, self)


# add a new pin to the screen:

# pin callback to print the state
def new_pin_changed(data):
    print(data)

# Special function that gets called before the HAL component is set ready.
# Here we used the function to add a bit input pin with a callback.
def after_override__(self):
    try:
        pin = hdlr.QHAL.newpin("new_pin", hdlr.QHAL.HAL_BIT, hdlr.QHAL.HAL_IN)
        pin.value_changed.connect(new_pin_changed)
    except Exception as e:
        print(e)

# Here we are instance patching the original handler file to add a new function
# that calls our new function (of the same name) defined in this file.
self.after_override__ = types.MethodType(after_override__, self)
```

If you wish to **modify a stock screen** with full control, *copy its UI and handler file to your configuration folder*.

There is a QtVCP panel to help with this:

- Abra una ventana de terminal y ejecute el comando siguiente:

```
qtvcp copy
```

- Select the screen and destination folder in the dialog
- If you wish to **name your screen** differently than the builtin screen's default name, change the *basename* in the edit box.
- There should be a folder in the config folder; for screens: named *<CONFIG FOLDER>/qtvcp/screens/* for panels: named *<CONFIG FOLDER>/qtvcp/panels/* add the folders if they are missing and copy your folder/files in it.
- Validate to copy all the files
- Delete the files you don't wish to modify so that the original files will be used.

12.5.3. VCP Panels

QtVCP can be used to create control panels that interface with **HAL**.

12.5.3.1. Builtin Panels

Hay varios **paneles HAL interconstruídos** disponibles.

En una terminal teclee `qtvcp <return>` para ver una lista:

`test_panel`
: Collection of useful widgets for testing HAL components, including speech of LED state.

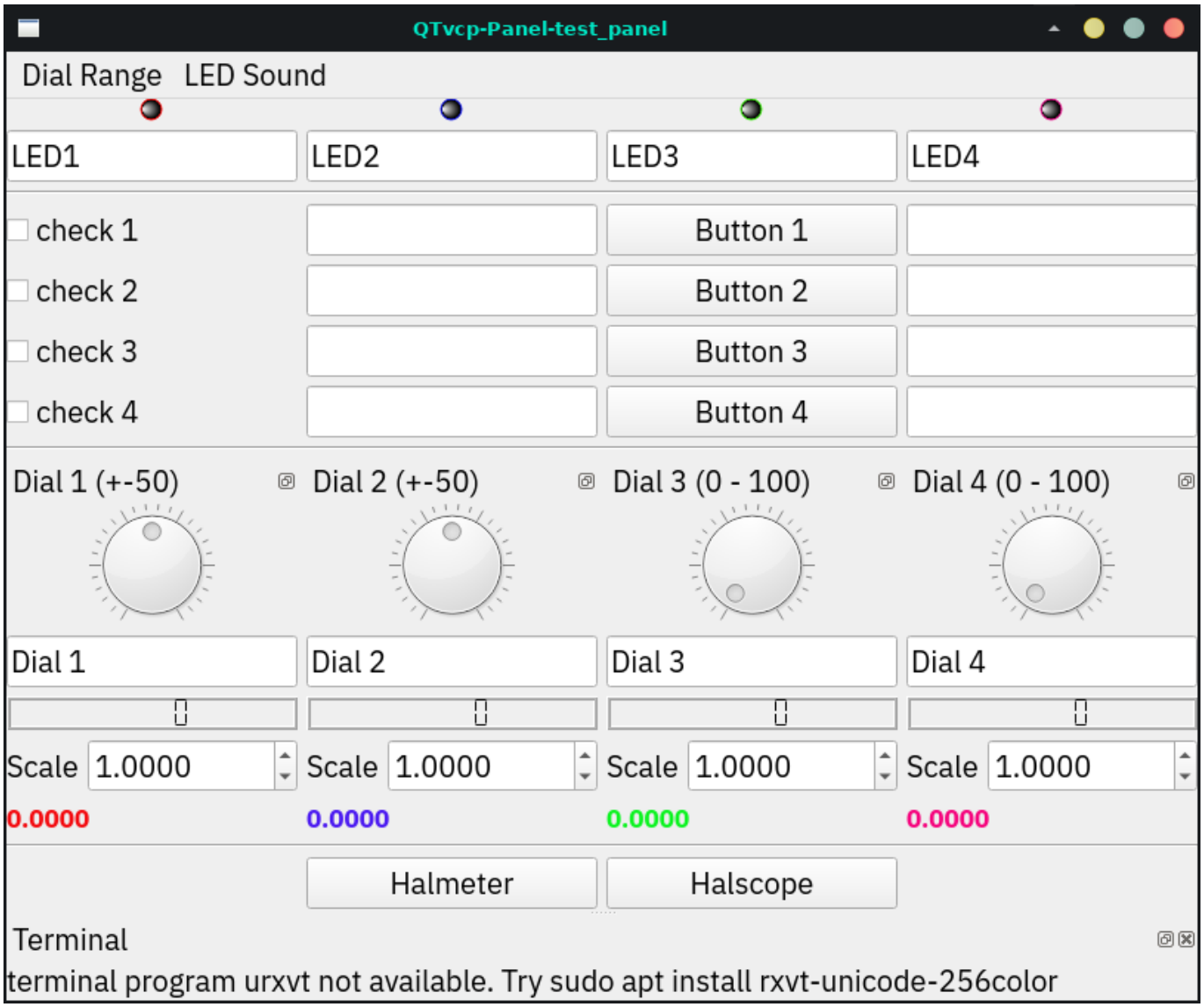


Figura 12.62: QtVCP HAL Test Builtin Panel

cam_align
: Un widget de visualización de cámara para alineación rotacional.



Figura 12.63: cam_align - Camera Alignment VCP

sim_panel
: Un pequeño panel de control para simular controles de trote MPG, etc.
Para configuraciones simuladas.



Figura 12.64: QtVCP Sim Builtin Panel

vismach_mill_xyz
vista 3D de OpenGL de una máquina fresadora de 3 ejes.

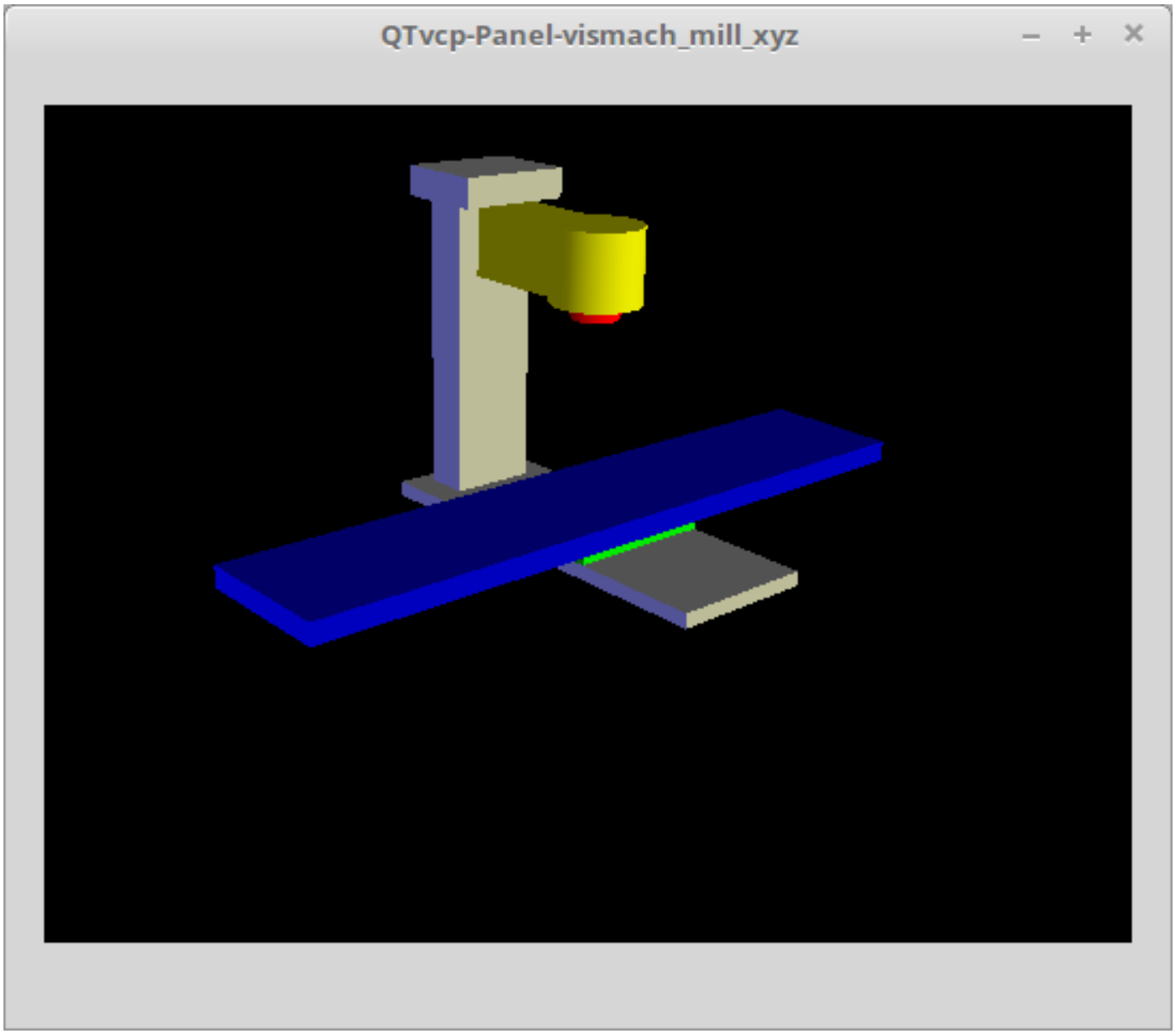


Figura 12.65: QtVismach - 3-Axis Mill Builtin Panel

Puede cargarlos desde la terminal o desde un archivo HAL con este comando básico:

```
loadusr qtvcp test_panel
```

But more typically like this:

```
loadusr -Wn test_panel qtvcp test_panel
```

In this way HAL will wait till the HAL pins are made before continuing on.

12.5.3.2. Custom Panels

You can of course **make your own panel and load it**.

Si has creado un archivo UI nombrado `my_panel.ui` y un archivo HAL nombrado `my_panel.hal`, podrías cargarlos desde una terminal con:

```
halrun -I -f my_panel.hal
```

Example HAL file loading a QtVCP panel

```
# load realtime components
loadrt threads
loadrt classicladder_rt

# load non-realtime programs
loadusr classicladder
loadusr -Wn my_panel qtvcp my_panel.ui  # (1)

# add components to thread
addf classicladder.0.refresh thread1


# connect pins
net bit-input1     test_panel.checkbox_1        classicladder.0.in-00
net bit-hide       test_panel.checkbox_4        classicladder.0.hide_gui

net bit-output1    test_panel.led_1             classicladder.0.out-00

net s32-in1        test_panel.doublescale_1-s   classicladder.0.s32in-00

# start thread
start
```

(1) En esta caso cargamos `qtvcp` usando **`-Wn`** el cual espera a que se termine de cargar el panel antes de ejecutar a continuación el siguiente comando HAL.
Esto es para *asegurar que los pines HAL del panel creado estén terminados* en caso de ser usados en lo que resta del archivo.

12.5.4. Build A Simple Clean-sheet Custom Screen

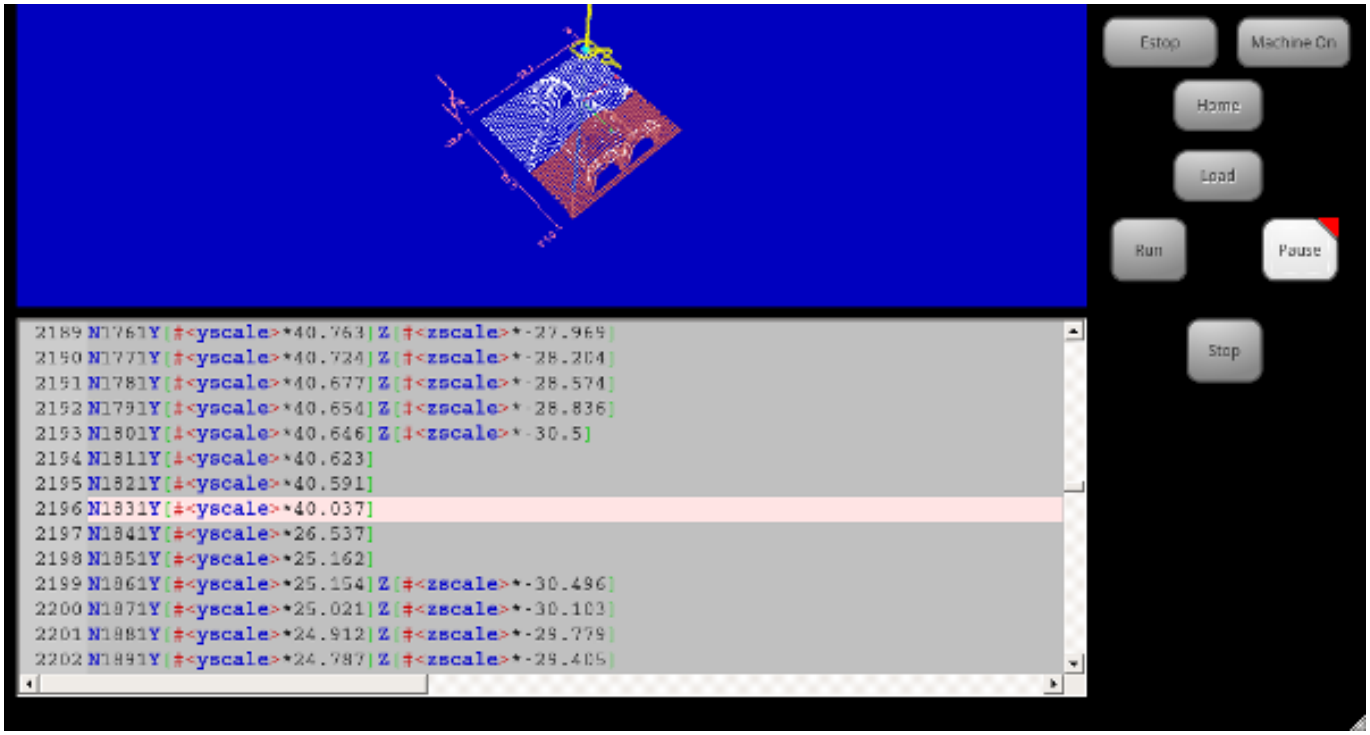


Figura 12.66: QtVCP Ugly custom screen

12.5.4.1. Descripción general

To build a panel or screen:

- Use Qt Designer to build a design you like and save it to your configuration folder with a name of your choice, ending with `.ui`
- Modify the configuration INI file to load QtVCP using your new `.ui` file.
- Then connect any required HAL pins in a HAL file.

12.5.4.2. Get Qt Designer To Include LinuxCNC Widgets

Install Qt Designer Primero debe tener **Qt Designer instalado**.
Estos comandos deberían agregarlo a su sistema, o utilice su administrador de paquetes para lo mismo:

```
sudo apt-get install qttools5-dev-tools qttools5-dev libpython3-dev
```

Add qtvcp_plugin.py link to the Qt Designer Search Path Then you must add a link to the `qtvcp_plugin.py` in one of the folders that Qt Designer will search into.

In a *RIP* version of LinuxCNC `qtvcp_plugin.py` will be:

`'~/LINUXCNC_PROJECT_NAME/lib/python/qtvcp/plugins/qtvcp_plugin.py'`

For a *Package installed* version it should be:

```
'usr/lib/python2.7/qtvcp/plugins/qtvcp_plugin.py' or
'usr/lib/python2.7/dist-packages/qtvcp/plugins/qtvcp_plugin.py'
```

Cree un enlace simbólico al archivo anterior y muévalo a uno de los lugares en los que busca Qt Designer.

Qt Designer searches in these two place for links (pick one):

```
'/usr/lib/x86_64-linux-gnu/qt5/plugins/designer/python' or
'~/.designer/plugins/python'
```

Es posible que tenga que crear la carpeta `plugins/python`.

Start Qt Designer:

- Para una *RIP install*:
  Abra una terminal, configure el entorno para LinuxCNC <1>, luego cargue Qt Designer <2> con :

```
. scripts/rip-environment    (1)
designer -qt=5               (2)
```

- Para una *instalación de paquete*:
  Abra una terminal y teclee:

```
designer -qt=5
```

Si todo va bien, se lanzará QtDesigner y verá los widgets de LinuxCNC seleccionables en el lado izquierdo.

12.5.4.3. Build The Screen `.ui` File

Create MainWindow Widget Cuando Qt Designer se inicia por primera vez, aparece un *cuadro de diálogo 'Nuevo formulario'*.
Elija *'Ventana principal'* y presione el botón *'Crear'*.
Se muestra un *widget* `MainWindow`.

We are going to make this window a specific non resizeable size:

Set `MainWindow` Minimum and Maximum Size

- Grab the corner of the window and resize to an appropriate size, say 1000x600.

- Haga clic derecho en la ventana y haga clic en *tamaño mínimo*.
- Do it again and set *maximum size*.

Our sample widget will now not be resizable.

Add the ScreenOptions Widget Drag and drop the `ScreenOptions` widget anywhere onto the main window.

This widget doesn't add anything visually but sets up some **common options**.

It's recommended to always *add this widget before any other*.

Right click on the main window, not the `ScreenOptions` widget, and set the *layout* as vertical to make the `ScreenOptions` fullsized.

Add Panel Content On the right hand side there is a panel with tabs for a *Property editor* and an *Object inspector*.

En el inspector de objetos, haga clic en *ScreenOptions*.
Entonces cambie al Editor de Propiedades y bajo el encabezado *ScreenOptions*, alterne **`filedialog_option`**.

Arrastre y suelte un *widget* **`GCodeGraphics`** y un *widget* **`GcodeEditor`**.
Colóquelos y cambie su tamaño como mejor le parezca, dejando espacio para botones.

Add Action Buttons Add 7 action buttons on to the main window.

If you double click the button, you can add text.
Edit the button labels for *Estop*, *Machine On*, *Home*, *Load*, *Run*, *Pause* and *stop*.

Action buttons *default to no action* so we must change the properties for defined functions. You can edit the properties:

- directly in the *property editor* on the right side of Qt Designer, or
- conveniently, left double clicking on the button to launch a *properties dialog* that allows selecting actions while only displaying relevant data to the action.

We will describe the convenient way first:

- Haga clic derecho sobre el botón *Encender máquina* y seleccione *Establecer acciones*.
- When the dialog displays, use the combobox to navigate to `MACHINE CONTROLS - Machine On`.
- In this case there is no option for this action so select *OK*.

Now the button will turn the machine on when pressed.

Y ahora el camino directo con el editor de propiedades de Qt Designer:

- Select the *Machine On* button.
- Go to the Property Editor on the right side of Qt Designer.
- Scroll down until you find the *ActionButton* heading.
- Click the `machine_on` action checkbox you will see in the list of properties and values.

The button will now control machine on/off.

Do the same for all the other button with the addition of:

- Con el botón *Inicio* también debemos cambiar la propiedad `joint_number` a `-1`.
  Esto le dice al controlador que *lleve a casa a todos los ejes* en lugar de un eje específico.
- Con el botón *Pause*:
  - Under the `Indicated_PushButton` heading check the `indicator_option`.
  - Under the `QAbstactButton` heading check `checkable`.

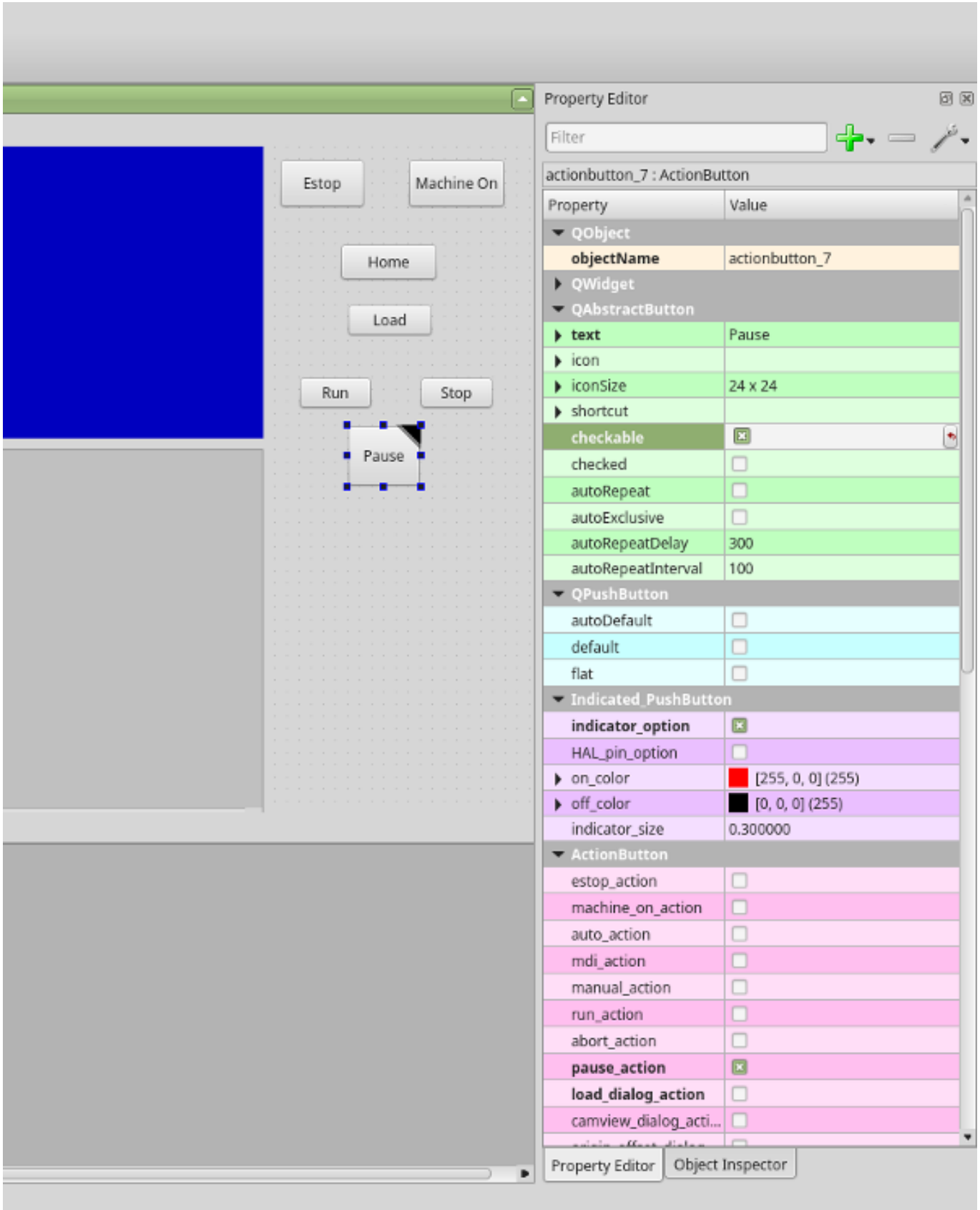


Figura 12.67: Qt Designer: Selecting Pause Button’s Properties

Save The .ui File We then need to save this design as `tester.ui` in the `sim/qtvcp` folder.

We are saving it as *tester* as that is a file name that QtVCP recognizes and will use a built in handler file to display it.

12.5.4.4. Handler file

A handler file is **required**.

Permite que personalizaciones sean escritas en Python.

Por ejemplo, los *controles del teclado* generalmente se escriben en el archivo del controlador.

En este ejemplo, el archivo integrado `tester_handler.py` se usa automáticamente: Hace lo mínimo requerido para mostrar la pantalla definida `tester.ui` y hacer trote básico por teclado.

12.5.4.5. Configuración de INI

Sección [DISPLAY] If you are using QtVCP to make a CNC control screen, under the *INI file* `[DISPLAY]` heading, set:

```
DISPLAY = qtvcp <nombre-de-pantalla>
```

`<screen_name>` is the *base name* of the `.ui` and `_handler.py` files.

In our example there is already a sim configuration called tester, that we will use to display our test screen.

Sección [HAL] If your screen used *widgets with HAL pins*, then you must **connect them in a HAL file**.

QtVCP looks in the *INI file*, under the `[HAL]` heading for the entries below:

`POSTGUI_HALFILE=<nombre-de-archivo>`
: Typically `<filename>` would be `+<screen_name>_postgui.hal+`, but can be any legal filename.
You can have *multiple `POSTGUI_HALFILE` lines* in the INI: each will be run one after the other in the order they appear.
These commands are *executed after the screen is built*, guaranteeing the widget HAL pins are available.

`POSTGUI_HALCMD=<command>`
: `<command>` would be *any valid HAL command*.
You can have *multiple `POSTGUI_HALCMD` lines* in the INI: each will be run one after the other in the order they appear.
To guaranty the widget HAL pins are available, these commands are executed:

- *after the screen is built*,
- *after all the POSTGUI_HALFILEs are run*.

In our example there are no HAL pins to connect.

12.5.5. Handler File In Detail

Handler files are used to *create custom controls using Python*.

12.5.5.1. Descripción general

Aquí un archivo manejador de muestra.

Está dividido en secciones para facilitar la discusión.

```
############################
# **** IMPORT SECTION **** #
############################
import sys
import os
import linuxcnc

from PyQt5 import QtCore, QtWidgets

from qtvcp.widgets.mdi_line import MDILine as MDI_WIDGET
from qtvcp.widgets.gcode_editor import GcodeEditor as GCODE
from qtvcp.lib.keybindings import Keylookup
from qtvcp.core import Status, Action

# Set up logging
from qtvcp import logger
LOG = logger.getLogger(__name__)

# Set the log level for this module
#LOG.setLevel(logger.INFO) # One of DEBUG, INFO, WARNING, ERROR, CRITICAL

############################################
# **** INSTANTIATE LIBRARIES SECTION **** #
############################################

KEYBIND = Keylookup()
STATUS = Status()
ACTION = Action()
###################################
# **** HANDLER CLASS SECTION **** #
###################################

class HandlerClass:

    ########################
    # **** INITIALIZE **** #
    ########################
    # widgets allows access to  widgets from the QtVCP files
    # at this point the widgets and hal pins are not instantiated
    def __init__(self, halcomp,widgets,paths):
        self.hal = halcomp
        self.w = widgets
        self.PATHS = paths

    ########################################
    # SPECIAL FUNCTIONS SECTION            #
    ########################################

    # at this point:
    # the widgets are instantiated.
    # the HAL pins are built but HAL is not set ready
    # This is where you make HAL pins or initialize state of widgets etc
    def initialized__(self):
        pass

    def processed_key_event__(self,receiver,event,is_pressed,key,code,shift,cntrl):
        # when typing in MDI, we don't want keybinding to call functions
```

```
        # so we catch and process the events directly.
        # We do want ESC, F1 and F2 to call keybinding functions though
        if code not in(QtCore.Qt.Key_Escape,QtCore.Qt.Key_F1 ,QtCore.Qt.Key_F2,
                    QtCore.Qt.Key_F3,QtCore.Qt.Key_F5,QtCore.Qt.Key_F5):

            # search for the top widget of whatever widget received the event
            # then check if it is one we want the keypress events to go to
            flag = False
            receiver2 = receiver
            while receiver2 is not None and not flag:
                if isinstance(receiver2, QtWidgets.QDialog):
                    flag = True
                    break
                if isinstance(receiver2, MDI_WIDGET):
                    flag = True
                    break
                if isinstance(receiver2, GCODE):
                    flag = True
                    break
                receiver2 = receiver2.parent()

            if flag:
                if isinstance(receiver2, GCODE):
                    # if in manual do our keybindings - otherwise
                    # send events to G-code widget
                    if STATUS.is_man_mode() == False:
                        if is_pressed:
                            receiver.keyPressEvent(event)
                            event.accept()
                        return True
                elif is_pressed:
                    receiver.keyPressEvent(event)
                    event.accept()
                    return True
                else:
                    event.accept()
                    return True

        if event.isAutoRepeat():return True

        # ok if we got here then try keybindings
        try:
            return KEYBIND.call(self,event,is_pressed,shift,cntrl)
        except NameError as e:
            LOG.debug('Exception in KEYBINDING: {}'.format (e))
        except Exception as e:
            LOG.debug('Exception in KEYBINDING:', exc_info=e)
            print('Error in, or no function for: %s in handler file for-%s'%(KEYBIND. ↩
                convert(event),key))
            return False

    ########################
    # CALLBACKS FROM STATUS #
    ########################

    ######################
    # CALLBACKS FROM FORM #
    ######################

    ####################
    # GENERAL FUNCTIONS #
    ####################
```

```
    # keyboard jogging from key binding calls
    # double the rate if fast is true
    def kb_jog(self, state, joint, direction, fast = False, linear = True):
        if not STATUS.is_man_mode() or not STATUS.machine_is_on():
            return
        if linear:
            distance = STATUS.get_jog_increment()
            rate = STATUS.get_jograte()/60
        else:
            distance = STATUS.get_jog_increment_angular()
            rate = STATUS.get_jograte_angular()/60
        if state:
            if fast:
                rate = rate * 2
            ACTION.JOG(joint, direction, rate, distance)
        else:
            ACTION.JOG(joint, 0, 0, 0)

    ######################
    # KEY BINDING CALLS #
    ######################

    # Machine control
    def on_keycall_ESTOP(self,event,state,shift,cntrl):
        if state:
            ACTION.SET_ESTOP_STATE(STATUS.estop_is_clear())
    def on_keycall_POWER(self,event,state,shift,cntrl):
        if state:
            ACTION.SET_MACHINE_STATE(not STATUS.machine_is_on())
    def on_keycall_HOME(self,event,state,shift,cntrl):
        if state:
            if STATUS.is_all_homed():
                ACTION.SET_MACHINE_UNHOMED(-1)
            else:
                ACTION.SET_MACHINE_HOMING(-1)
    def on_keycall_ABORT(self,event,state,shift,cntrl):
        if state:
            if STATUS.stat.interp_state == linuxcnc.INTERP_IDLE:
                self.w.close()
            else:
                self.cmnd.abort()

    # Linear Jogging
    def on_keycall_XPOS(self,event,state,shift,cntrl):
        self.kb_jog(state, 0, 1, shift)

    def on_keycall_XNEG(self,event,state,shift,cntrl):
        self.kb_jog(state, 0, -1, shift)

    def on_keycall_YPOS(self,event,state,shift,cntrl):
        self.kb_jog(state, 1, 1, shift)

    def on_keycall_YNEG(self,event,state,shift,cntrl):
        self.kb_jog(state, 1, -1, shift)

    def on_keycall_ZPOS(self,event,state,shift,cntrl):
        self.kb_jog(state, 2, 1, shift)

    def on_keycall_ZNEG(self,event,state,shift,cntrl):
        self.kb_jog(state, 2, -1, shift)
```

```
    def on_keycall_APOS(self,event,state,shift,cntrl):
        pass
        #self.kb_jog(state, 3, 1, shift, False)

    def on_keycall_ANEG(self,event,state,shift,cntrl):
        pass
        #self.kb_jog(state, 3, -1, shift, linear=False)

    ############################
    # **** closing event **** #
    ############################

    ##############################
    # required class boiler code #
    ##############################

    def __getitem__(self, item):
        return getattr(self, item)
    def __setitem__(self, item, value):
        return setattr(self, item, value)

################################
# required handler boiler code #
################################

def get_handlers(halcomp,widgets,paths):
     return [HandlerClass(halcomp,widgets,paths)]
```

12.5.5.2. IMPORT Section

This section is for **importing required library modules** for your screen.

Sería típico importar las librerías de QtVCP *keybinding*, *Status* y *Action*.

12.5.5.3. INSTANTIATE LIBRARIES Section

By instantiating the libraries here we **create global reference**.

You can note this by the commands that don't have `self.` in front of them.

By convention we *capitalize the names of globally referenced libraries*.

12.5.5.4. HANDLER CLASS Section

El **código personalizado** se coloca *en una clase para que QtVCP* pueda utilizarlo_.

Esta es la definición de la clase manejadora.

12.5.5.5. INITIALIZE Section

Like all Python libraries the **+`__init__`+ function** is called when the library is *first instantiated*.

This is where you would set up *defaults*, as well as *reference variables* and *global variables*.

The widget references are not available at this point.

The variables `halcomp`, `widgets` and `paths` give access to QtVCP's HAL component, widgets, and path info respectively.

12.5.5.6. SPECIAL FUNCTIONS Section

There are several *special functions* that QtVCP looks for in the handler file. If QtVCP finds these it will call them, if not it will silently ignore them.

`class_patch __ (self):`
Los *parches de clase*, también conocidos como *parches de mono*, le permiten **anular llamadas a funciones en un módulo importado**.
Los parches de clase deben hacerse *antes de que el módulo se instancia* y *modifica todas las instancias* hechas después de eso.
Un ejemplo podría ser parchear llamadas de botones del editor de código G para llamar a funciones en el archivo del controlador en su lugar.
La función de parche de clase definida aquí será llamada con la instancia de HandlerClass como *self* en lugar de la instancia de clase parchada. Esto puede hacer mas difícil el acceso a las variables/funciones de la clase parchada.
Cuando se parcha una clase fuera de clase HandlerClass, la llamada a la función usará la instancia de clase parchada como *self*.

`initialized__(self):`
Esta función se *llama después de que se construyen los widgets y los pines HAL*.
Puede manipular los widgets y los pines HAL o agregar más pines HAL aquí.
Típicamente pueden ser

- preferences checked and set,
- styles applied to widgets,
- status of LinuxCNC connected to functions.
- keybindings would be added.

`pre_hal_init__(self):`
This function is called before the HAL-ified widgets have their hal_init_ function called.
Some property changes need to be done before HAL_init is called on the widget.

`after_override__(self):`
This function is called after the optional override file is loaded but
before the optional HAL file is loaded or HAL component is set ready.

`processed_key_event__(self,receiver,event,is_pressed,key,code,shift,cntrl):`
Esta función se llama para facilitar el *desplazamiento del teclado*, etc
Al usar la *biblioteca* `keybinding` se pueden agregar fácilmente funciones vinculadas a las pulsaciones de teclas.

`keypress_event__(self,receiver, event):`
Esta función proporciona **eventos de pulsación de tecla sin procesar**.
Tiene *precedencia sobre* el `evento_de_tecla_procesado`.

`keyrelease_event__(receiver, event):`
Esta función proporciona **eventos de liberación de tecla sin procesar**.
Toma *precedencia sobre* el `evento_de_tecla_procesado`.

`before_loop__(self):`
This function is *called just before the Qt event loop is entered*. At that point, all widgets/libraries/initialization code has completed and the screen is already displayed.

`system_shutdown_request__(self):`
Si está presente, esta función **anula la función normal llamada para un apagado total del sistema**.
Podría usarse para realizar *tareas de limpieza antes del cierre*.

Bibliografía

[1] + El sistema no se apagará si usa esta función, tendrás que hacerlo tu mismo. QtVCP/LinuxCNC se apagará sin un aviso después de que esta función regrese.

`closing_cleanup__(self):`
This function is *called just before the screen closes*. It can be used to do cleanup before closing.

12.5.5.7. STATUS CALLBACKS Section

By convention this is where you would put functions that are **callbacks from STATUS definitions**.

12.5.5.8. CALLBACKS FROM FORM Section

Por convención, aquí es donde pondría funciones que son **devoluciones de llamada desde los widgets que ha conectado a MainWindow** con el editor QtDesigner.

12.5.5.9. GENERAL FUNCTIONS Section

Por convención, aquí es donde pones tus **funciones generales**.

12.5.5.10. KEY BINDING Section

Si esta *utilizando la biblioteca `keybinding`* aquí es donde coloca sus **rutinas de llamadas clave personalizadas**.

La firma de la función es:

```
def on_keycall_KEY(self,event,state,shift,cntrl):
    if state:
        self.do_something_function()
```

KEY being the code (from the keybindings library) for the desired key.

12.5.5.11. CLOSING EVENT Section

Putting the **`closeEvent` function here will catch closing events**.

This *replaces any predefined `closeEvent`* function from QtVCP.

```
def closeEvent(self, event):
    self.do_something()
    event.accept()
```

---

nota
It is usually better to use the special `closing_cleanup__` function.

12.5.6. Connecting Widgets to Python Code

It is possible to connect widgets to Python code using **signals and slots**.

In this way you can:

- *Give new functions to LinuxCNC widgets*, or
- *Utilize standard Qt widgets to control LinuxCNC*.

12.5.6.1. Descripción general

En el editor de Qt Designer:

- You *create user function slots*
- You *connect the slots to widgets using signals*.

In the handler file:

- You *create the slot's functions* defined in Qt Designer.

12.5.6.2. Using Qt Designer to add Slots

When you have loaded your screen into Qt Designer, add a plain `PushButton` to the screen.
You could change the name of the button to something interesting like *test_button*.

There are two ways to edit connections - This is the graphical way.

- There is a button in the top tool bar of Qt Designer for editing signals. After pushing it, if you click-and-hold on the button it will show an arrow (looks like a ground signal from electrical schematic).
- Slide this arrow to a part of the main window that does not have widgets on it.
- A *Configure Connections* dialog will pop up.
  - The list on the left are the available signals from the widget.
  - The list on the right are the available slots on the main window and you can add to it.
- Pick the signal `clicked()` - this makes the slots side available.
- Click *Edit* on the slots list.
- A *Slots/Signals of MainWindow* dialog will pop up.
- On the slots list at the top there is a + icon - click it.
- You can now edit a new slot name.
- Erase the default name `slot()` and change it to `test_button()`.
- Press the *OK* button.
- You'll be back to the *Configure Connections* dialog.
- Ahora puede seleccionar su ranura nueva en la lista de ranuras.
- Then press *OK* and save the file.

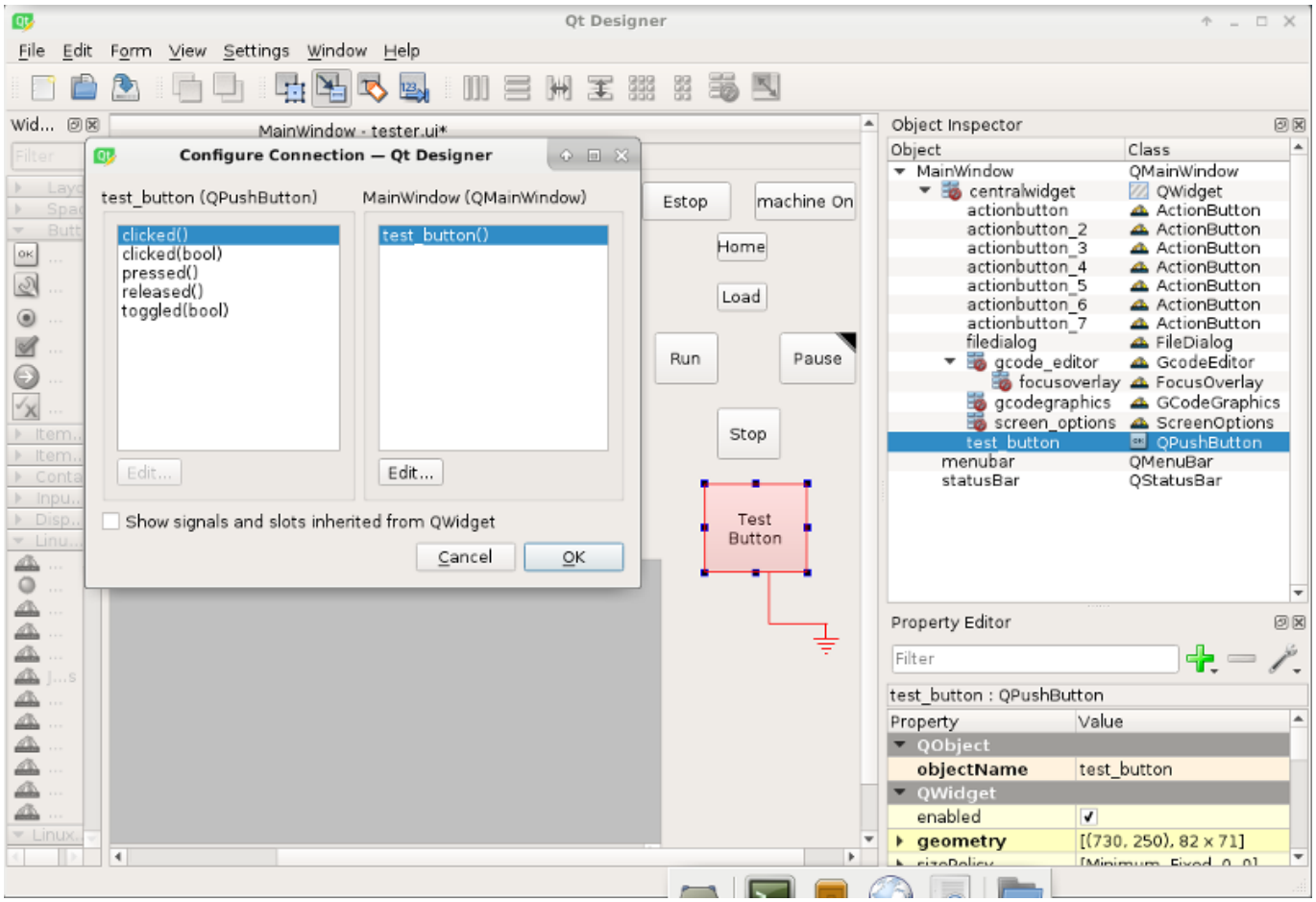


Figura 12.68: Selección de señal/ranura de QtDesigner

12.5.6.3. Python Handler Changes

Ahora debe **agregar la función al archivo manejador**.

The function signature is **`def slot_name(self):`**.

For our example, we will add some code to print the widget name:

```
def test_button(self):
    name = self.w.sender().text()
    print(name)
```

Add this code under the section named:

```
##########################################
# devoluciones de llamada del formulario #
##########################################
```

De hecho, no importa en qué parte de la clase de controlador coloque los comandos pero por convención, aquí es donde ponerlo.

Guarde el archivo manejador.

Now when you load your screen and press the button it should print the name of the button in the terminal.

12.5.7. More Information

QtVCP Builtin Virtual Control Panels

QtVCP Widgets

QtVCP Libraries

Qt Vismach

QtVCP Handler File Code Snippets

QtVCP Development

QtVCP Custom Qt Designer Widgets

12.6. QtVCP Virtual Control Panels

QtVCP can be used to **create control panels** that interface with *HAL*.

12.6.1. Builtin Virtual Control Panels

Hay varios **paneles HAL interconstruídos** disponibles.

In a terminal type `qtvcp list` to see a list.

12.6.1.1. `copy`

Used for **copying QtVCP's builtin Screens/VCP Panels/QtVismach code to a folder** so one can *customize* it.

En una terminal ejecute:

```
qtvcp copy
```

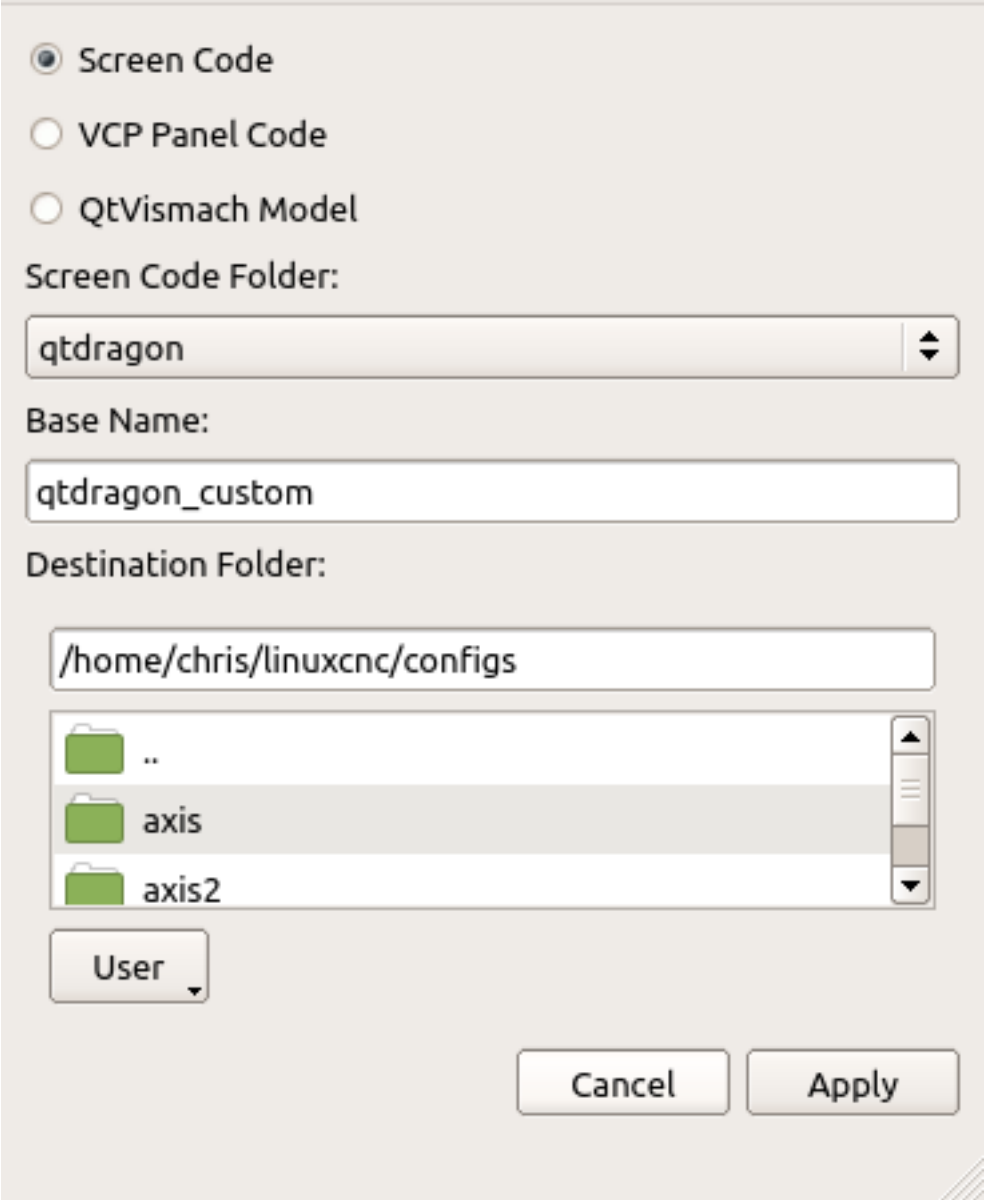
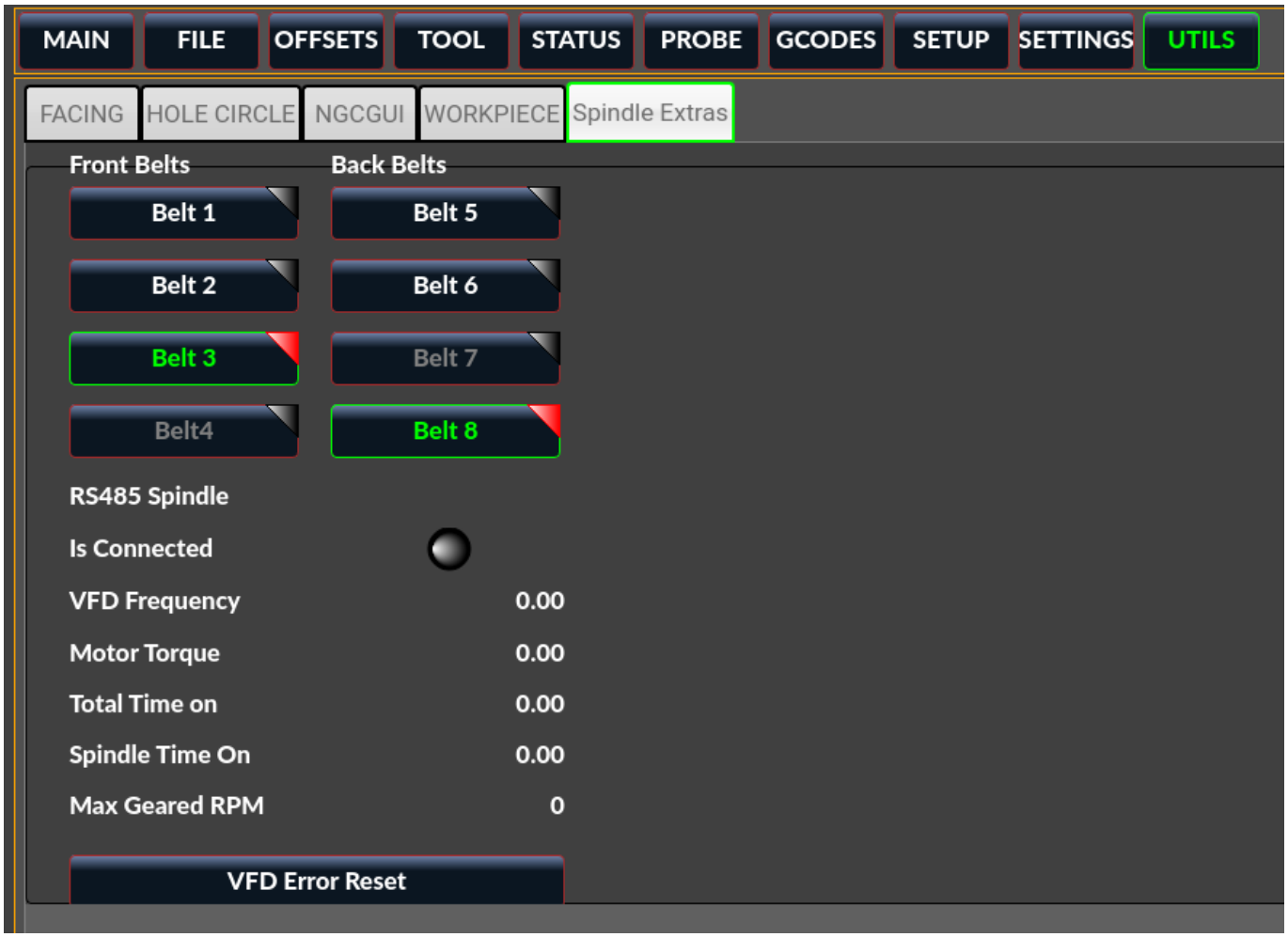


Figura 12.69: QtVCP `copy` Dialog - Screen, VCP Panel or QtVismach Code Copy Panel

12.6.1.2. spindle_belts

This panel is designed to display additional RS485 VFD data and also to configure a 4 sheave, 2 belt spindle drive via a series of buttons.



In addition, it is also a useful template to use for your custom panel because it includes:

- Display of additional HAL data
- Buttons and button groups
- Dynamic changes to button enabled/disabled state based on the state of other buttons
- Saving data to the qtdragon.prefs file
- Custom button to reset the VFD

Modify this panel to suit your own requirements. Most common features are used. The advantage of using panels is that it separates your custom display code from the qtdragon core code so upgrading the system will not break your customization.

- A spindle drive (for instance VFDMOD)
- A custom component that scales the VFD frequency to obtain the desired spindle speed.
- A belt driven spindle that uses two belts and an intermediate idler pulley much like a drill press.
- Connect the input pins qtdragon.belts.<pin-name> in your postgui HAL file.

The belts are broken into two button groups, the front belts and the rear belts. These are numbered as per the plate on the machine. Buttons in a group are mutually exclusive, i.e., only one can be selected in the group.

Additionally, it’s not possible to have both belts on the same level with this kind of mechanism because you cannot fit two belts to the one idler pulley sheave. So if a belt is selected, its opposite button is disabled. E.g., if belt 3 is selected, belt 7 is disabled.

Add these lines to the [DISPLAY] section in your .ini file
The example tab_location is for the QtDragon screen.

```
EMBED_TAB_NAME=Spindle Extras
EMBED_TAB_COMMAND=qtvcp spindle_belts
EMBED_TAB_LOCATION=tabWidget_utilities
```

Here is how to load `spindle_belt` from a HAL script:

```
loadusr qtvcp spindle_belts
```

Customizing the panel:

- Copy the files located in /user/share/qtvcp/qtdragon/panels/belts to: ~/linuxcnc/configs/<my_configuratio
(you can use the copy dialog panel to do this)
- Edit belts.ui with designer.
- Edit belts_handler.py with a text editor
- Conecte los pines relevantes en un archivo postgui.hal
- Make sure your postgui file is loaded by your .ini file.

For information on the finer points, consult the QtVCP and QtDragon documentation. The Python handler file also provides a useful template for any custom panel.

12.6.1.3. `test_dial`

- This panel has a **dial that adjusts S32 and Float HAL output pins**.
- The dial’s range can be adjusted from a drop down menu.
- The output can be scaled with the `spinbox`.
- A `combobox` can be used to automatically select and connect to a signal.

```
loadusr qtvcp test_dial
```

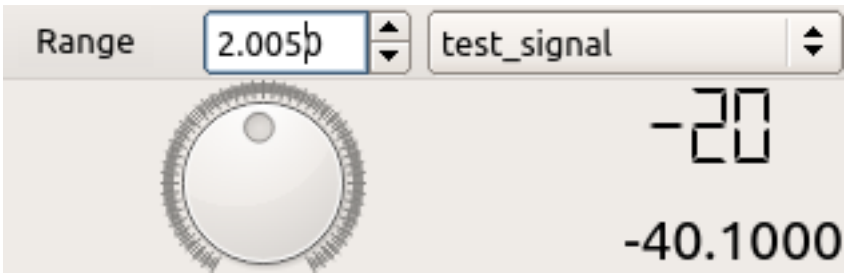


Figura 12.70: QtVCP `test_dial` Panel - Test Dial VCP

12.6.1.4. test_button

- This panel has a **button that will set a HAL pin**.
- The button can be selected as a *momentary* or a *toggle* button.
- A HAL pin will be created that follows the button state.
- The button's *indicator color* can be adjusted from a drop down menu.
- A HAL pin or signal can be selected to follow the button state.
- You can add more buttons from the drop down menu.
- You can load a Halmeter from the drop down menu.
- You can load a test LED from the drop down menu.
- The button can be detached from the main windows.

Here is how to load `test_button` from a HAL script:

```
loadusr qtvcp test_button
loadusr qtvcp -o 4 test_button
```

The `-o` switch sets how many buttons the panel starts with.
If loading directly from a terminal omit the `loadusr`.

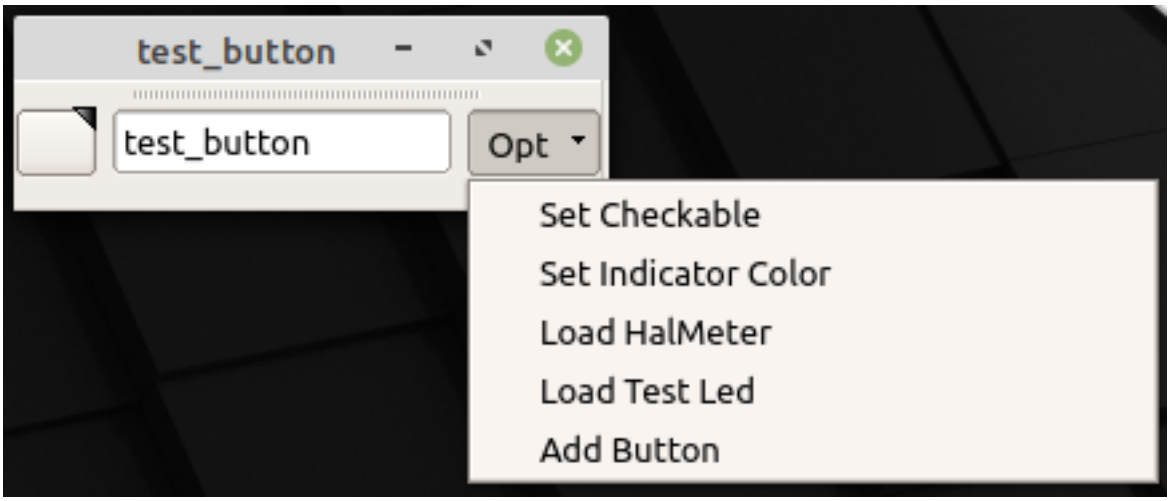


Figura 12.71: QtVCP `test_button` - Test Button VCP

12.6.1.5. test_led

- This panel has an **LED that can selected to watch HAL bit pins/signals**.
- The LED's color can be adjusted from a drop down menu.
- The text box and state can be output as speech if sound is selected.
- A `combobox` can be used to automatically select and connect to a pin/signal.
- You can add more LEDs from the drop down menu.
- The LED can be detached from the main windows.

Here is how to load `test_led` from a HAL script:

```
loadusr qtvcp test_led
loadusr qtvcp -o 4 test_led
```

The `-o` switch sets how many LEDs the panel starts with.
If loading directly from a terminal omit the *loadusr*.

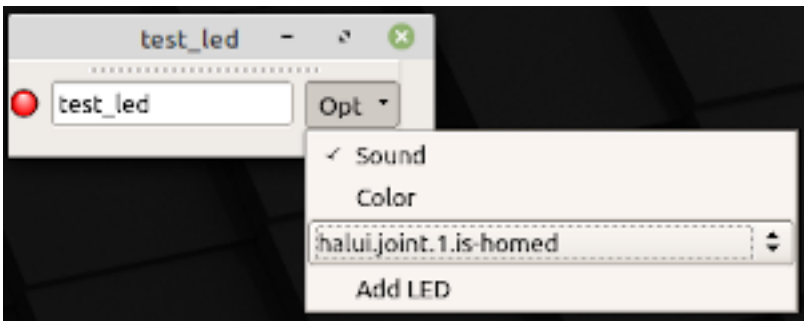


Figura 12.72: QtVCP `test_dial` Panel - Test LED VCP

12.6.1.6. `test_panel`

Collection of useful widgets for testing HAL component, including speech of LED state.

```
loadusr qtvcp test_panel
```

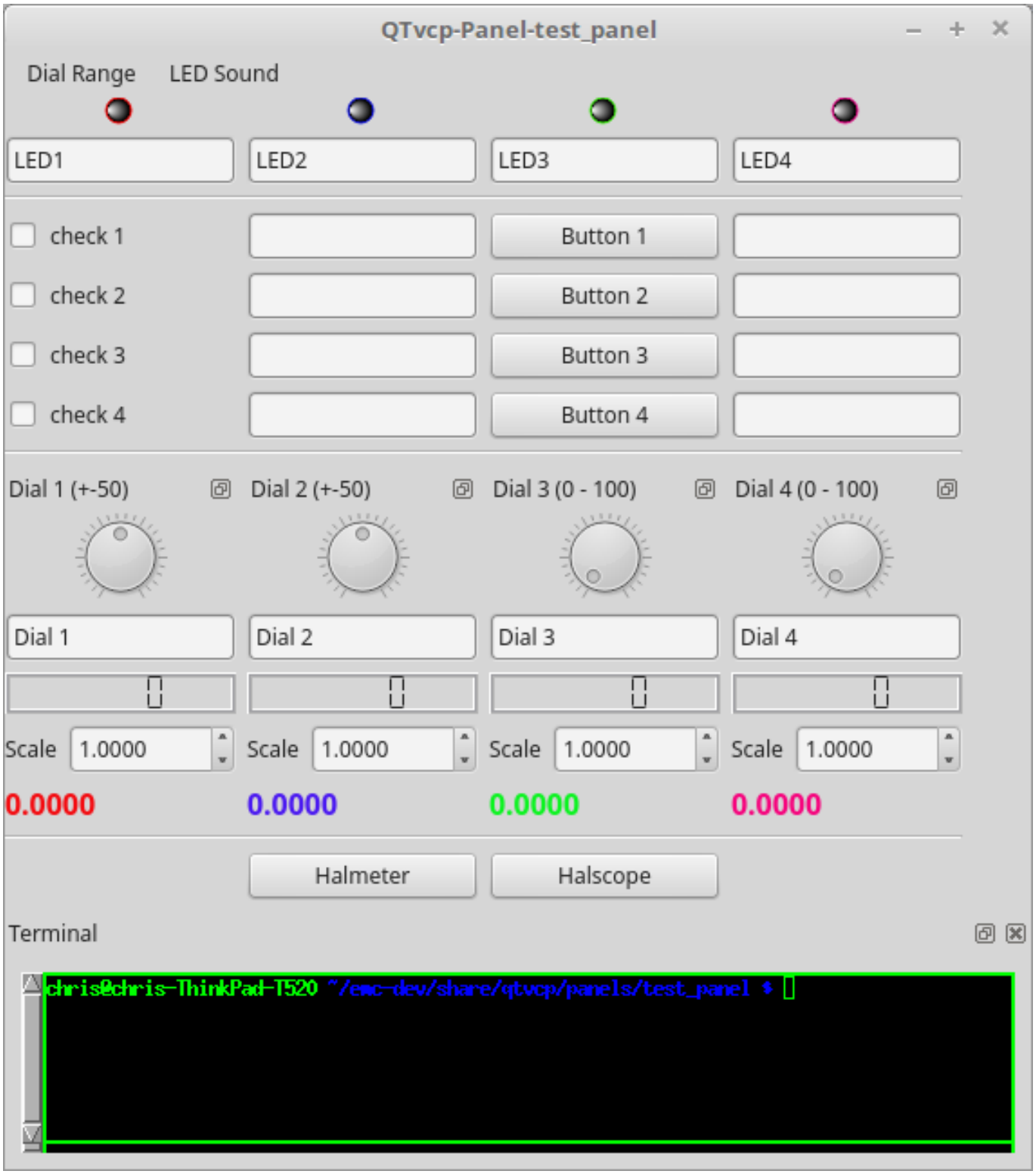


Figura 12.73: QtVCP `test_panel` - HAL Component Testing Panel

12.6.1.7. `cam_align`

Un **widget de visualización de cámara para alineación rotacional**.



Figura 12.74: QtVCP `cam_align` Panel - Camera Based Alignment Panel

Utilización Add these lines to the INI file:

```
[DISPLAY]
EMBED_TAB_NAME = cam_align
EMBED_TAB_COMMAND = halcmd loadusr -Wn qtvcp_embed qtvcp -d -c qtvcp_embed -x {XID}  ↩
    cam_align
# The following line is needed if embedding in GMOCCAPY
EMBED_TAB_LOCATION = ntb_preview
```

nota
All <options> must appear before the `cam_align` panel name.

Qtvcp Options

- `-c NAME` HAL component name. Default is to use the UI file name.
- `-d` Debugging on. or remove for no minimum output

- `-v` Verbose debugging on. You can find all available resolutions.
- `-x {XID}` used for embedding into AXIS or Gmoccapy
- `-o <option>` Options passed to cam_align. can use multiple -o entries

Cam_align Options These are the available -o option:

- `size=400,400` Size of the embedded window (width,height)
- `imagesize=300,300` Size of the image inside the window (width,height)
- `rotincr=5` Sets the increment of the crosshair rotation. (degrees)
- `xscale=100` Scales the image in X. A negative value will flip the image in X (percent)
- `yscale=100` Scales the image in Y. A negative value will flip the image in Y (percent)
- `camnumber=1` Sets what system camera to use
- `api=V4L2` Sets the opencv backend library for the camera
- `res=1280,720` Sets the requested resolution (width,height)

For instance you can add window width and height size, rotation increment, and camera number from the INI with -o options.

```
EMBED_TAB_COMMAND = halcmd loadusr -Wn qtvcp_embed qtvcp -d -c qtvcp_embed -x {XID} -o size ↩
    =400,400 -o rotincr=.2 -o camnumber=0 cam_align
```

Controles de ratón:

- left mouse single click - increase cross hair rotation one increment
- right mouse single click - decrease cross hair rotation one increment
- middle mouse single click - cycle through rotation increments
- left mouse hold and scroll - scroll camera zoom
- right mouse hold and scroll - scroll cross hair rotation angle
- mouse scroll only - scroll circle diameter
- left mouse double click - reset zoom
- right mouse double click - reset rotation
- middle mouse double click - reset circle diameter

To use the top buttons you have to assign a command (or a sub-routine). This could look like this:

```
[MDI_COMMAND_LIST]
MDI_COMMAND_CAM_ALIGN1=G10 L20 P1 X0 Y0,Set XY\nOrigin
MDI_COMMAND_CAM_ALIGN2=G0 X0 Y0,Go To\nOrigin
```

Where the first command is referring to the button ”SET origin” and the second to the button ”GOTO Origin”.
Note the comma and text after is optional - it will override the default button text.
These buttons are QtVCP action buttons and follow those rules.

12.6.1.8. `sim_panel`

Small control panel to **simulate MPG jogging controls etc** for simulated configurations.
The MPG, selection buttons and control buttons export HAL pins to connect to linuxcnc.
The selection and control group boxes can be hidden if not needed by using the *-o hide=* option.
groupBoxControl and *groupBoxSelection* are the widget names that can be hidden.
If you want to hide both, use a comma between them with no spaces.
The *-a* option will make the panel always-on-top of all windows.

```
loadusr qtvcp sim_panel
```

Here we load the panel with no MPG selection buttons and the always-on-top option.

```
loadusr qtvcp -a -o hide=groupBoxSelection sim_panel
```



Figura 12.75: QtVCP `sim_panel` - Simulated Controls Panel For Screen Testing.

12.6.1.9. `tool_dialog`

Manual tool change dialog that gives tool description.

```
loadusr -Wn tool_dialog qtvcp -o speak_on -o audio_on tool_dialog
```

Opciones:

- `-o notify_on` - *use desktop notify* dialogs instead of QtVCP native ones.
- `-o audio_on` - *play sound on tool change*
- `-o speak_on` - *speak announcement of tool change*

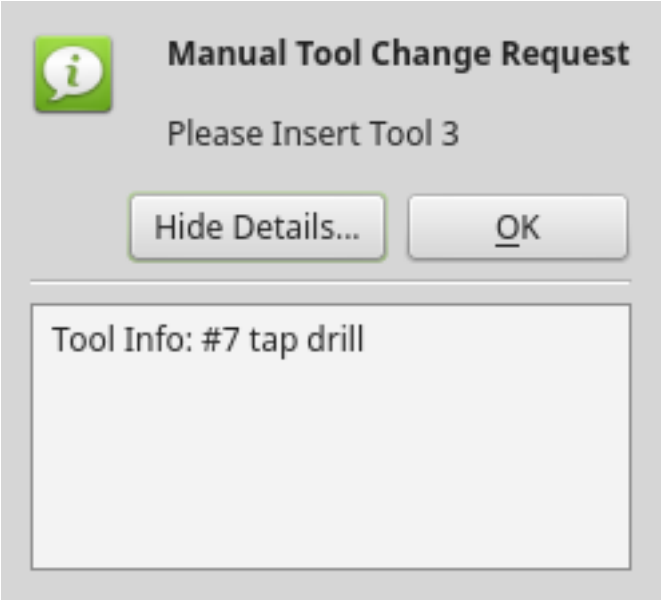


Figura 12.76: QtVCP `tool_dialog` - Manual Tool Change Dialog

12.6.2. `vismach` 3D Simulation Panels

These panels are prebuilt simulation of common machine types.
These are also embed-able in other screens such as AXIS or GMOCCAPY.

12.6.2.1. QtVCP `vismach_mill_xyz`

Vista 3D de OpenGL de una *máquina fresadora de 3 ejes*.

```
loadusr qtvcp vismach_mill_xyz
```

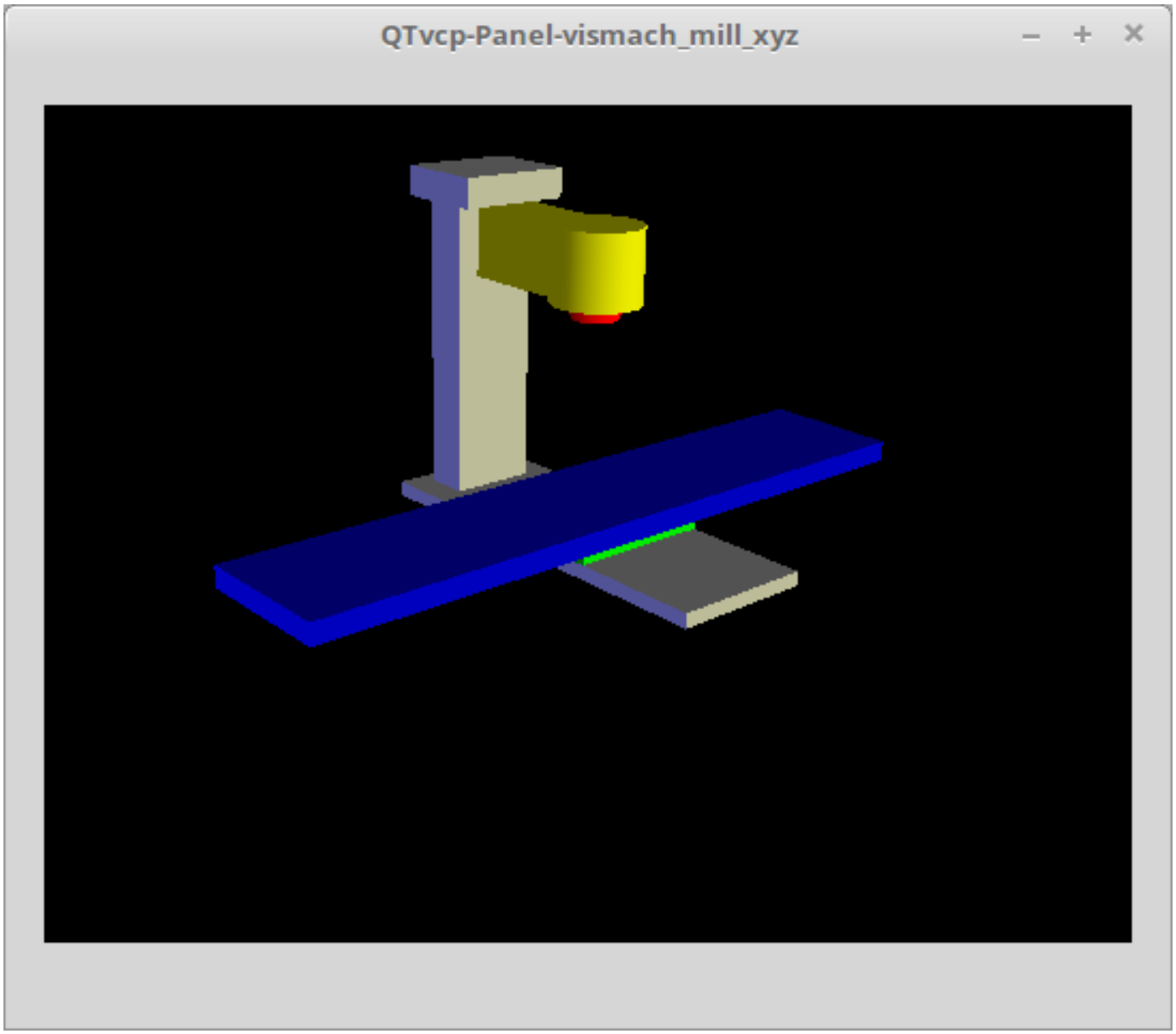


Figura 12.77: QtVCP `vismach_mill_xyz` - 3-Axis Mill 3D View Panel

12.6.2.2. QtVCP `vismach_router_atc`

3D OpenGL view of a *3-Axis router style, gantry bed milling machine*.
This particular panel shows how to define and connect the model parts in the handler file, rather then importing the pre-built model from QtVCP's vismach library.

```
loadusr qtvcp vismach_router_atc
```

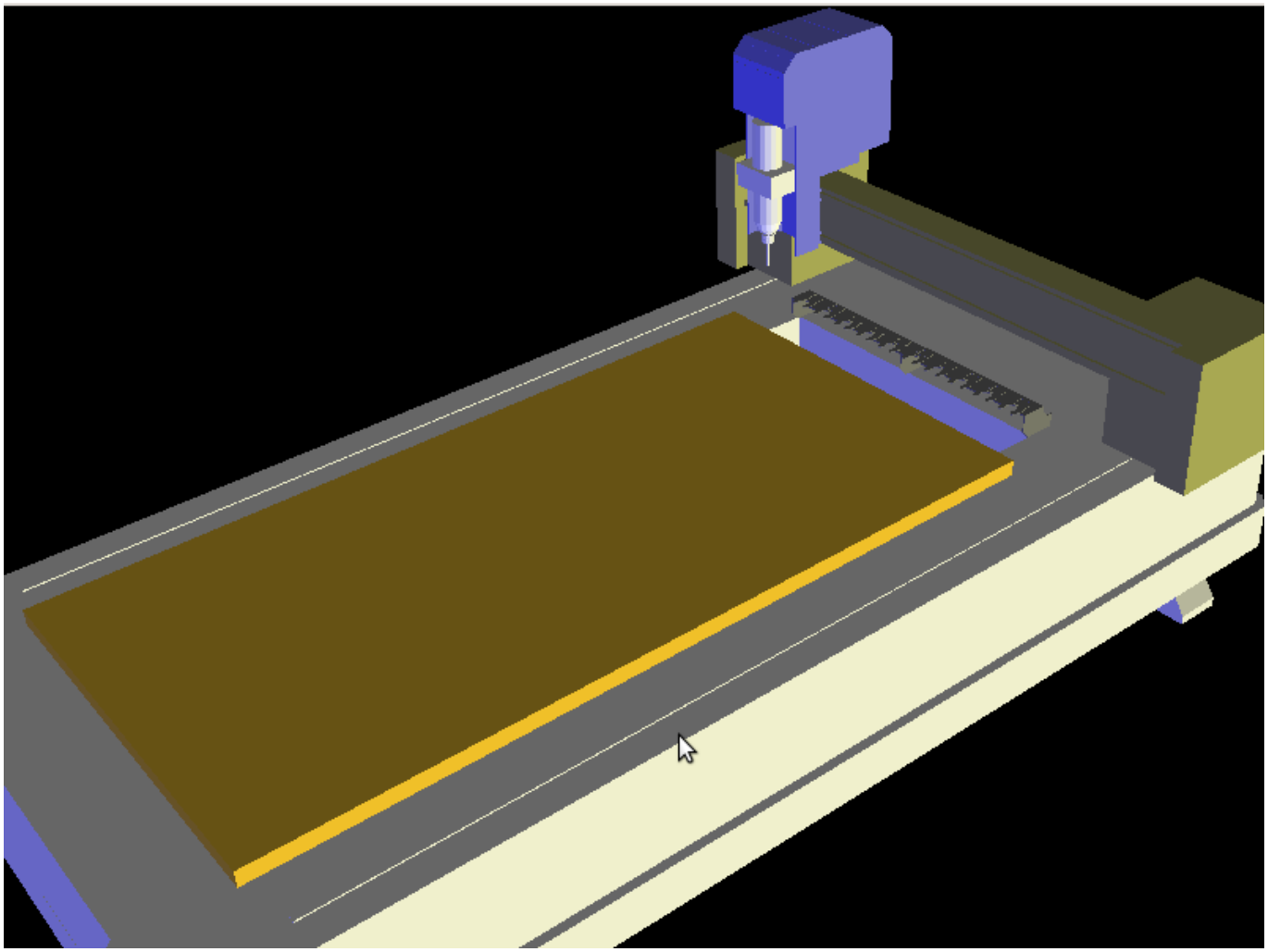


Figura 12.78: QtVCP `vismach_router_atc` - 3-Axis Gantry Bed Mill 3D View Panel

12.6.2.3. QtVCP `vismach_scara`

Vista 3D de OpenGL de una *máquina fresadora basada en SCARA*.

```
loadusr qtvcp vismach_scara
```

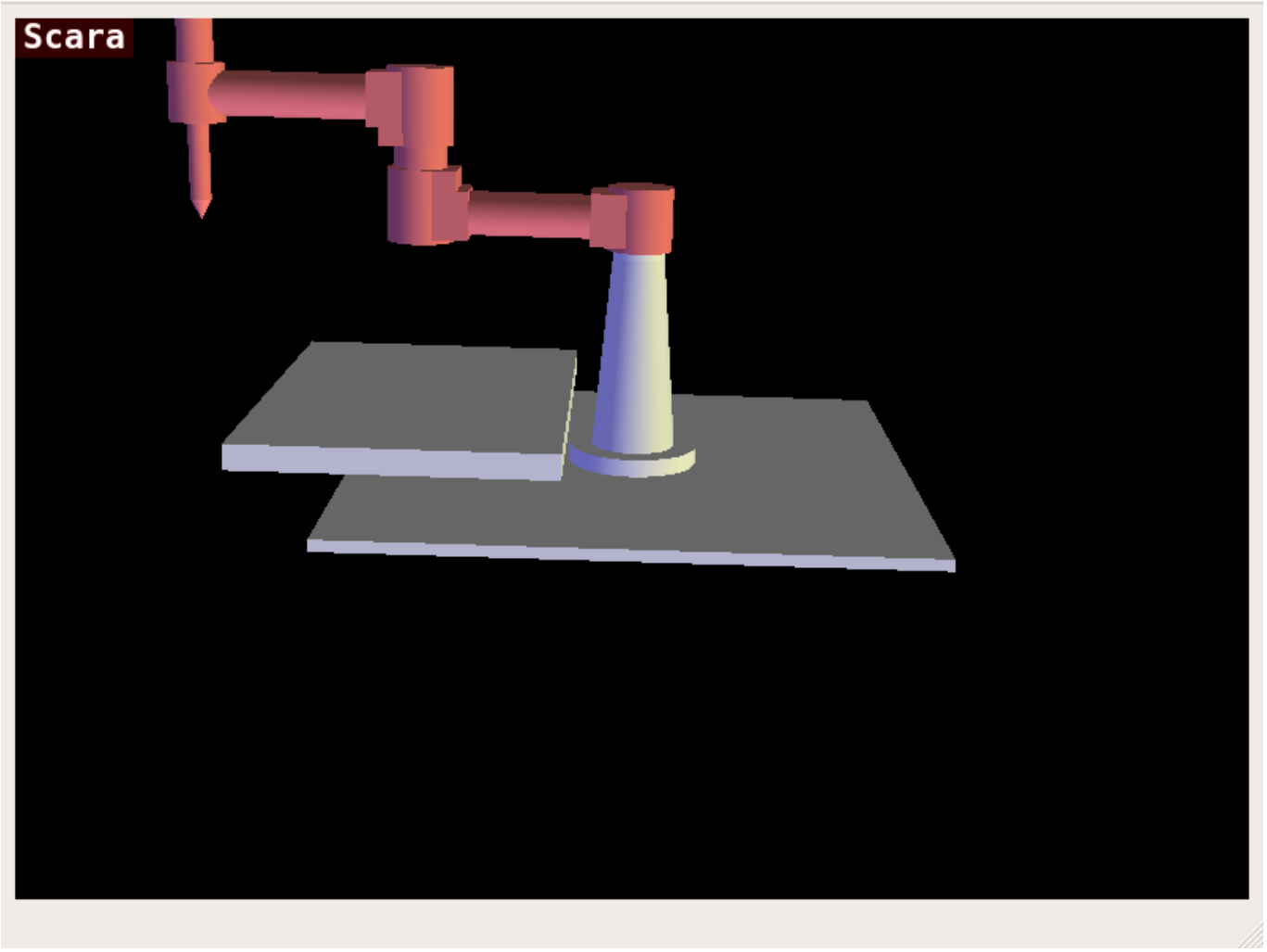


Figura 12.79: QtVCP `vismach_scara` - SCARA Mill 3D View Panel

12.6.2.4. QtVCP `vismach_millturn`

Vista 3D de OpenGL de una *máquina fresadora de 3 ejes con un eje/husillo A*.

```
loadusr qtvcp vismach_millturn
```

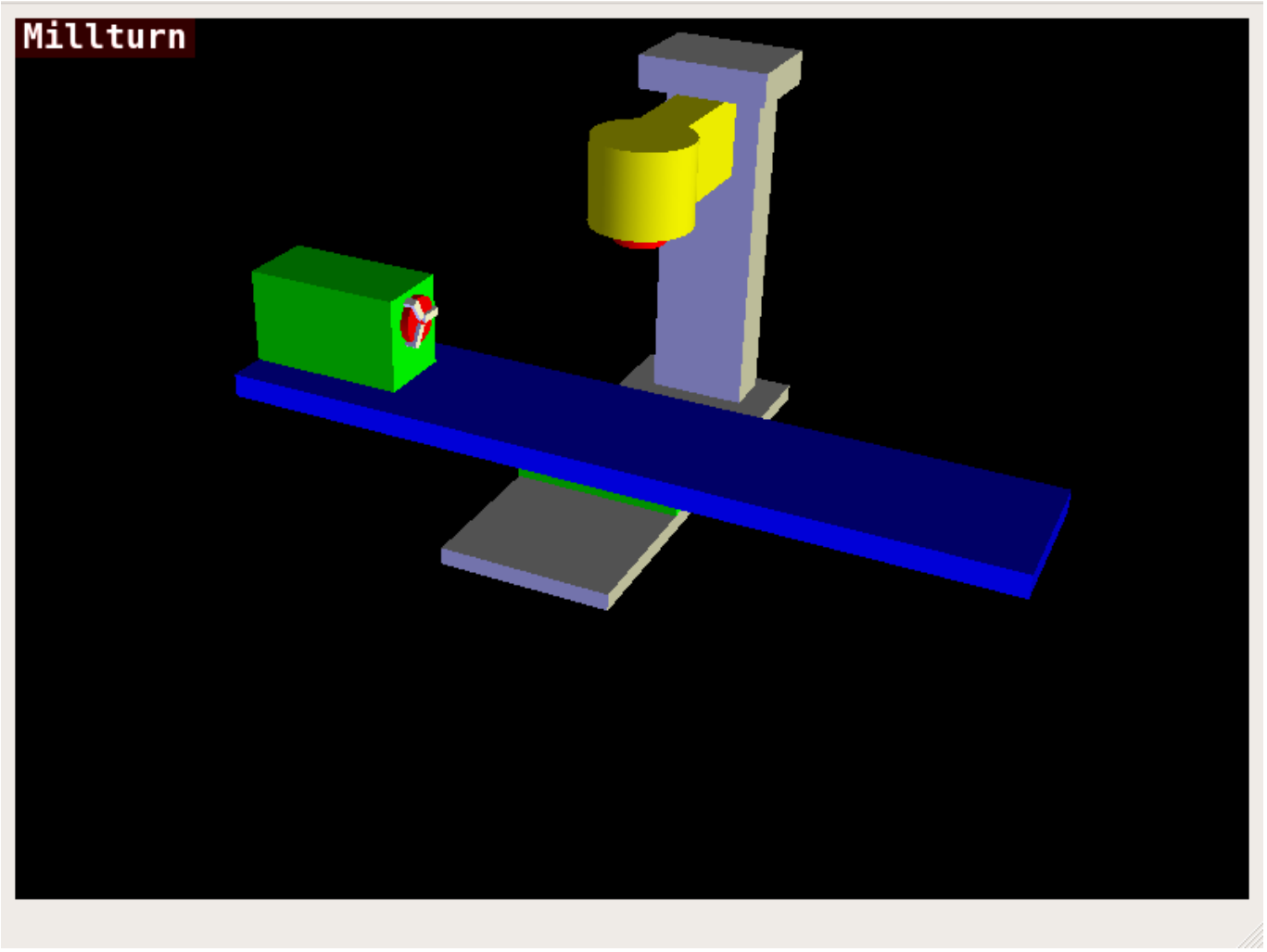


Figura 12.80: QtVCP `vismach_millturn` - 4 Axis MillTurn 3D View Panel

12.6.2.5. QtVCP `vismach_mill_5axis_gantry`

Vista 3D de OpenGL de una *máquina fresadora tipo gantry de 5 ejes*.

```
loadusr qtvcp vismach_mill_5axis_gantry
```

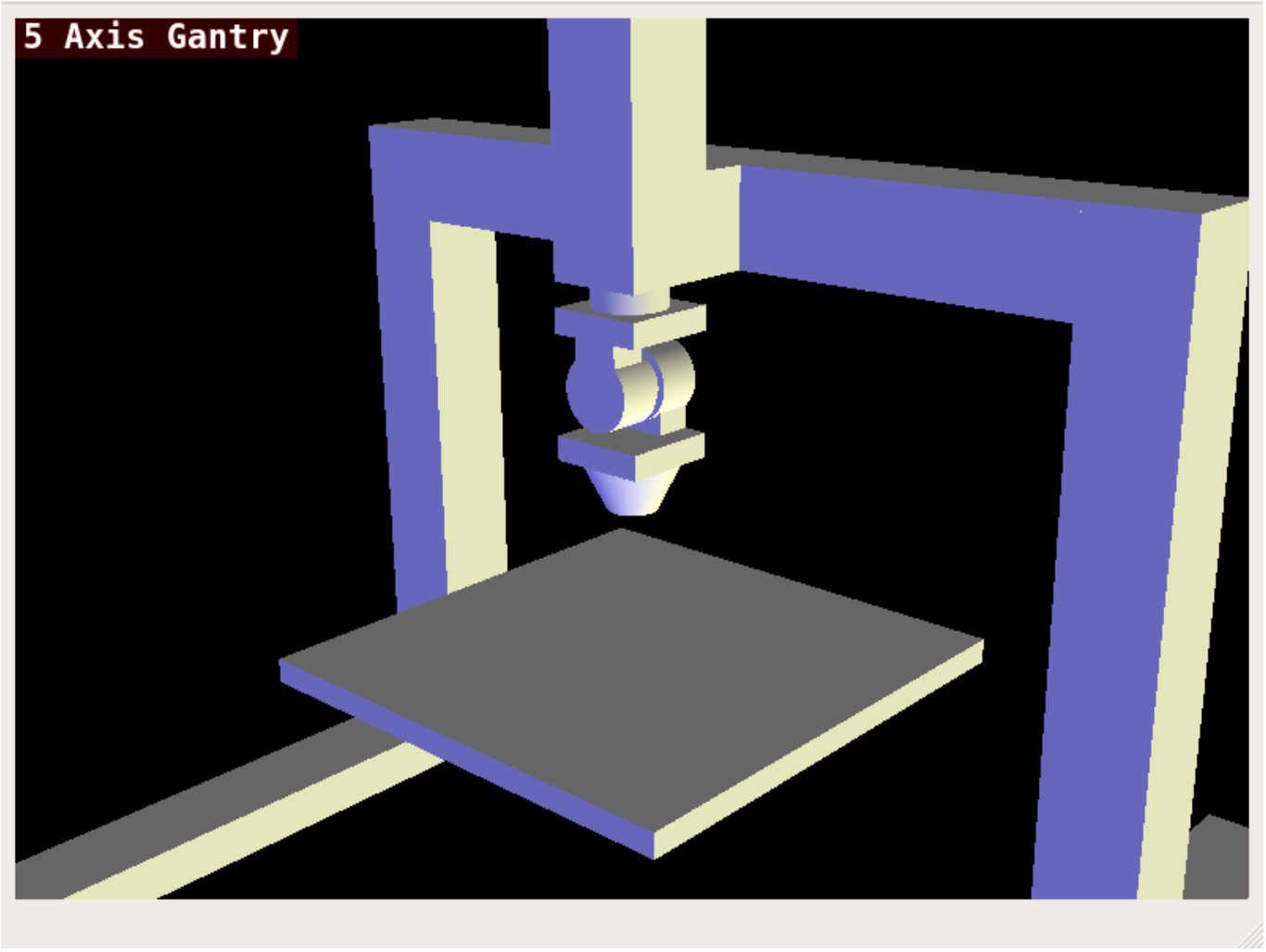


Figura 12.81: QtVCP `vismach_mill_5axis_gantry` - 5-AxIs Gantry Mill 3D View Panel

12.6.2.6. QtVCP `vismach_fanuc_200f`

3D openGL view of a *6 joint robotic arm*.

```
loadusr qtvcp vismach_fanuc_200f
```

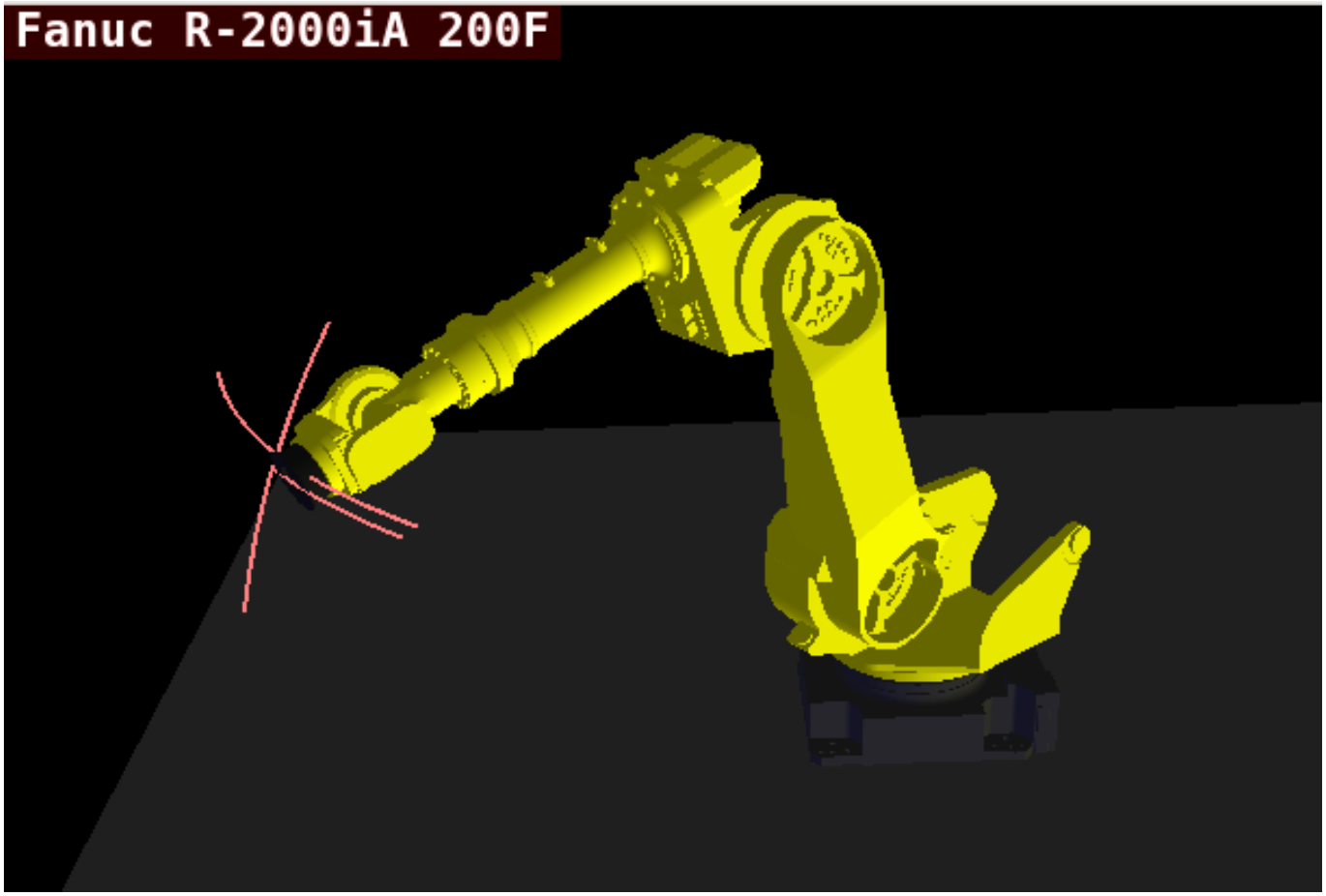


Figura 12.82: QtVCP `vismach_fanuc_200f` - 6 Joint Robotic Arm

12.6.3. Custom Virtual Control Panels

You can of course **make your own panel and load it**.

Si has creado un archivo UI nombrado `my_panel.ui` y un archivo HAL nombrado `my_panel.hal`, podrías cargarlos desde una terminal con:

```
halrun -I -f my_panel.hal
```

Example HAL file loading a QtVCP panel

```
# load realtime components
loadrt threads
loadrt classicladder_rt

# load non-realtime programs
loadusr classicladder
loadusr -Wn my_panel qtvcp my_panel.ui  # (1)

# add components to thread
addf classicladder.0.refresh thread1


# connect pins
net bit-input1     test_panel.checkbox_1        classicladder.0.in-00
```

```
net bit-hide       test_panel.checkbox_4        classicladder.0.hide_gui

net bit-output1    test_panel.led_1             classicladder.0.out-00

net s32-in1        test_panel.doublescale_1-s   classicladder.0.s32in-00

# start thread
start
```

(1), (1) En esta caso cargamos `qtvcp` usando **`-Wn`** el cual espera a que se termine de cargar el panel antes de ejecutar a continuación el siguiente comando HAL.
Esto es para *asegurar que los pines HAL del panel creado estén terminados* en caso de ser usados en lo que resta del archivo.

12.6.4. Embedding QtVCP Virtual Control Panels into QtVCP Screens

QtVCP panels can be embedded into most QtVCP screens and avoids problems such as focus transferring that can be a problem in non-native embedding.

12.6.4.1. Embedding Commands

A typical screen such as QtDragon will search the INI file under the heading [DISPLAY] for commands to embed a panel.

```
[DISPLAY]
EMBED_TAB_NAME=Embedding demo
EMBED_TAB_COMMAND=qtvcp simple_hal
EMBED_TAB_LOCATION=tabWidget_utilities
```

EMBED_TAB_NAME
: will typically be the title of the tab.

EMBED_TAB_LOCATION
: will be specific to the screen and specifies the tabWidget or stackedWidget to embed into.

EMBED_TAB_COMMAND
: is the command used to invoke loading of the panel. For native embedded panels the first word will always be *qtvcp*, the last will be the panel name to load. You can also pass options to the panel with -o switches in the command line between *qtvcp* and the panel name. The panel will follow the debugging mode setting of the main screen.

12.6.4.2. Location of builtin Panels

There are panels available that are included with LinuxCNC. To see a list open a terminal and type *qtvcp* and press return.
You will get a help printout and a list of builtin screen and panels.
Pick any of the names from the panel list and add that to the COMMAND entry after *qtvcp*.
The builtin panel search path is *share/qtvcp/panels/PANELNAME*.
Run-In-Place and installed versions of LinuxCNC have these in different locations on the system.

12.6.4.3. Location of Custom Panels

Custom panels can be embedded too -either a modified builtin panel or a new user-built one.
When loading panels, QtVCP looks in the configuration folders path for *qtvcp/panels/PANELNAME/-PANELNAME.ui*.
PANNELNAME being any valid string with no spaces. If no path is found there, then looks in the builtin file path.
QtVCP will do the same process for the optional handler file: *qtvcp/panels/PANELNAME/PANELNAME_handler.py*

12.6.4.4. Handler Programming Tips

In a screen handler file, the reference used for the window is *self.w*.
In QtVCP panels, that reference will refers to the panel's window.
To reference the main window use *self.w.MAIN*. If your panel is to be able to run independently and embedded, you must trap errors from referencing objects not available. (Note, main screen objects are not available in an independent panel.)

E.g., this would use the panel's preference file if there is one.

```
try:
    belt_en = self.w.PREFS_.getpref('Front_Belt_enabled', 1, int, 'SPINDLE_EXTRAS')
except:
    belt_en = 1
```

This would use the main screen preference file if there is one.

```
try:
    belt_en = self.w.MAIN.PREFS_.getpref('Front_Belt_enabled', 1, int, 'SPINDLE_EXTRAS')
except:
    belt_en = 1
```

12.6.4.5. Designer Widget Tips

When using Python command option in Action Button widgets of an embedded panel:

`INSTANCE`
: refers to the panel window. E.g., `INSTANCE.my_panel_handler_function_call(True)`

`MAIN_INSTANCE`
: refers to the main screen window. E.g., `MAIN_INSTANCE.my_main_screen_handler_function_call(Tru`

If the panel is not embedded, both refer to the panel window.

12.6.4.6. Handler Patching - Subclassing Builtin Panels

We can have QtVCP load a subclassed version of the standard handler file. In that file we can manipulate the original functions or add new ones.
Subclassing just means our handler file first loads the original handler file and adds our new code on top of it - effectively a patch of changes.
This is useful for changing/adding behaviour while still retaining standard handler updates from LinuxCNC repositories.

You may still need to use the handler copy dialog to copy the original handler file to decide how to patch it.

There should be a folder in the config folder; for panel: named *<CONFIG FOLDER>/qtvcp/panels/<PANEL NAME>/*
add the handle patch file there, named like so <ORIGINAL PANEL NAME>_handler.py,
i.e. for cam_align the file would be called *cam_align_handler.py*.

Here is a sample to change the circle color in cam_align:

```
import sys
import os
import importlib
from PyQt5.QtCore import Qt
from qtvcp.core import Path

PATH = Path()

# get reference to original handler file so we can subclass it
sys.path.insert(0, PATH.PANELDIR)
panel = os.path.splitext(os.path.basename(os.path.basename(__file__)))[0]
base = panel.replace('_handler','')
module = "{}.{}".format(base,panel)
mod = importlib.import_module(module, PATH.PANELDIR)
sys.path.remove(PATH.PANELDIR)
HandlerClass = mod.HandlerClass

# regresa nuestro objeto de subclase manejador a Qtvcp
def get_handlers(halcomp, widgets, paths):
    return [UserHandlerClass(halcomp, widgets, paths)]

# subclassed from HandlerClass which was imported above
class UserHandlerClass(HandlerClass):
    print('Custom subclassed panel handler loaded\n')

    def initialized__(self):
        # call original handler initialized function
        super().initialized__()

        # add our customization
        self.w.camview.circle_color = Qt.green
```

12.7. QtVCP Widgets

Qtscreen uses *QtVCP widgets* for LinuxCNC integration.

Widget is the general name for the *UI objects* such as buttons and labels in PyQt.

You are free to use any available **default widgets** in the *Qt Designer* editor.

There are also **special widgets** made for LinuxCNC that make integration easier. These are split in two, heading on the right side of the editor:

- Uno es para **widgets solo para HAL**.
- The other is for **CNC control widgets**.

You are free to mix them in any way on your panel.

nota
This description of widget properties can easily be out of date due to further development and lack of people to write docs (a good way to give back to the project). The definitive descriptions are found by looking in the source code.

12.7.1. HAL Only Widgets

These widgets usually have *HAL pins* and **don't react to the machine controller**.

12.7.1.1. `CheckBox` Widget

This widget allows the user to **check a box to set a HAL pin true or false**.

It is based on PyQt's *QCheckButton*.

12.7.1.2. *DetachTabWidget* - Container Widget With User Detachable Panels

This container widget works just like a QTabWidget -it displays multiple panels one at a time with tabs to select.
If you double click the tab or drag the tab, the tab will detach from the main window.
When a tab is detached, the contents are placed into a QDialog.
The tab can be re-attached by closing the dialog or by double clicking on its window frame.

It is based on PyQt's QTabWidget.

12.7.1.3. `DoubleScale` - Spin Button Entry Widget

This widget is a **spin button entry** widget used for *setting a s32 and float HAL pin*.

It has an internal *scale factor*, set to a default of 1, that can be set programmatically or using a QtSignal.

The `setInput` slot can be connected to an integer, or a float signal.

`[HALLabelName].setInput(some_value)`
: This is a function call to change the internal scaling factor.

The HAL pins will be set to the value of the *internal scale times the widget displayed value*.

12.7.1.4. `FocusOverlay` - Focus Overlay Widget

This widget places a **colored overlay over the screen**, usually while a dialog is showing.

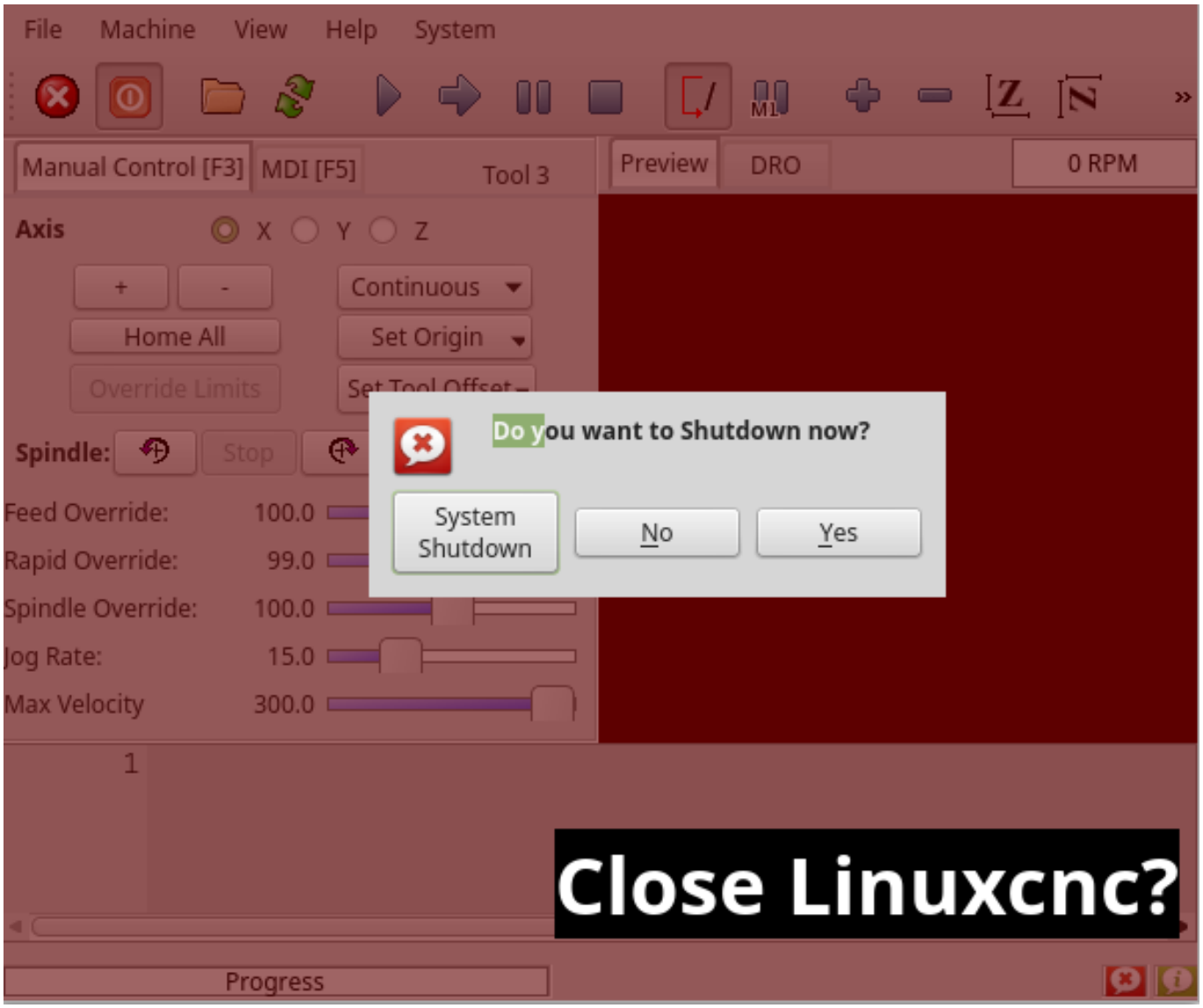


Figura 12.83: Focus overlay example for confirm close prompt

Used to create a *focused* feel and to draw attention to critical information.

It can also show a translucent image.
It can also display message text and buttons.

This widget *can be controlled with STATUS messages*.

12.7.1.5. `Gauge` - Round Dial Gauge Widget

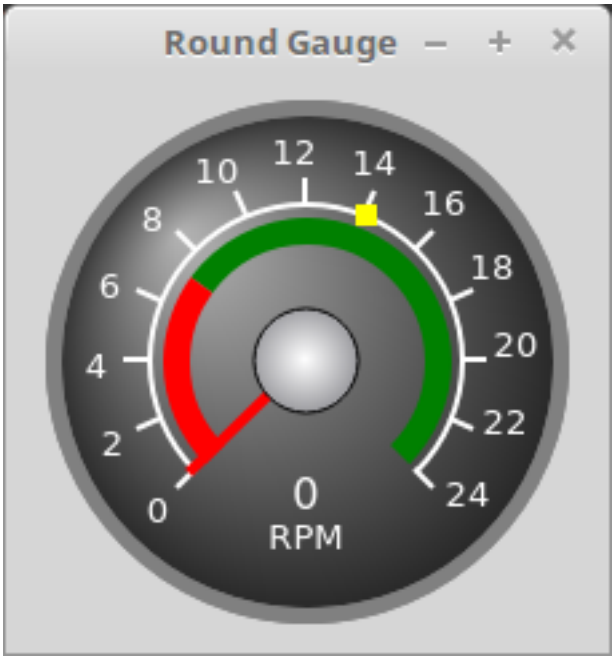


Figura 12.84: QtVCP `Gauge`: Round Dial Gauge Widget

Round Gauge can be used in a LinuxCNC GUI to **display an input parameter** on the dial face.

Customizable Parameters There are several properties that are user settable in order to customize the *appearance of the gauge*.

The following parameters can be set either programmatically or via the Qt Designer property editor.

`halpin_option`
Setting this to `True` will *create 2 HAL pins*:

- One is for setting the `value` input
- The other is for setting the `setpoint`.

If this option is not set, then `value` and `setpoint` must be connected programmatically, i.e., in the handler file.

`max_reading`
This value determines the *highest number displayed* on the gauge face.

`max_value`
This is the *maximum expected value of the value input signal*.
In other words, it is the full scale input.

`num_ticks`
This is the *number of ticks/gauge readings* on the gauge face.
It should be set to a number that ensures the text readings around the gauge face are readable.
The minimum allowed value is 2.

`zone1_color`
Zone1 extends *from the maximum reading to the threshold point*.
It can be set to any RGB color.

`zone2_color`
Zone2 extends *from the threshold point to the minimum reading,* which is 0.
It can be set to any RGB color.

`bezel_color`
This is the color of the *outer ring of the gauge*.

`bezel_width`
This is the width of the *outer ring of the gauge*.

`threshold`
The threshold is the *transition point between the zones*.
It should be set to a value between 0 and the maximum value.
The maximum allowed value is set to the gauge's `max_value` and minimum value is 0.

`gauge_label`
This is the *text below the value readout,* near the bottom of the gauge.
The function of the gauge is then easily visible.

`base_color`
El color del indicador.

`base_gradient_color`
The highlight color of the gauge.

`center_color`
The color of the center of the gauge.

`center_gradient_color`
The highlight color of the center of the gauge.

Parámetros no personalizables There are 2 inputs that are not customizable. They can be set via HAL pins, programmatically or via signals from other widgets:

`value`
This is the *actual input value* that will be displayed with the gauge needle and in the digital readout.
It must be set to a value between 0 and `max_value` maximum value.

`setpoint`
This is a value that determines the location of a small *marker on the gauge face*. It must be set to a value between 0 and the maximum value.

12.7.1.6. `GeneralHALInput` - General Signals/Slots Input Connection Widget

This widget is used to **connect an arbitrary Qt widget to HAL using signals/slots**.

It is used *for widgets that should **respond** to HAL pin changes*.

12.7.1.7. `GeneralHALOutput` - General Signals/Slots Output Connection Widget

This widget is used to **connect an arbitrary Qt widget to HAL using signals/slots**.

It is used *for widgets that should **control** HAL pins*.

12.7.1.8. `GridLayout` - Grid Layout Widget

This widget **controls if the widgets inside it are enabled or disabled**.
Disabled widgets typically have a different color and do not respond to actions.
It is based on PyQt's `QGridLayout`.

12.7.1.9. *HalBar* - HAL Bar Level Indicator

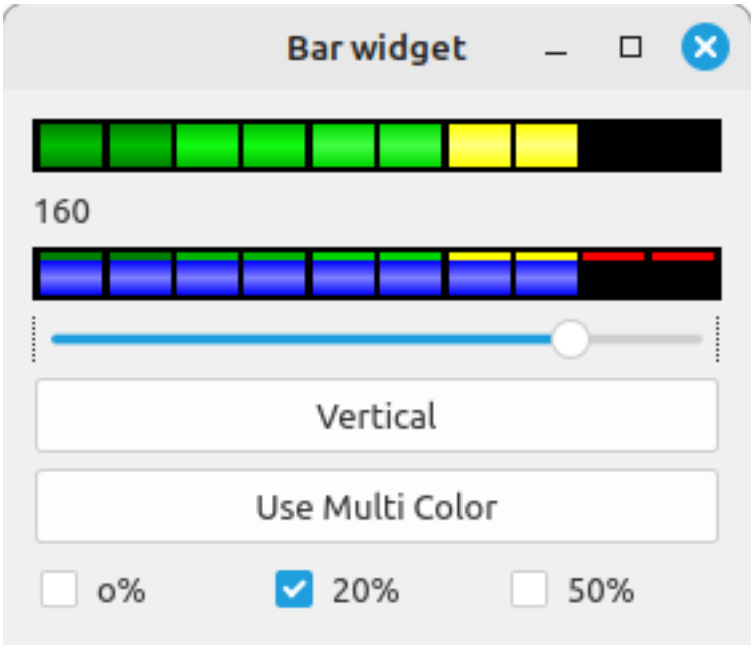


Figura 12.85: QtVCP `HalBar`: Panel demonstrating the HAL Bar Level Indicator

This widget is used to indicate level or value, usually of a HAL s32/float pin.
You can also disable the HAL pin and use Qt signals or Python commands to change the level.

HalBar is a subclass of the Bar widget, so it inherits these properties:

- *stepColorList*: a list of color strings, the number of colors defines the number of bars.
- *backgroundColor*: a QColor definition of the background color.
- *indicatorColor*: a QColor definition of the optional single color current value bar.
- *useMultiColorIndicator*: bool switch for choosing the option of single or multicolor value bar.
- *split*: the integer percentage split of max value bar versus current value bar (0 to 50 %).
- *setVertical*: bool switch for choosing vertical or horizontal indicator.
- *setInverted*: bool switch for choosing inverted direction.
- *setMaximum*: an integer that defines the maximum level of indication.
- *setMinimum*: an integer that defines the lowest level of indication.

- *pinType*: to select **HAL pins type**:
  - `NONE` no HAL pin will be added
  - `S32` A S32 integer pin will be added
  - `FLOAT` A Float pin will be added
- *pinName*: to change the **HAL pin name** otherwise the widget base name is used.

The above Bar properties could be set in *styles sheets*.
pinType and pinName properties can not be changed in stylesheets.

nota
In style sheets, stepColorList is a single string of color names separated by commas.

```
HalBar{
    qproperty-backgroundColor: #000;
    qproperty-stepColorList: 'green,green,#00b600,#00b600,#00d600,#00d600,yellow,yellow,red ↩
        ,red';
}
```

12.7.1.10. `HALPad` - HAL Buttons Joypad

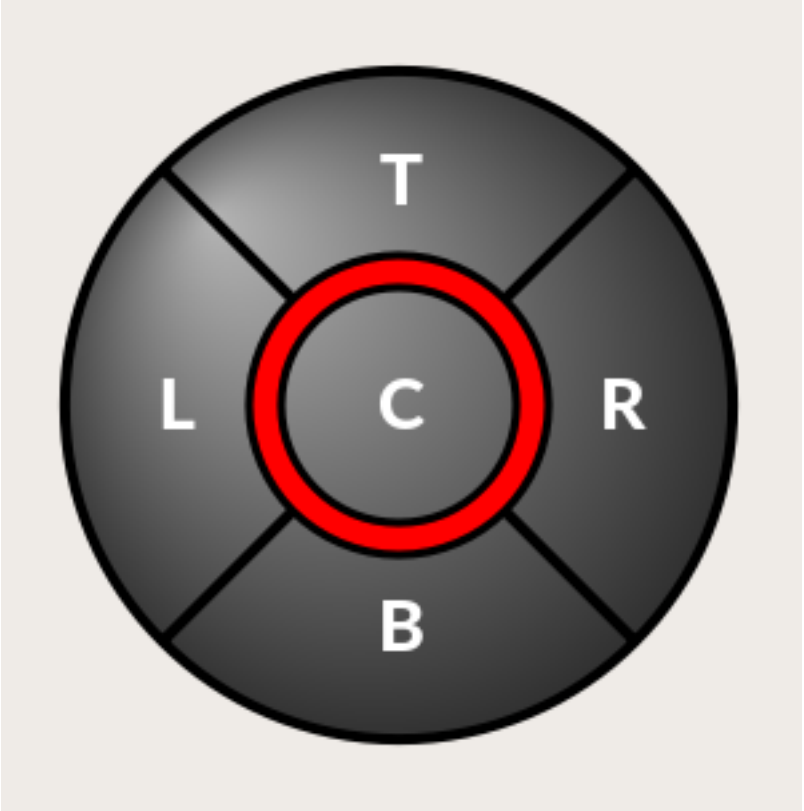


Figura 12.86: QtVCP `HALPad`: HAL Buttons Joypad

This widget looks and acts like a **5 buttons D-pad**, with an LED ring.

Each button has an selectable type (Bit, S32 or Float) output HAL pin.

The LED center ring has selectable colors for off and on and is controlled by a bit HAL pin.

HALPad ENUMS There are *enumerated constants* used:

- To reference **indicator positions**:
  - NONE
  - LEFT
  - RIGHT
  - CENTER
  - TOP
  - BOTTOM
  - LEFTRIGHT
  - TOPBOTTOM
- For **HAL pins type**:
  - NONE
  - BIT
  - S32
  - FLOAT

You use the widget name in Qt Designer plus the reference constant:

```
self.w.halpadname.set_highlight(self.w.halpadname.LEFTRIGHT)
```

HALPad Properties

pin_name
Optional name to use for the *HAL pins basename*. If left blank, the Qt Designer widget name will be used.

pin_type
Select the *HAL output pin type*. This property is only used at startup. Selection can be set in Qt Designer:

- NONE
- BIT
- S32
- FLOAT

left_image_path , right_image_path , center_image_path , top_image_path , bottom_image_path

File or resource path to an image to display in the described button location.
If the reset button is pressed in the Qt Designer editor property, the image will not be displayed (allowing optional text).

left_text , right_text , center_text , top_text , bottom_text
A text string to be displayed in the described button location.
If left blank an image can be designated to be displayed.

`true_color` , `false_color`
: Color selection for the center LED ring to be displayed when the `<BASENAME>.light.center` HAL pin is `True` or `False`.

`text_color`
: Color selection for the button text.

`text_font`
: Font selection for the button text.

HALPad Styles The above properties could be set in *styles sheets*.

```
HALPad{
    qproperty-on_color: #000;
    qproperty-off_color: #444;
}
```

12.7.1.11. `HALLabel` - HAL Label Widget

This widget **displays values sent to it**.

Values can be sent from:

- *HAL pins*
  The input pin can be selected as Bit, S32, Float or no pin selected
- *Programmatically*
- *A `QtSignal`*

There is a `textTemplate` property to set the rich text and/or to format the text.
Basic formatting might be:

- `%r` for booleans
- `%d` for integers
- `%0.4f` for floats.

A rich text example might be:

```
self.w.my_hal_label.setProperty(textTemplate,"""
<html>
<head/>
<body>
  <p><span style="font-size:12pt;font-weight:600;color:#f40c11;">%0.4f</span></p>
</body>
</html>
"""
)
```

The `setDisplay` slot can be connected to an integer, a float or a bool signal.

If the property `pin_name` is not set the widget name will be used.

There are function calls to display values:

`[HALLabelName].setDisplay(some_value)`
: Can be used to set the display if no HAL pin is selected.

`[HALLabelName].setProperty(textTemplate,"%d")`
: Sets the template of the display.

It is based on PyQt's *QLabel*.

12.7.1.12. `LCDNumber` - LCD Style Number Readout Widget

This widget *displays HAL float/s32/bit values in a LCD looking way*.

It can display numbers in decimal, hexadecimal, binary and octal formats by setting the **`mode`** property.

When using floats you can set a formatting string.

You must set the **`digitCount`** property to an appropriate setting to display the largest number.

Properties

`pin_name`
: Option string to be used as the HAL pin name.
If set to an empty string the widget name will be used.

`bit_pin_type`
: Selects the input pin as type BIT.

`s32_pin_type`
: Selects the input pin as type S32.

`float_pin_type`
: Select the input pin as type `FLOAT`.

`floatTemplate`
: A string that will be used as a Python3 format template to tailor the LCD display.
Only used when a `FLOAT` pin is selected, e.g., `{:.2f}` will display a float rounded to 2 numbers after the decimal.
A blank setting will allow the decimal to move as required.

It is based on PyQt's *QLCDNumber*.

12.7.1.13. `LED` - Indicator Widget

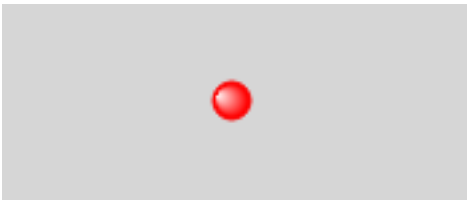


Figura 12.87: QtVCP `LED`: LED Indicator Widget

A **LED like indicator** that optionally follows a HAL pin's logic.

`halpin_option`
: Selects if the LED follows an input HAL pin or program state.

`diameter`
: Diámetro del LED (predeterminado a 15).

`color`
: Color del LED cuando esta encendido (predeterminado a verde).

`off_color`
: Color del LED cuando esta apagado (predeterminado a negro).

gradient
: turns the gradient high light on or off (defaults to on).

on_gradient_color
: Color highlight of the LED when on (defaults to white).

off_gradient_color
: Color highlight of the LED when off (defaults to white).

alignment
: Sugerencia de alineación de Qt.

state
: Current state of LED

flashing
: Enciende y apaga la opción de parpadeo.

flashRate
: Sets the flash rate.

The `LED` properties can be defined in a *stylesheet* with the following code added to the `.qss` file, `name_of_led` being the widget name defined in Qt Designer's editor:

```
LED #name_0f_led{
  qproperty-color: red;
  qproperty-diameter: 20;
  qproperty-flashRate: 150;
}
```

12.7.1.14. `PushButton` - HAL Pin Toggle Widget

This widget allows a user to **set a HAL pin true or false** with the push of a button.

As an option it can be a *toggle button*.

For a *LED Indicator Option,* see Sección 12.7.5.1[IndicatedPushButton] below for more info.

It also has other options.

It is based on PyQt's *QPushButton*.

12.7.1.15. `RadioButton` Widget

This widget allows a user to **set HAL pins true or false**. Only one `RadioButton` widget of a group can be true at a time.

It is based on PyQt's *QRadioButton*.

12.7.1.16. `Slider` - HAL Pin Value Adjusting Widget

Allows one to **adjust a HAL pin value using a sliding pointer**.

12.7.1.17. `TabWidget` - Tab Widget

This widget allows the tab height to be adjusted with stylesheets.

The `TabWidget` properties can be defined in a *stylesheet* with the following code added to the `.qss` file.
`name_of_tab` being the widget name defined in Qt Designer's editor.
If you omit the *#name_of_tab* text, all TabWidgets tab height will be set.
This shows how to set a particular widget's tab height:

```
TabWidget #name_of_tab{
  qproperty-tabsize: 1.5;
}
```

It is based on PyQt's QTabWidget.

12.7.1.18. `WidgetSwitcher` - Multi-widget Layout View Switcher Widget

This is used to switch the view of a multi-widget layout to show just one widget, i.e. to **flip between a large view of a widget and a smaller multi widget view**.

It is *different from a stacked widget* as it can pull a widget from anywhere in the screen and place it in its page with a different layout than it originally had.

The *original widget must be in a layout* for switcher to put it back.

En Qt Designer:

- Agrega el widget `WidgetSwitcher` a la pantalla.
- Right click the `WidgetSwitcher` and add a page.
- Populate it with the widgets/layouts you wish to see in a default form.
- Add as many pages as there are views to switch to.
- On each page, add a layout widget.
  After adding the layout you must right click the widget switcher again and set the layout option.
- Click on the `WidgetSwitcher` widget and then scroll to the bottom of the property editor.
- Look for the dynamic property `widget_list` and double click to the right of it.
- A dialog pops up allowing you to add the names of the widgets to move to the pages you added to the `WidgetSwitcher`.

There are *function calls* to display specific widgets.
By calling one of these functions, you control what widget is currently displayed:

`[_WidgetSwitcherName_].show_id_widget(_number_)`, `[_WidgetSwitcherName_].show_named_widget(`

This shows the `page 0` layout, and puts all other widgets back to where they were as initially built in Qt Designer.

`[_WidgetSwitcherName_].show_next()`
Muestra el widget siguiente.

It is based on the *QStack* widget.

12.7.1.19. `XEmbed` - Program Embedding Widget

Allows one to **embed a program into the widget**.

Only programs that utilize the `xembed` protocol will work such as:

- GladeVCP virtual control panels
- Onboard virtual keyboard
- QtVCP virtual control panels
- mplayer video player

12.7.2. Machine Controller Widgets

These widgets **interact with the Machine Controller state**.

12.7.2.1. `ActionButton` - Machine Controller Action Control Widget

These buttons are used for **control actions on the machine controller**.

They are built on top of `IndicatedPushButton` so can have LEDs overlaid.

---

nota
If you left double click on this widget you can launch a dialog to set any of these actions. The dialogs will help to set the right related data to the selected action. You can also change these properties directly in the property editor.

Actions You can select one of these:

`Estop` , `Machine On` , `Auto` , `mdi` , `manual` , `run` , `run_from_line status`
: Gets line number from `STATUS` message `gcode-line-selected`.

`run_from_line slot`
: Gets line number from Qt Designer int/str slot `setRunFromLine`.

`abort` , `pause` , `load dialog`
: Requires a dialog widget present.

`Camview dialog`
: Requires `camview` dialog widget present.

`origin offset dialog`
: Requires origin offset dialog widget present.

`macro dialog`
: Requires macro dialog widget present.

`Launch Halmeter` , `Launch Status` , `Launch Halshow` , `Home`
: Set the joint number to -1 for `all-home`.

`Unhome`
: Set the joint number to -1 for `all-unhome`.

`Home Selected`
: Homes the joint/axis selected by `STATUS`.

Unhome Selected
Unhomes the joint/axis selected by STATUS.

zero axis , zero G5X
Poner a ceros los desplazamientos del sistema de coordenadas de usuario actual.

zero G92
Puesta cero de los desplazamientos G92 opcionales.

zero Z rotational
Zeros the rotation offset.

jog joint positive
Establece el número de articulación.

jog joint negative
Establece el número de articulación.

jog selected positive
Selected with a different widget or STATUS.

jog selected negative
Selected with a different widget or STATUS.

jog increment
Set metric/imperial/angular numbers.

jog rate
Establece el número flotante float/alt.

feed override
Establece el número flotante float/alt.

rapid override
Establece el número flotante float/alt.

spindle override
Establece el número flotante float/alt.

spindle fwd , spindle backward , spindle stop , spindle up , spindle down , view change
Set view_type_string.

limits override , flood , mist , block delete , optional stop , mdi command
Set command_string, i.e.,calls a hard coded MDI command

INI mdi number
Set ini_mdi_number, i.e., calls an INI based MDI command

dro absolute , dro relative , dro dtg , exit screen
Closes down LinuxCNC

Override limits
Temporarily override hard limits

launch dialogs
Pops up dialogs if they are included in ui file.

set DRO to relative , set DRO to absolute , set DRO to distance-to-go

Attributes These set *attributes* of the selected action (availability depends on the widget):

toggle float option
Allows jog rate and overrides to toggle between two rates.

joint number
Selects the joint/axis that the button controls.

incr imperial number
Sets the imperial jog increment (set negative to ignore).

incr mm number
Sets the metric jog increment (set negative to ignore).

incr angular number
Sets the angular jog increment (set negative to ignore).

float number
Used for `jograte` and overrides.

float alternate number
For `jograte` and overrides that can toggle between two float numbers.

cadena de tipo de vista
Can be:

- `p`,
- `x`, `y`, `y2`, `z`, `z2`,
- `zoom-in`, `zoom-out`,
- `pan-up`, `pan-down`, `pan-left`, `pan-right`,
- `rotate-up`, `rotate-down`, `rotate-cw`, `rotate-ccw`
- `clear`.

command string
MDI command string that will be invoked if the MDI command action is selected.

ini_mdi_number
(Legacy way)
A reference to the *INI file* `[MDI_COMMAND_LIST]` section.
Set an integer of select one line under the INI's `[MDI_COMMAND]` line starting at 0.
Then in the INI file, under the heading `[MDI_COMMAND_LIST]` add appropriate lines.
Commands separated by the `;` will be run one after another
The button label text can be set with any text after a comma, the `\n` symbol adds a line break.

ini_mdi_key
(preferred way)
A reference to the *INI file* `[MDI_COMMAND_LIST]` section.
This string will be added to *MDI_COMMAND_* to form an entry to look for
in the INI file, under the heading `[MDI_COMMAND_LIST]`.
Commands separated by the `;` will be run one after another
The button label text can be set with any text after a comma, the `\n` symbol adds a line break.

```
[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
```

Action buttons are subclassed from Sección 12.7.5.1[`IndicatedPushButton`]. See the following sections for more information about:

- LED Indicator option

- Enabled on State
- Text Changes On State
- Call Python Command On State

12.7.2.2. `ActionToolButton` - Optional Actions Menu Button Widget

`ActionToolButton` buttons are similar in concept to action buttons, but they use *QToolButtons* to allow for **optional actions** to be selected by pushing and holding the button till the option menu pops up. Currently there is only one option: `userView`.

It is based on PyQt's *QToolButton*. **userView Record and Set User View Widget**

User View tool button allows to **record and return to an arbitrary graphics view**.

Press and hold the button to have the menu pop up and press *record view* to record the currently displayed graphics view.

Click the button normally to return to the last recorded position.

The recorded position will be remembered at shutdown if a preference file option is set up.

---

nota
Due to programming limitations, the recorded position may not show exactly the same. Particularly, if you pan zoomed out and pan zoomed in again while setting the desired view.
Best practice is to select a main view, modify as desired, record, then immediately click the button to switch to the recorded position. If it is not as you like, modify its existing position and re-record.

12.7.2.3. `AxisToolButton` - Select and Set Axis Widget

This allows one to **select and set an axis**.

If the button is set checkable, it will indicate which axis is selected.

If you press and hold the button a pop up menu will show allowing one to:

- Zero the axis
- Divide the axis by 2
- Asigna el eje arbitrariamente
- Reset the axis to the last number recorded

You must have selected an entry dialog widget that corresponds to the dialog_code_string, usually this is selected from the screenOptions widget.

`halpin_option`
: Pondrá el pin HAL en verdadero cuando se seleccione el eje.

`joint_number`
: Debería asignársele el número de articulación apropiado.

`axis_letter`
: Debería asignársele la letra de eje apropiada.

These are the click-and-hold menu properties:

showLast
: Muestra la acción *Set to last*.

showDivide
: Show the *Divide by 2* action.

showGotoOrigin
: Show the *Go to G53/G5x origin* action.

showZeroOrigin
: Show the *Zero Origin* action.

showSetOrigin
: Show the *Set Origin* action.

dialog_code_string
: Sets which dialog will pop up with numerical entry, i.e. *ENTRY* or *CALCULATOR* to call a typing only entry dialog or a touch/typing calculator type entry dialog.

Here is a sample stylesheet entry:

```
AxisToolButton {
    /* Modify all the menu options */
    qproperty-showLast: false;
    qproperty-showDivide : true;
    qproperty-showGotoOrigin: true;
    qproperty-showZeroOrigin: true;
    qproperty-showSetOrigin: false;
    qproperty-dialog_code_string: CALCULATOR;
}
```

It is based on PyQt's *QToolButton*.

12.7.2.4. `BasicProbe` - Simple Mill Probing Widget

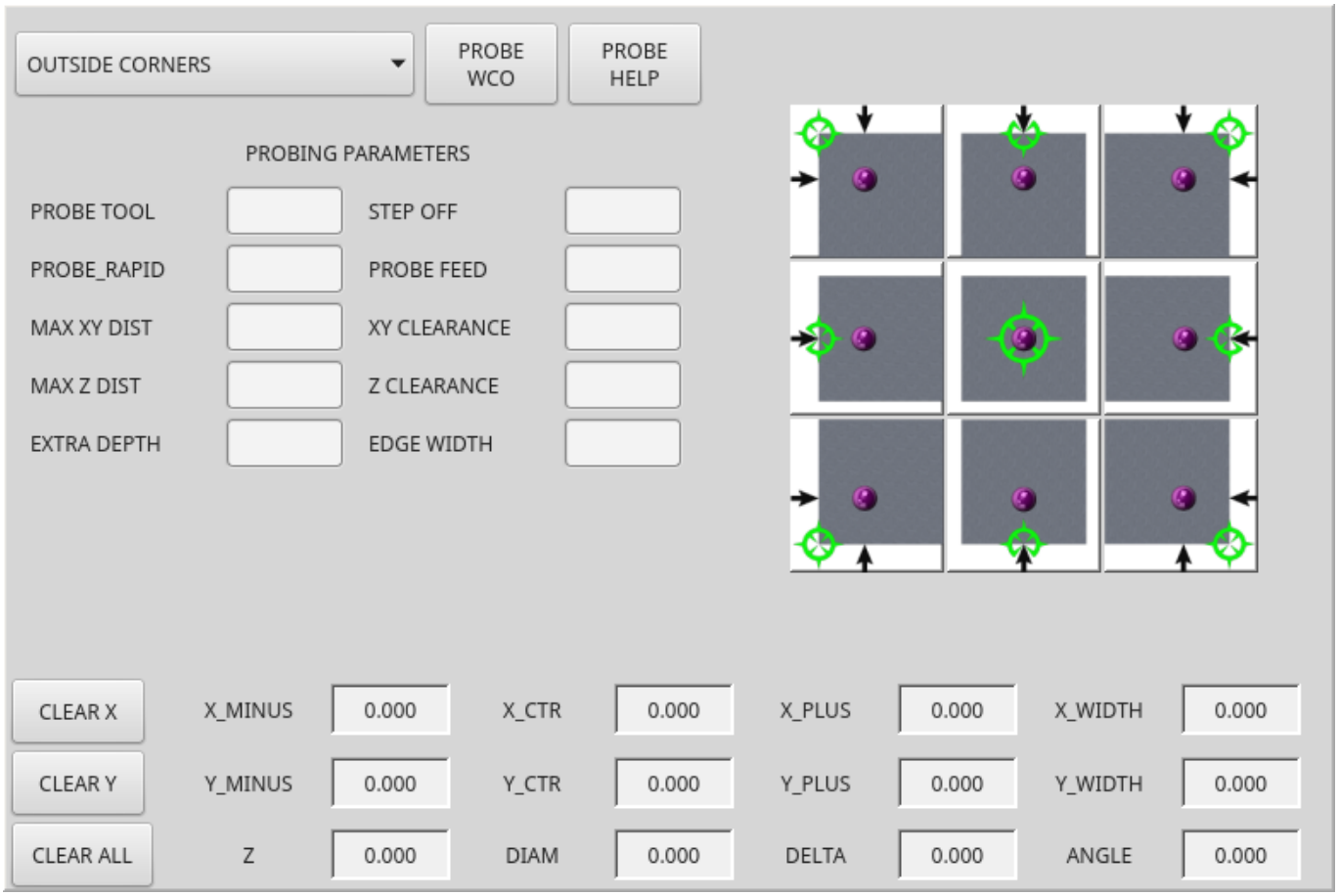


Figura 12.88: QtVCP `BasicProbe`: Simple Mill Probing Widget

Widget for **probing on a mill**. Used by the *QtDragon* screen.

12.7.2.5. `CamView` - Workpiece Alignment and Origin Setting Widget

This widget **displays a image from a web camera**.

It *overlays an adjustable circular and cross hair target* over the image.

CamView was built with precise visual positioning in mind.

This is used to **align the work piece or zero part features using a webcam**.

It uses *OpenCV* vision library.

12.7.2.6. `DROLabel` - Axis Position Display Widget

Esto **mostrará la posición actual de un eje**.
También puede hacer clic en la etiqueta y ver una lista de acciones.

Qjoint_number
Joint index number (X=0 Y=1) of offset to display (10 will specify rotational offset).

Qreference_type
Actual, relative or distance to go (0,1,2).

metric_template
Formato de visualización, p. ej. `%10.3f`.

imperial_template
Formato de visualización, p. ej. `%9.4f`.

angular_template
Forma de visualización, p. ej. `%Rotacional: 10.1f`.

always_display_diameter
Toggles display option

always_display_radius
Toggles display option

display_as_per_m7m8
Toggles display option. Will follow the current M7/8 mode.

follow_reference_changes
Toggles display option. Will follow the STATUS message reference mode, i.e. you can use Action buttons to set how it is currently displayed.

These are the click-on-menu options:

showLast
Muestra la acción *Set to last*.

showDivide
Show the *Divide by 2* action.

showGotoOrigin
Show the *Go to G53/G5x origin* action.

showZeroOrigin
Show the *Zero Origin* action.

showSetOrigin
Show the *Set Origin* action.

dialogName
Sets which dialog window will pop up with numerical entry, i.e. ENTRY or CALCULATOR.

The `DROLabel` widget holds a property **`isHomed`** that can be used with a stylesheet to change the *color of the `DRO_Label` based on homing state of the joint* number in LinuxCNC.

Here is a sample stylesheet entry that:

- Sets the font of all `DRO_Label` widgets,
- Sets the text template (to set resolution) of the DRO,
- Then sets the text color based on the Qt `isHomed` property.
- muestra todas la opciones del menú.

```
DROLabel {
    font: 25pt ”Lato Heavy”;
    qproperty-imperial_template: ’%9.4f’;
    qproperty-metric_template: ’%10.3f’;
    qproperty-angular_template: ’%11.2f’;

    /* Modify all the menu options */
    qproperty-showLast: true;
    qproperty-showDivide : true;
    qproperty-showGotoOrigin: true;
    qproperty-showZeroOrigin: true;
    qproperty-showSetOrigin: true;
    qproperty-dialogName: CALCULATOR;
}

DROLabel[isHomed=false] {
    color: red;
}

DROLabel[isHomed=true] {
    color: green;
}
```

Here is how you specify a particular widget by its `objectName` in Qt Designer:

```
DROLabel #dr0_x_axis [isHomed=false] {
    color: yellow;
}
```

It is based on PyQt’s *QLabel*.

12.7.2.7. `FileManager` - File Loading Selector Widget

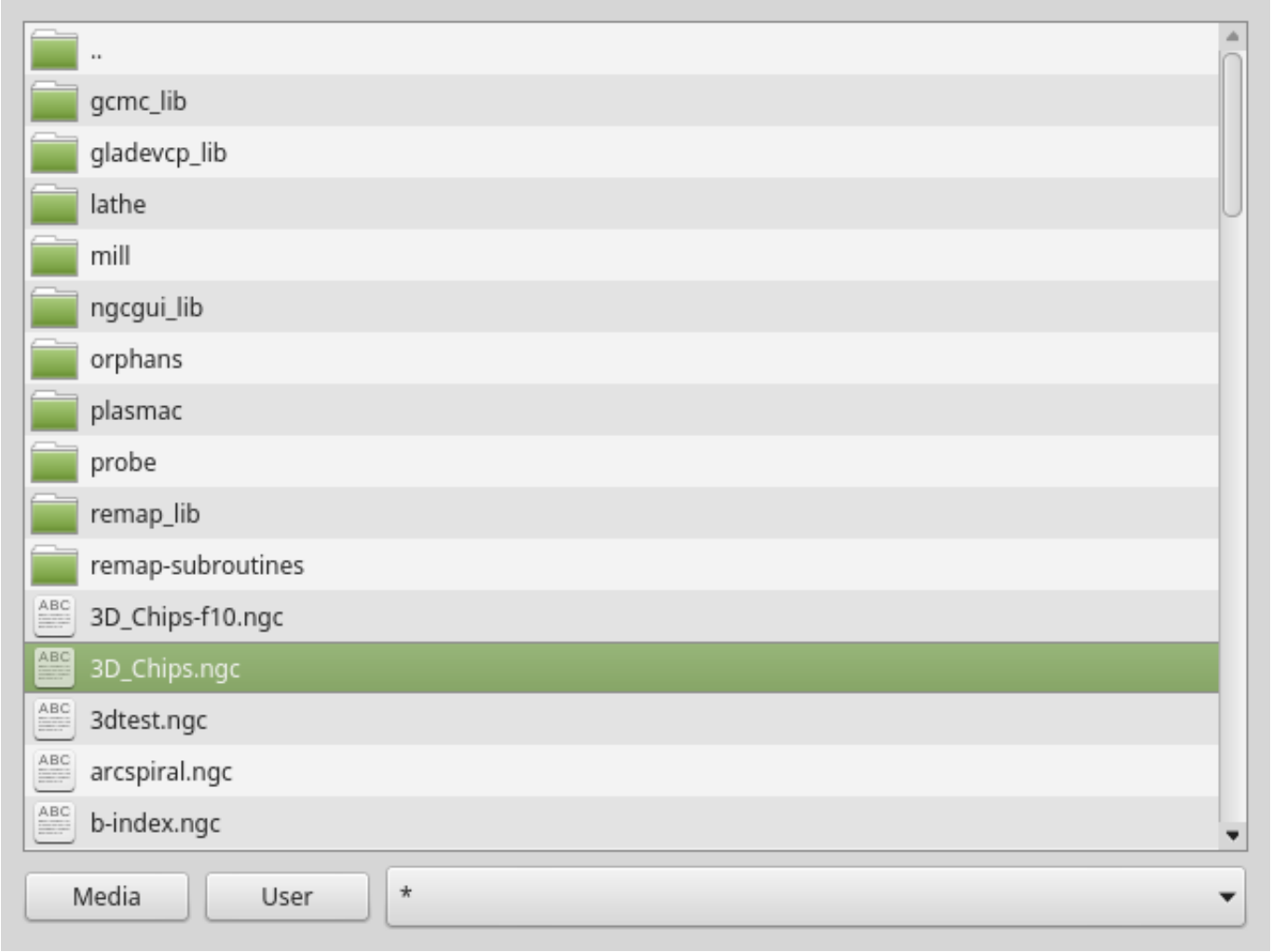


Figura 12.89: QtVCP `FileManager`: File Loading Selector Widget

This widget is used to **select files to load**.

It has a the ability to scroll the names with hardware such as a MPG.

One can class patch the function `load(self,fname)` to customize file loading.

The function `getCurrentSelected()` will return a Python tuple, containing the file path and whether it is a file.

```
temp = FILEMANAGER.getCurrentSelected()
print('filepath={}'.format(temp[0]))
if temp[1]:
    print('Is a file')
```

Stylesheets Properties

`doubleClickSelection` *(bool)*
Determines whether or not to *require double clicking on a folder*.
Single clicking a folder (False) is enabled by default and is intended for touch screen users.
The following shows an example of how to set this property:

```
#filemanager {
    qproperty-doubleClickSelection: True;
}
```

showListView *(bool)*
Determines whether or not to *show the file/folder structure in list form*.
Table view (False) is enabled by default.
The following shows an example of how to set this property:

```
#filemanager {
    qproperty-showListView: True;
}
```

It is based on PyQt's FIXME

12.7.2.8. `GcodeDisplay` - G-code Text Display Widget

This **displays G-code in text form**, highlighting the currently running line.

This can also display:

- **MDI history** when LinuxCNC is in `MDI` mode.
- **Log entries** when LinuxCNC is in `MANUAL` mode.
- **Preference file entries** if you enter `PREFERENCE` in capitals into the `MDILine` widget.

It has a *signal* **`percentDone(int)`** that can be connected to a slot (such as a `progressBar` to display percent run).

`auto_show_mdi_status`
Set true to have the widget switch to MDI history when in MDI mode.

`auto_show_manual_status`
Set true to have the widget switch to machine log when in Manual mode.

The `GcodeDisplay` properties can be set in a stylesheet with the following code added to the .qss file (the following color choices are random).

```
EditorBase{
    qproperty-styleColorBackground: lightblue;
    qproperty-styleColorCursor:white;
    qproperty-styleColor0: black;
    qproperty-styleColor1: #000000; /* black */
    qproperty-styleColor2: blue;
    qproperty-styleColor3: red;
    qproperty-styleColor4: green;
    qproperty-styleColor5: darkgreen;
    qproperty-styleColor6: darkred;
    qproperty-styleColor7: deeppink;
    qproperty-styleColorMarginText: White;
    qproperty-styleColorMarginBackground: blue;
    qproperty-styleFont0: ”Times,12,-1,0,90,0,0,0,0,0”;
    qproperty-styleFont1: ”Times,18,-1,0,90,1,0,0,0,0”;
    qproperty-styleFont2: ”Times,12,-1,0,90,0,0,0,0,0”;
    qproperty-styleFont3: ”Times,12,-1,0,90,0,0,0,0,0”;
    qproperty-styleFont4: ”Times,12,-1,0,90,0,0,0,0,0”;
```

```
    qproperty-styleFont5: ”Times,12,-1,0,90,0,0,0,0,0”;
    qproperty-styleFont6: ”Times,12,-1,0,90,0,0,0,0,0”;
    qproperty-styleFont7: ”Times,12,-1,0,90,0,0,0,0,0”;
    qproperty-styleFontMargin: ”Times,14,-1,0,90,0,0,0,0,0”;
}
```

For `GcodeDisplay` widget’s *default G-code lexer*:

- **styleColor0 = Default**: Everything not part of the groups below
- **styleColor1 = LineNo and Comments**: Nxxx and comments (characters inside of and including *()* or anything after *;* (when used outside of parenthesis) with the exception of the note below)
- **styleColor2 = G-code**: G and the digits after
- **styleColor3 = M-code**: M and the digits after
- **styleColor4 = Axis**: XYZABCUVW
- **styleColor5 = Other**: EFHIJKDQLRPST (feed, rpm, radius, etc.)
- **styleColor6 = AxisValue**: Values following XYZABCUVW
- **styleColor7 = OtherValue**: Values following EFHIJKDQLRPST$

---

nota
For comments, the ”OtherValue” color (Color 5) can be used to highlight ”print,” ”debug,” ”msg,” ”logopen,” ”logappend,” ”logclose” ”log,” ”pyrun,” ”pyreload” ”abort,” ”probeopen” ”probeclose” inside of a parenthesis comment in a line of G-code. As well as ”py,” if a line that starts with ”;py,”. Examples: (print, text), (log, text), (msg, text), or (debug, text). Only the last of the examples will be highlighted if there are more than one on the same line.

Definición de fuentes:

```
”style name, size, -1, 0, bold setting (0-99), italics (0-1),
underline (0-1),0,0,0”
```

It is based on PyQt’s *QsciScintilla*.

12.7.2.9. `GcodeEditor` - G-code Program Editor Widget

This is an extension of the `GcodeDisplay` widget that **adds editing convenience**.

It is based on PyQt’s *QWidget* which incorporates `GcodeDisplay` widget.

12.7.2.10. `GCodeGraphics` - G-code Graphic Backplot Widget

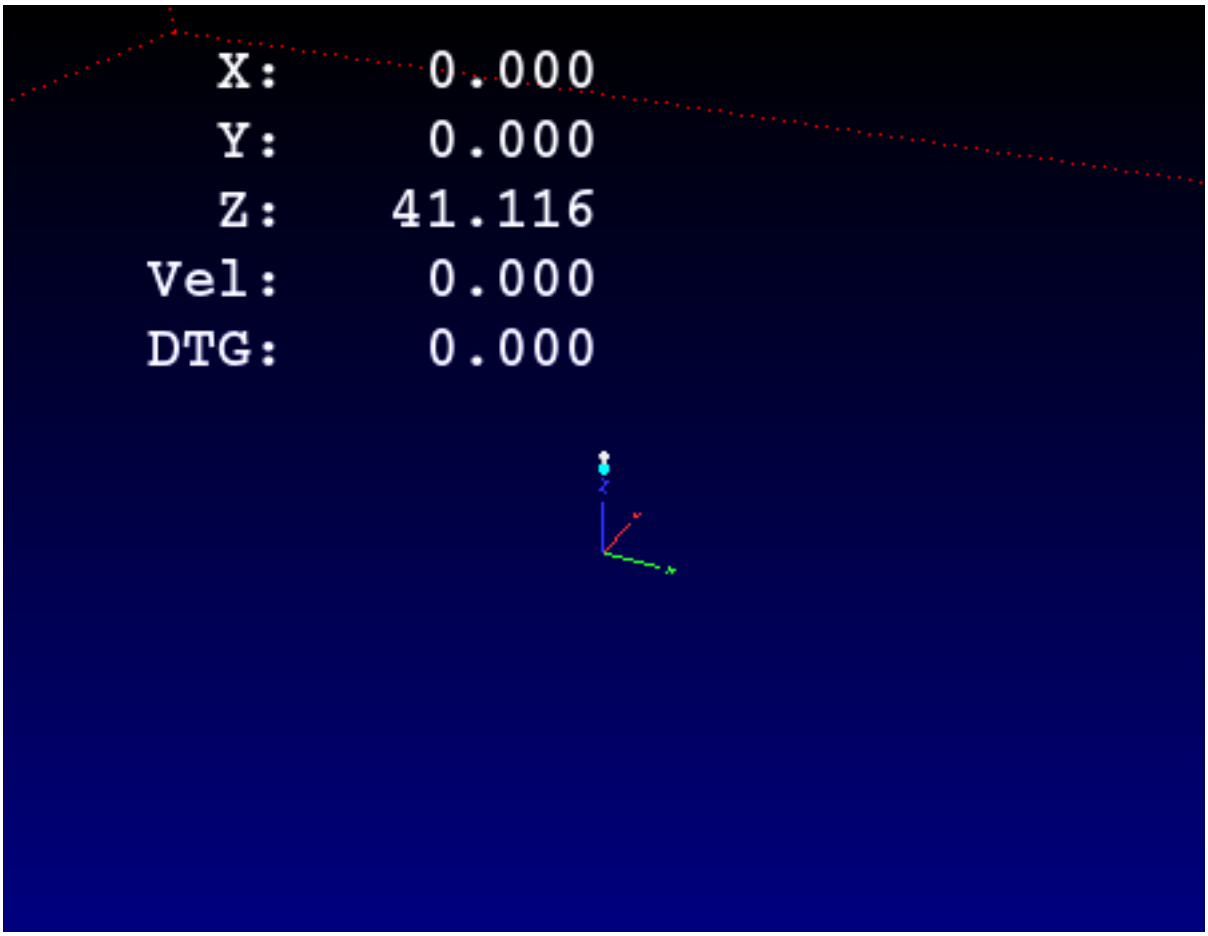


Figura 12.90: QtVCP GcodeGraphics: G-code Graphic Backplot Widget

This **displays the current G-code in a graphical form**.

Stylesheets Properties

`dro-font/dro-large-font` *(string)*
Sets the small and large DRO font properties
Here we reference with the widget base name; GCodeGraphics

```
GCodeGraphics{
    qproperty-dro_font:”monospace  bold 12”;
}
GCodeGraphics{
    qproperty-dro_large_font:”Times 25”;
}
```

`_view` *(string)*
Sets the *default view orientation* on GUI load.
Valid choices for a lathe are p, y, y2. For other screens, valid choices are p, x, y, z, z2.
The following shows an example of how to set this property (referenced using the widget user selected name):

```
#gcodegraphics{
    qproperty-_view: z;
}
```

_dro ***(bool)***
Determines whether or not to *show the DRO*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_dro: False;
}
```

_dtg ***(bool)***
Determine whether or not to *show the Distance To Go*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_dtg: False;
}
```

_metric ***(bool)***
Determines whether or not to *show the units in metric by default*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_metric: False;
}
```

_overlay ***(bool)***
Determines whether or not to *show the overlay by default*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_overlay: False;
}
```

_offsets ***(bool)***
Determines whether or not to *show the offsets by default*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_offsets: False;
}
```

_small_origin ***(bool)***
Determines whether or not to *show the small origin by default*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_small_origin: False;
}
```

overlay_color ***(primary, secondary, or RGBA formatted color)***
Sets the *default overlay color*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-overlay_color: blue;
}
```

`overlay_alpha` *(float)*
Sets the *default overlay alpha value*. This affects the opacity of the overlay when set between 0.0 and 1.0.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-overlay_alpha: 0.15;
}
```

`background_color` *(primary, secondary, or RGBA formatted color)*
Sets the *default background color*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-background_color: blue;
}
```

`+_use_gradient_background+` *(bool)*
Determines whether or not *use a gradient background by default*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_use_gradient_background: False;
}
```

`jog_color` *(primary, secondary, or RGBA formatted color)*
Sets the *default jog color*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-jog_color: red;
}
```

`Feed_color` *(primary, secondary, or RGBA formatted color)*
Sets the *default feed color*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-Feed_color: green;
}
```

`Rapid_color` *(primary, secondary, or RGBA formatted color)*
Sets the *default rapid color*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-Rapid_color: rgba(0, 0, 255, .5);
}
```

`InhibitControls` *(bool)*
Determines whether or not to *inhibit external controls by default*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-InhibitControls:True;
}
```

`MouseButtonMode` *(int)*
Changes the *mouse button behavior* to rotate, move or zoom within the preview.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-MouseButtonMode: 1;
}
```

There are 12 valid modes:

Mode	Move	Zoom	Rotate
0	Left	Middle	Right
1	Middle	Right	Left
2	Middle	Left	Right
3	Left	Right	Middle
4	Right	Left	Middle
5	Right	Middle	Left

Modes 6-11 are intended for machines that only require a 2D preview such as plasma or some lathes and have no rotate button assigned.

Mode	Move	Zoom
6	Left	Middle
7	Middle	Left
8	Right	Left
9	Left	Right
10	Middle	Right
11	Right	Middle

`MouseWheelInvertZoom` *(bool)*
Determines whether or not to *invert the zoom direction* when zooming with the mouse wheel. The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-MouseWheelInvertZoom:True;
}
```

ACTION functions The `ACTION` library can control the G-code graphics widget.

`ACTION.RELOAD_DISPLAY()`
Reload the current program which recalculates the origin/offsets.

`ACTION.SET_GRAPHICS_VIEW(_view_)`
The following `view` commands can be sent:

- `clear`
- `zoom-in`
- `zoom-out`
- `pan-up`
- `pan-down`
- `pan-right`
- `pan-left`
- `rotate-cw`
- `rotate-ccw`
- `rotate-up`
- `rotate-down`

- `overlay-dro-on`
- `overlay-dro-off`
- `overlay-offsets-on`
- `overlay-offsets-off`
- `alpha-mode-on`
- `alpha-mode-off`
- `inhibit-selection-on`
- `inhibit-selection-off`
- `dimensions-on`
- `dimensions-off`
- `grid-size`
- `record-view`
- `set-recorded-view`
- `P`
- `X`
- `Y`
- `Y2`
- `Z`
- `Z2`
- *set-large-dro*
- *set-small-dro*

ACTION.ADJUST_PAN(_X,Y_)
Directly set the relative pan of view in x and y direction.

ACTION.ADJUST_ROTATE(_X,Y_)
Directly set the relative rotation of view in x and y direction.

It is based on PyQt's *OpenGL* widget.

12.7.2.11. `JointEnableWidget` - FIXME

FIXME JointEnableWidget documentation

12.7.2.12. `JogIncrements` - Jog Increments Value Selection Widget

This widget allows the user to **select jog increment values for jogging**.

The jogging values come from the *INI file* under:

- `[DISPLAY]INCREMENTS`, or
- `[DISPLAY]ANGULAR_INCREMENTS`

This will be *available to all widgets* through `STATUS`.

You can select linear or angular increments by the property **`linear_option`** in Qt Designer property editor.

It is based on PyQt's *ComboBox*.

12.7.2.13. `MacroTab` - Special Macros Widget

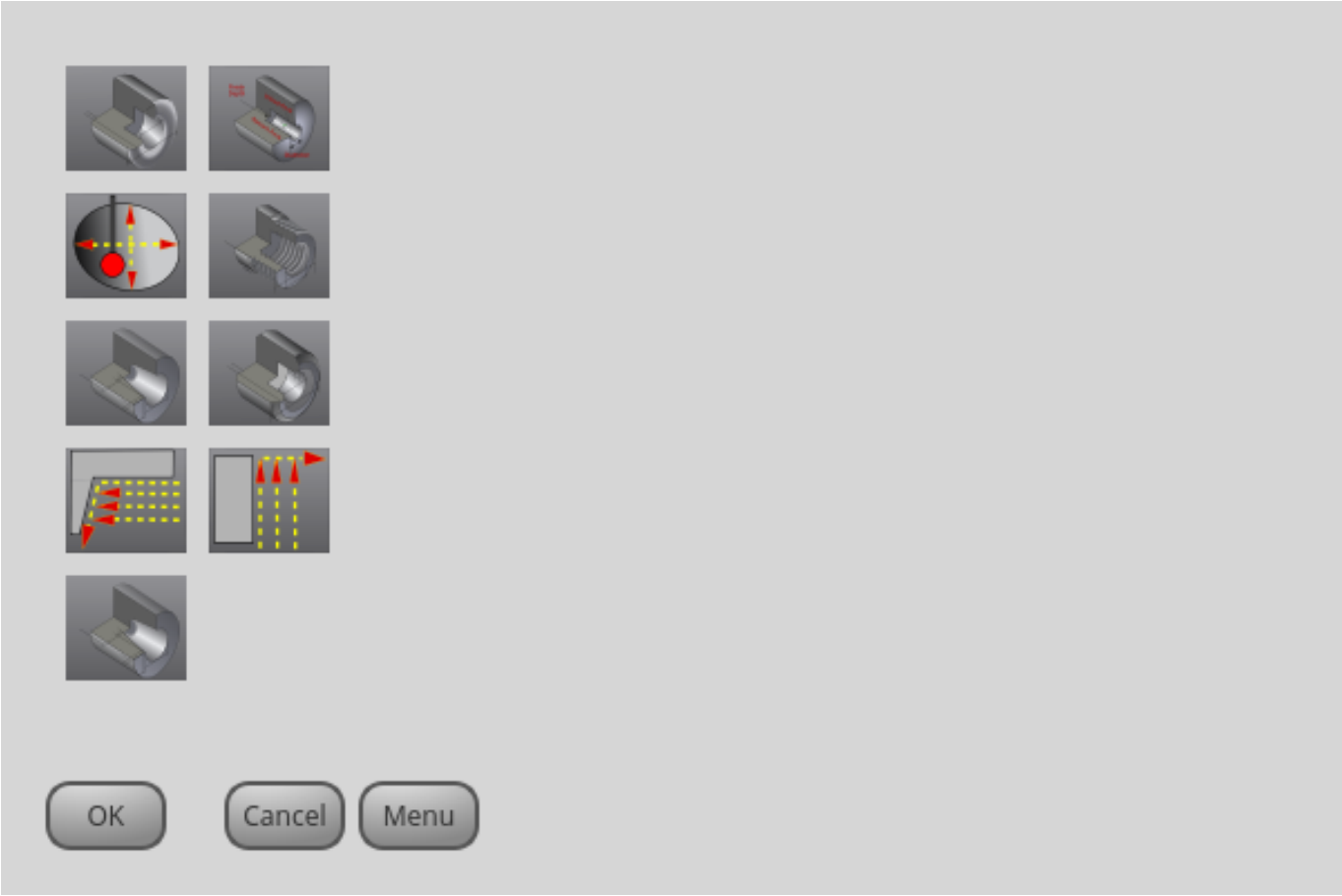


Figura 12.91: QtVCP `MacroTab`: Special Macros Widget

This widget allows a user to **select and adjust special macro programs** for doing small jobs.

It uses *images for visual representation* of the macro and for an icon.

It searches for special macros using the *INI definition*:

```
[RS274NGC]
SUBROUTINE_PATH =
```

The macros are **`O-word` subroutines with special comments** to work with the launcher. The first three lines *must* have the keywords below, the fourth is optional.

Here is a sample for the first four lines in an *O-word file*:

```
; MACROCOMMAND = Entry1,Entry2
; MACRODEFAULTS = 0,true
; MACROIMAGE = my_image.svg,Icon layer number,Macro layer number
; MACROOPTIONS = load:yes,save:yes,default:default.txt,path:~/macros
```

MACROCOMMAND This is the *first line* in the O-word file.

It is a **comma separated list of text to display above an entry**.
There will be **one for every variable required** in the O-word function.

If the macro does not require variables, leave it empty:

```
; MACROCOMMAND=
```

MACRODEFAULTS This must be the *second line* in the O-word file.

It is a **comma separated list of the default values for each variable** in the O-word function.

If you use the word `radiotrue`, `radiofalse`, `true` or `false` in the list, a *`radiobutton`* will be shown.
If you use the word `checktrue` or `checkfalse` in the list, a *`checkbox`* will be shown.
If you use the word `buttontrue` or `buttonfalse` in the list, a *`Checkable Pushbutton`* will be shown.
If the default has a decimal, macroTab assumes you want a float value otherwise an integer.

---

nota
When using radiobuttons, only set one radiobutton as true. Radio button are used for exclusive choices.

MACROIMAGE This must be the *third line* in the O-word file.

- **SVG Images**
  If using SVG image files, they must end with the `.svg` extension.

  The images must be added to *SVG layers* which are used to define the different images for macro and icon.

  Value is comma separated list of three ordered fields:

  ```
  ; MACROIMAGE=filename.svg,macro_layer_name[,icon_layer_name]
  ```

  With:

  filename.svg
  : SVG image file name as first field.
    It is assumed to be in the same folder as the O-word file.

  ***macro_layer_name**
  : Macro image layer name as second field.

  icon_layer_name
  : Icon image layer name as optional third field. If the third entry is missing, the same image will be used for macro and icon.

- **PNG/JPG Images**:
  Value remains a comma separated list:

  ```
  ; MACROIMAGE=macro_image.(png|jpg)[,icon_image.(png|jpg)]
  ```

  With:

  _macro_image_.(png|jpg)
  : Macro image file name as first field.
    It is assumed that the image file are in the same folder than the macro.

  _icon_image_.(png|jpg)
  : **Icon image file name** as optional second field.
    If the second entry is missing the same image will be used for macro and image.

If the keyword is present but the entries are missing, no images will be used.

MACROOPTIONS This *optional line must be the fourth* line in the O-word file.

It is a comma separated list of keyword and data, all of which are optional:

LOAD:yes
Shows a load button.

SAVE:yes
Shows a save button.

DEFAULT:ThisMacroData.txt
Sets the default preselected filename when loading/saving data for this macro.
It can be any valid filename but must end in *.txt*.

PATH:~/linuxcnc/nc_files/mySavedMacrosData
Sets the default directory folder to preselect when loading/saving data for this macro.

Here are stylesheet hints for adjusting the MacroTab widget.

```
MacroTab CustomButton{
    width: 20px;
    height: 40px;
}

MacroTab QPushButton {
    width: 80px;
    height: 40px;
}

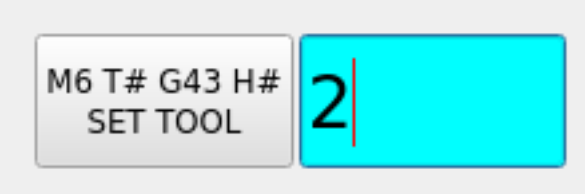
MacroTab QLabel {
    font: 24pt ”Lato Heavy”;
}

TouchSpinBox  LineEdit {
    font: 12pt ”Lato Heavy”;
}

TouchSpinBox  QPushButton {
    width: 60px;
    height: 100px;
}
```

12.7.2.14. `OperatorValueLine` - Operator Value Line Entry Widget

The operator enters values into this widget, which will be applied to a template and then optionally issued to the MDI either immediately or applied at a later time. The widget supports the optional popup calculator, keyboard, or tool chooser for touchscreen-friendly entry by setting the `dialog_keyboard_option`. To change which type of dialog is presented, edit the `dialog_code_option`.



The widget supports a formatting option which is passed to Python’s string `format()` to produce the final output for the MDI command. The special token `{value}` can be inserted anywhere in this format string where the value should appear. The formatting property is called `mdi_command_format_option`, e.g.:

- `M3 S{value}` to start the spindle at the speed entered by the operator.

- `M6 T{value} G43 H{value}` to issue a tool change and tool length offset change from the tool number entered

The widget may be configured to automatically issue the MDI command upon submit when `issue_mdi_on_su` is set to `True`. If `False` issuing the command may be done at a later time via a signal or function call from another widget.

In cases where `issue_mdi_on_submit_option` is `False`, calling the `issue_mdi()` function will issue the command. Slots attached to widgets such as PushButtons can trigger the MDI command when pressed, e.g.:

```
    def setSpindleSpeed(self, event):
        self.w.lineSpindleSpeed.issue_mdi()
        ACTION.SET_MANUAL_MODE()

    def setToolNumber(self, event):
        self.w.lineToolNumber.issue_mdi()
        ACTION.SET_MANUAL_MODE()
```

The widget tracks whether a value entered is pending and has not yet been issued via the property `isPendingValue`. This may be used to style the widget via the stylesheet. This can be used to alert the operator that they entered a value but another action must be taken to apply it.

The following style sheet excerpt will highlight the entry widget with a cyan background when values are pending and have not been applied.

```
#lineSpindleSpeed[isPendingValue=true],
#lineToolNumber[isPendingValue=true] {
    background: cyan;
}

#lineSpindleSpeed[isPendingValue=false],
#lineToolNumber[isPendingValue=false] {
    background: none;
}
```

12.7.2.15. `MDILine` - MDI Commands Line Entry Widget

One can **enter MDI commands** here.

A popup keyboard is available.

Embedded Commands También hay **comandos embebidos** disponibles para este widget.

Enter any of these *case sensitive* commands to load the respective program or access the feature:

HALMETER
: Starts LinuxCNC `halmeter` utility.

HALSHOW
: Starts LinuxCNC `halshow` utility.

HALSCOPE
: Starts LinuxCNC `halscope` utility.

STATUS
: Starts LinuxCNC `status` utility.

CALIBRATION
: Inicia LinuxCNC Calibration

CLASSICLADDER
Starts the ClassicLadder GUI if the *ClassicLadder realtime HAL component* was loaded by the machine's config files.

PREFERENCE
Loads the preference file into the `GcodeEditor`.

CLEAR HISTORY
Clears the MDI History.

net
See `halcmd net` commands.
An error will result if the command is unsuccessful.

- *Sintaxis*: `net <nombre-de-señal> <nombre-del-pin>`
- *Example*: `net plasmac:jog-inhibit motion.jog-stop`

setp
Establece el valor de un pin o parámetro.
Los valores válidos dependerán del tipo de objeto de pin o parámetro.
Resulta en error si los tipos de datos no coinciden o el pin es conectado a una señal.

- *Sintaxis*: `setp <nombre-pin/parámetro> <valor>`
- *Example*: `setp plasmac.resolution 100`

unlinkp
Disconnects a pin from a signal.
An error will result if the pin does not exist.
Running LinuxCNC from terminal may help determine the root cause as error messages from `hal_lib.c` will be displayed there.

- *Sintaxis*: `unlinkp <nombre-pin>`
- *Example*: `unlinkp motion.jog-stop`

---

nota
The `MDILine` function **`spindle_inhibit`** can be used by a GUI's handler file to inhibit `M3`, `M4`, and `M5` spindle commands if necessary.

It is based on PyQt's *QLineEdit*.

12.7.2.16. `MDIHistory` - MDI Commands History Widget

Displays a **scrollable list of past MDI command**.

An edit line is embedded for MDI commands. The same MDILine embedded commands may be accessed from this widget.

The history is *recorded on a file defined in the INI* under the heading `[DISPLAY]` (this shows the default):

```
MDI_HISTORY_FILE = '~/.axis_mdi_history'
```

12.7.2.17. `MDITouchy` - Touch Screen MDI Entry Widget

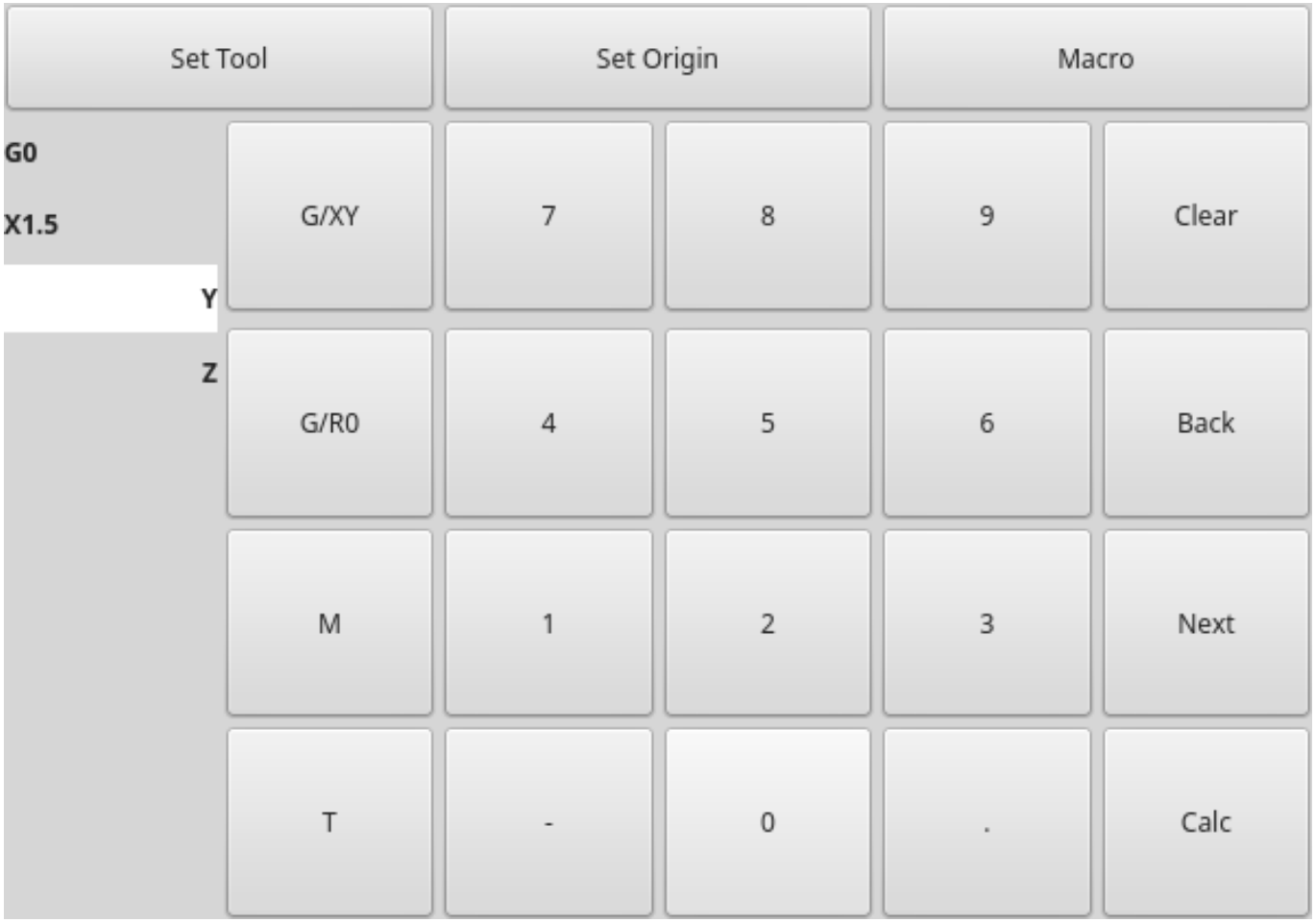


Figura 12.92: QtVCP `MDITouchy`: Touch Screen MDI Entry Widget

This widget displays **buttons and entry lines to use for entering MDI commands**.

Based on LinuxCNC's Touchy screen's MDI entry process, its large buttons are most useful for touch screens.

Para usar `MDITouchy`:

- First press one of the `G/XY`, `G/R0`, `M` or `T` button. On the left will show the entry fields that can be filled out.
- Then press `Next` and `Back` to navigate between fields.
- `Calc` will pop up a calculator dialog.
- `Clear` limpia la entrada actual.
- `Set Tool` invocará un cambio de herramienta.
- `Set Origin` will allow setting the origin of the current G6x system.
- `Macro` will call any available macro ngc programs.

The widget *requires an explicit call to MDITouchy Python code* to *actually run the MDI command*:

- **For handler file code**
  If the widget was named *mditouchy* in Qt Designer, the command below would run the displayed MDI command:

```
self.w.mditouchy.run_command()
```

- **For action button use**
  If the widget was named *mditouchy* in Qt Designer, use the action button's *Call Python commands* option and enter:

```
INSTANCE.mditouchy.run_command()
```

El botón macro *cicla a través de las macros definidas en el encabezado [DISPLAY] de INI*.

Add one or more `MACRO` lines of the following format:

```
MACRO = macro_name [param1] [... paramN]
```

In the example below, `increment` is the name of the macro, and it accepts two parameters, named `xinc` and `yinc`.

```
MACRO = incerment xinc yinc
```

Now, place the macro in a file named `macro_name.ngc` in the `PROGRAM_PREFIX` directory, or into any directory in the `SUBROUTINE_PATH` specified in the INI file.

Keeping on with the example above, it would be named `increment.ngc` and its content could look like:

```
O<increment> sub
G91 G0 X#1 Y#2
G90
O<increment> endsub
```

Notice the *name of the sub matches the file name and macro name exactly*, including case.

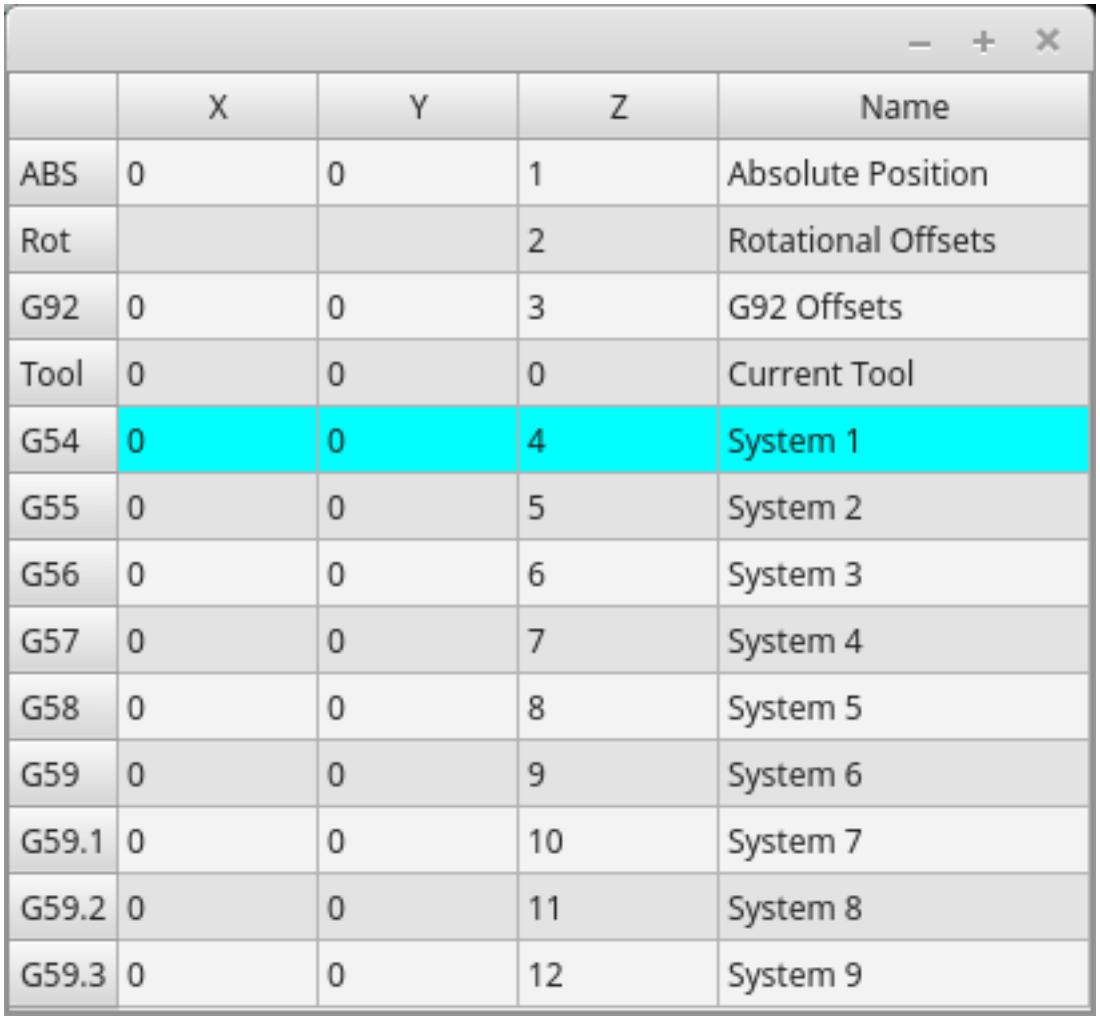
Cuando invoque la macro presionando el botón Macro, puede ingresar valores de parámetros (`xinc` y `yinc èn nuestro ejemplo). + Estos son pasados a la macro posicionalmente:  #1` y `#2... #N` respectivamente.
Los parámetros en blanco se pasan como valor `0`.

If there are several different macros, press the Macro button repeatedly to cycle through them.

In this simple example, if you enter -1 for xinc and invoke the running of the MDI cycle, a rapid *G0* move will be invoked, moving one unit to the left.

This macro capability is useful for edge/hole probing and other setup tasks, as well as perhaps hole milling or other simple operations that can be done from the panel without requiring specially-written G-code programs.

12.7.2.18. `OriginOffsetView` - Origins View and Setting Widget



	X	Y	Z	Name
ABS	0	0	1	Absolute Position
Rot			2	Rotational Offsets
G92	0	0	3	G92 Offsets
Tool	0	0	0	Current Tool
G54	0	0	4	System 1
G55	0	0	5	System 2
G56	0	0	6	System 3
G57	0	0	7	System 4
G58	0	0	8	System 5
G59	0	0	9	System 6
G59.1	0	0	10	System 7
G59.2	0	0	11	System 8
G59.3	0	0	12	System 9

Figura 12.93: QtVCP `OriginOffsetsView`: Origins View and Setting Widget

This widget allows one to **visualize and modify User System Origin offsets** directly.

It will *update LinuxCNC's Parameter file* for changes made or found.

The settings can only be changed in LinuxCNC after homing and when the motion controller is idle.

The display and entry will change between metric and imperial, based on LinuxCNC's *current* `G20` / `G21` setting.

The current in-use user system will be highlighted.

Extra actions can be integrated to manipulate settings.
These actions depend on extra code added either to a combined widget, like `originoffsetview` dialog, or the screens handler code.
Typical actions might be *Clear Current User offsets* or *Zero X*.

Clicking on the columns and rows allows one to adjust the settings.

A dialog can be made to popup for data or text entry.

The comments section will be recorded in the preference file.

It is based on PyQt's *QTableView*, *QAbstractTableModel*, and *ItemEditorFactory*.
Properties, functions and styles of the PyQt base objects are always available.

Properties `OriginOffsetView` has the following properties:

`dialog_code_string`
Sets which dialog will pop up with numerical entry.

`test_dialog_code_string`
Sets which dialog will pop up with text entry.

`metric_template`
Metric numerical data format.

`imperial_template`
Imperial numerical data format.

`styleCodeHighlight`
Current in-use user system highlight color.

These can be set in:

- Qt Designer, in
- Python handler code

```
self.w.originoffsetview.setProperty('dialog_code','CALCULATOR')
self.w.originoffsetview.setProperty('metric_template','%10.3f')
```

- Or (if appropriate) in stylesheets

```
OriginOffsetView{
  qproperty-styleColorHighlist: lightblue;
}
```

12.7.2.19. `RadioAxisSelector` - FIXME

FIXME RadioAxisSelector documentation

12.7.2.20. `RoundButton` - Round Shapped `ActionButton` Widget

Round buttons work the same as *ActionButtons* other than the button is cropped round.

They are intended only to be visually different.

They have *two path properties* for displaying **images on true and false**.

12.7.2.21. `StateLabel` - Controller Modes State Label Display Widget

This will **display a label based on the machine controller modes true/false states**.

You can select between different texts based on true or false.

Propiedades de selección de estados The states are selectable via these properties:

`css_mode_status`
`True` when machine is in `G96` *Constant Surface Speed Mode*.

diameter_mode_status
`True` when machine is in `G7` *Lathe Diameter Mode*.

fpr_mode_status
`True` when machine is in `G95` *Feed per revolution Mode*.

metric_mode_status
`True` when machine is in `G21` *Metric Mode*.

Text templates properties

true_textTemplate
This will be the text set when the option is `True`.
You can use *Qt rich text* code for different fonts/colors etc.
Typical template for metric mode in true state, might be: *Metric Mode*

false_textTemplate
This will be the text set when the option is `False`.
You can use *Qt rich text* code for different fonts/colors etc.
Typical template for metric mode in false state, might be: *Imperial Mode*.

It is based on PyQt's *QLabel*.

12.7.2.22. `StatusLabel` - Controller Variables State Label Display Widget

This will display a label based on selectable status of the machine controller.
You can change how the status will be displayed by substituting Python formatting code in the text template. You can also use rich text for different fonts/colors etc.

Selectable States These states are selectable:

actual_spindle_speed_status
Used to display the actual spindle speed as *reported from the HAL pin* `spindle.0.speed-i`.
It's converted to *RPM*.
A `textTemplate` of `%d` would typically be used.

actual_surface_speed_status
Used to display the actual cutting surface speed on a lathe based on X axis and spindle speed.
It's converted to distance per minute.
A `textTemplate` of `%4.1f` (feet per minute) and `altTextTemplate` of `%d` (meters per minute) would typically be used.

blendcode_status
Muestra la configuración `G64` actual.

current_feedrate_status
Muestra la velocidad de alimentación real actual.

current_FPU_status
Muestra la alimentación real actual por unidad.

fcode_status
Muestra la configuración de código `F` programada actual.

feed_override_status
Muestra la configuración de anulación de alimentación actual en porcentaje.

filename_status
Shows the last loaded file name.

filepath_status
Shows the last loaded full file path name.

gcode_status
Muestra todos los códigos G activos.

gcode_selected_status
Muestra la línea de código G seleccionada actual.

halpin_status
Shows the HAL pin output of a selected HAL pin.

jograte_status
Muestra la velocidad de trote basada en QtVCP actual.

jograte_angular_status
Muestra la velocidad actual de trote angular basada en QtVCP.

jogincr_status
Muestra el incremento de trote basado en QtVCP actual.

jogincr_angular_status
Muestra el incremento actual de trote angular basado en QtVCP.

machine_state_status
Shows the current *machine interpreter state* using the text described from the machine_state_list. The interpreter states are:

- `Estopped`
- `Running`
- `Stopped`
- `Paused`
- `Waiting`
- `Reading`

max_velocity_override_status
Muestra la configuración actual de anulación de velocidad máxima del eje.

mcode_status
Shows *all active M-codes*.

motion_type_status
Shows current type of machine motion using the text described from the motion_type_list.

- *None*
- *Rapid*
- *Feed*
- *Arc*
- *Tool Change*
- *Probe*
- *Rotary Index*

requested_spindle_speed_status
Shows the requested spindle speed - actual may be different.

rapid_override_status
Muestra el ajuste manual de rápido actual en porcentaje (0-100).

spindle_override_status
Muestra la configuración actual de anulación de husillo en porcentaje.

timestamp_status
Shows the time based on the system settings.
An example of a useful `textTemplate` setting: `%I:%M:%S%p`.
See the Python time module for more info.

tool comment_status
Returns the comment text from the current loaded tool.

tool diameter_status
Returns the diameter from the current loaded tool.

tool_number_status
Devuelve el número de la herramienta cargada actualmente.

tool_offset_status
Returns the offset of the current loaded tool, indexed by `index_number` to select axis (0=x,1=y,etc.).

user_system_status
Muestra el *sistema de coordenadas de usuario activo* (configuración `G5x`).

Other Properties

index_number
Integer that specifies the tool status index to display.

state_label_list
List of labels used to describe different machine states.

motion_label_list
List of labels used to describe different motion types.

halpin_names
Name of a halpin to monitor (must be the complete name, including the HAL component basename).

textTemplate
This is usually used for **imperial (G20) or angular numerical settings**, though not every option has imperial/metric conversion.
This uses *Python formatting rules* to set the text output.
One can use `%s` for no conversion, `%d` for integer conversion, `%f` for float conversion, etc.
You can also use *Qt rich text* code.
Typical template used for formatting imperial float numbers to text would be `%9.4f` or `%9.4f inch`.

alt_textTemplate
This is usually used for **metric (G21) numerical settings**.
This uses *Python formatting rules* to set the text output.
Typical template used for formatting metric float to text would be `%10.3f` or `%10.3f mm`.

It is based on PyQt's *QLabel*.

12.7.2.23. `StatusImageSwitcher` - Controller Status Image Switcher

Status image switcher will **switch between images based on LinuxCNC states**.

***watch_spindle**
Toggles between *3 images*: `stop`, `fwd`, `revs`.

***watch_axis_homed**
Toggles between *2 images*: `axis not homed`, `axis homed`.

***watch_all_homed**
Would toggle between *2 images*: `not all homed`, `all homed`.

***watch_hard_limits**
Would toggle between *2 images or one per joint*.

Here is an example of using it to display an icon of Z axis homing state:

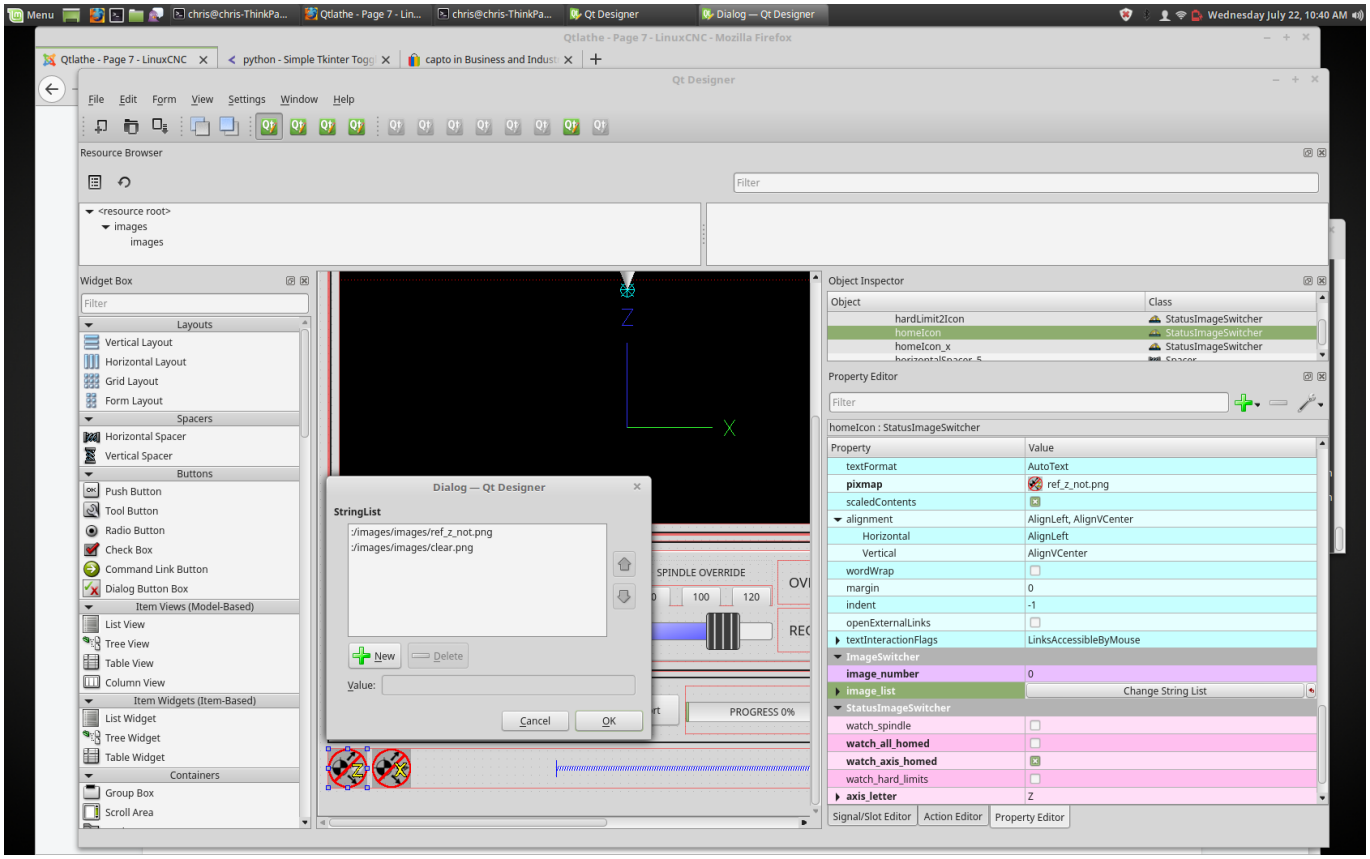


Figura 12.94: QtVCP `StatusImageSwitcher`: Controller Status Image Switcher

In the properties section notice that:

- `watch_axis_homed` is checked
- `axis_letter` is set to Z

If you double click the `image_list` a dialog will show and allow you to add image paths to.

If you have one image as an icon and one *clear image* then that will look like it shows and *hides the icon*.

Selecting image paths can be done by selecting the `pixmap` property and selecting an image.

---

nota
The `pixmap` setting is for test display only and will be ignored outside of Qt Designer.

- Right click the image name and you should see *Copy path*.
- Click *Copy path*.
- Now double click the *image list* property so the dialog shows.
- Click the *New* button.
- Paste the image path in the entry box.

Do that again for the next image.
Use a clear image to represent a hidden icon.

You can *test the images display* from the image list by changing the `image number`. In this case `0` is unhomed and `1` would be homed.
This is for test display only and will be ignored outside of Qt Designer.

12.7.2.24. `StatusStacked` - Mode Status Display Switching Widget

This widget **displays one of three panels based on LinuxCNC's mode**.

This allows you to automatically display different widgets on *Manual*, *MDI* and *Auto* modes. **TODO** It is based on PyQt's *QStacked* widget.

12.7.2.25. `ScreenOption` - General Options Setting widget

This widget doesn't add anything visually to a screen but **sets up important options**.

This is the *preferred way to use these options*.

Properties These properties can be set in Qt Designer, in Python handler code or (if appropriate) in stylesheets.

These include:

`halCompBaseName`
: If left empty QtVCP will use the screen's name as the HAL component's basename.
If set, QtVCP will use this string as the HAL component's basename.
If the `-c` command line option is used when loading QtVCP, it will use the name specified on the command line - it overrides all above options.
If you programmatically set the basename in the `handlerfile` - it will override all above options.
This property cannot be set in stylesheets.

`notify_option`
: Hooking into the desktop notification bubbles for error and messages.

`notify_max_messages`
: Number of messages shown on screen at one time.

`catch_close_option`
: Catching the close event to pop up a *'are you sure' prompt*.

`close_overlay_color`
: Color of transparent layer shown when quitting.

`catch_error_option`
: *Monitoring of the LinuxCNC error channel*.
This also sends the message through `STATUS` to anything that registers.

play_sounds_option
Playing sounds using `beep`, `espeak` and the system sound.

use_pref_file_option
Setting up a *preferences file path*.
Using the magic word `WORKINGFOLDER` in the preference file path will be replaced with the launched configuration path, e.g. `WORKINFOLDER/my_preferences`.

use_send_zmq_option
Used to initiate *ZMQ based outgoing messages*.

use_receive_zmq_messages
Used to initiate *ZMQ based in coming messages*.
These messages *can be used to call functions in the handler file*, allowing **external programs to integrate tightly with QtVCP** based screens.

embedded_program_option
Embed programs defined in the *INI*.

default_embed_tab
This is the property for a *default location to embed external programs*.
It should be set to name of a tab page widget in Qt Designer.

focusOverlay_option
Focus_overlay will put a transparent image or colored panel over the main screen to emphasize focus to an external event - typically a dialog.

messageDialog_option
Sets up the message dialog - used for general messages.

message_overlay_color
Color of transparent layer shown when the message dialog is shown.

closeDialog_option
Sets up the standard close screen prompt dialog.

entryDialog_option
Sets up the numerical entry dialog.

entryDialogSoftKey_option
Sets up a floating software keyboard when entry dialog is focused.

entry_overlay_color
Color of transparent layer shown when the entry dialog is shown.

toolDialog_option
Sets up the manual tool change dialog, including HAL pin.

tool_overlay_color
Color of transparent layer shown when the tool dialog is shown.

ToolUseDesktopNotify
Opción para usar diálogos de notificación de escritorio para cambio manual de herramienta.

ToolFrameless
Frameless dialogs can not be easily moved by users.

fileDialog_option
Sets up the file choosing dialog.

file_overlay_color
Color of transparent layer shown when the file dialog is shown.

keyboardDialog_option
Sets up a keyboard entry widget.

keyboard_overlay_color
Color of transparent layer shown when the keyboard dialog is shown.

vesaProbe_option
Sets up the Versa style probe dialog.

versaProbe_overlay_color
Color of transparent layer shown when the `versaProbe` dialog is shown.

macroTabDialog_option
Sets up the macro selection dialog.

macroTab_overlay_color
Color of transparent layer shown when the `macroTab` dialog is shown.

camViewDialog_option
Sets up the camera alignment dialog.

camView_overlay_color
Color of transparent layer shown when the `camView` dialog is shown.

toolOffset_option
Sets up the tool offset display/editor dialog.

toolOffset_overlay_color
Color of transparent layer shown when the `toolOffset` dialog is shown.

originOffset_option
Sets up the origin display/editor dialog.

originOffset_overlay_color
Color of transparent layer shown when the `originOffset` dialog is shown.

calculatorDialog_option
Sets up the calculator entry dialog.

calculator_overlay_color
Color of transparent layer shown when the calculator dialog is shown.

machineLogDialog_option
Sets up a dialog to display logs from the machine and QtVCP.

machineLog_overlay_color
Color of transparent layer shown when the `machineLog` dialog is shown.

runFromLineDialog_option
Sets up a dialog to display starting options when starting machine execution from a arbitrary line.

runFromLine_overlay_color
Color of transparent layer shown when the `runFromLine` dialog is shown.

user1Color
Optional color the screen designer can use in their design.

user2Color
Optional color the screen designer can use in their design.

user3Color
Optional color the screen designer can use in their design.

user4Color
Optional color the screen designer can use in their design.

user5Color
Optional color the screen designer can use in their design.

user6Color
Optional color the screen designer can use in their design.

user7Color
Optional color the screen designer can use in their design.

user8Color
Optional color the screen designer can use in their design.

user9Color
Optional color the screen designer can use in their design.

user10Color
Optional color the screen designer can use in their design.

Setting Properties Programmatically The screen designer chooses the **default settings of the `screenOptions` widget**.

Once chosen, most won't ever need to be changed. But if needed, some can be changed in the handler file or in stylesheets.

- **In the handler file**:
  Here we reference the widget by the Qt Designer user defined name:

```
# red,green,blue,alpha 0-255
color = QtGui.QColor(0, 255, 0, 191)
self.w.screen_options.setProperty('close_overlay_color', color)
self.w.screen_options.setProperty('play_sounds_option',False)
```

- **In style sheets**:
  Here we can reference the widget by Qt Designer user defined name or by widget class name.

```
/* red, green, blue 0-255, alpha 0-100% or 0.0 to 1.0 */
/* the # sign is used to refer to Qt Designer defined widget name */
/* matches/applied to only this named widget */
#screen_options {
  qproperty-close_overlay_color: rgba(0, 255, 0, 0.75)
}
```

Some settings are only checked on startup so will not cause changes after startup. In these cases you would need to *make the changes in Qt Designer only*.

Preference File Entries If the *preference file option is selected*, `screenOption` widget will make an **INI based preference file**.

While *other QtVCP widgets will add to this list*, the `screenOptions` widget will add these entries under the following headings:

[SCREEN_OPTIONS]

catch_errors *(bool)* , desktop_notify *(bool)*
Whether to display errors/messages in the system's notification mechanism.

notify_max_msgs *(int)*
Number of displayed errors at one time.

shutdown_check ***(bool)***
Whether to pop a confirmation dialog.

sound_player_on ***(bool)***
Enciende o apaga todos los sonidos.

[MCH_MSG_OPTIONS]

mchnMsg_play_sound ***(bool)***
To play alert sound when dialog pops.

mchnMsg_speak_errors ***(bool)***
To use Espeak to speak error messages.

mchnMsg_speak_text ***(bool)***
To use Espeak to speak all other messages.

mchnMsg_sound_type ***(str)***
Sound to play when messages displayed. See notes below.

[USER_MSG_OPTIONS]

usermsg_play_sound ***(bool)***
To play alert sound when dialog pops.

userMsg_sound_type ***(str)***
Sound to play when user messages displayed. See notes below.

userMsg_use_focusOverlay ***(bool)***

[SHUTDOWN_OPTIONS]

shutdown_play_sound ***(bool)*** **, shutdown_alert_sound_type** ***(str)***
Sound to play when messages displayed. See notes below.

shutdown_exit_sound_type ***(str)***
Sound to play when messages displayed. See notes below.

shutdown_msg_title ***(str)***
Short title string to display in dialog.

shutdown_msg_focus_text ***(str)***
Large text string to superimpose in focus layer.

shutdown_msg_detail ***(str)***
Cadena mas larga y descriptiva a mostrar en el diálogo.

NOTIFY_OPTIONS

notify_start_greeting ***(bool)***
Whether to display a greeting dialog on start-up.

notify_start_title ***(str)***
Short Title string.
If the speak option is also selected it will be spoken with Espeak.

notify_start_detail ***(str)***
Cadena de descripción mas larga.

notify_start_timeout ***(int)***
Time in seconds to display before closing.

*_sound_type entries

- **System Sounds**
  In Debian/Ubuntu/Mint based installations these *system sounds* should be available as sound-type entries above:

- ERROR
- READY
- DONE
- ATTENTION
- RING
- LOGIN
- LOGOUT
- BELL

These Sound options require `python3-gst1.0` installed.

- **Audio Files**
  You can also specify a *file path to an arbitrary audio file*.
  You can use ~ in path to substitute for the user home file path.
- **Kernel Beeps**
  If the `beep` *kernel module* is installed and it is not disabled, these sound-type entries are available:
  - BEEP
  - BEEP_RING
  - BEEP_START
- **Text-To-Speech**
  If the *Espeak* module (`python3-espeak`) is installed, you can use the `SPEAK` entry to pronounce text:
- **`SPEAK '_my message_'`**

12.7.2.26. `StatusSlider` - Controller Setting Adjustment Slider Widget

This widget allow the user to **adjust a LinuxCNC setting via a slider**.

The widget can adjust:

- Jog rate
- Angular jog rate
- Velocidad de alimentación
- Spindle override rate
- Rapid override rate

Properties `StatusSlider` has the following properties:

`halpin_option`
Sets option to make a HAL float pin that reflects current value.

`rapid_rate`
Selecciona un deslizante de porcentaje de ajuste de rápidos.

`feed_rate`
Selecciona un deslizante de ajuste velocidad de alimentación.

`spindle_rate`
Selecciona un deslizante de porcentaje de ajuste de rápidos.

jograte_rate
Selects a linear jograte slider.

jograte_angular_rate
Selects a angular jograte slider.

max_velocity_rate
Selecciona un deslizante de velocidad máxima.

alertState
String to define style change: `read-only`, `under`, `over` and `normal`.

alertUnder
Sets the float value that signals the stylesheet for *under* warning.

alertOver
Sets the float value that signals the stylesheet for *over* warning.

These can be set in:

- Qt Designer
- Python handler code,

```
self.w.status_slider.setProperty('spindle_rate',True)
self.w.status_slider.setProperty('alertUnder',35)
self.w.status_slider.setProperty('alertOver',100)
```

- Or (if appropriate) in stylesheets.

```
/* warning colors for overrides if out of normal range*/
/* widget object name is slider_spindle_ovr */

#slider_spindle_ovr[alertState='over'] {
    background: red;
}
#slider_spindle_ovr[alertState='under'] {
    background: yellow;
}
```

It is based on PyQt's *QSlider*.

12.7.2.27. `StateLED` - Controller State LED Widget

This widget gives **status on the selected LinuxCNC state**.

States The state options are:

is_paused_status , is_estopped_status , is_on_status , is_idle_status_ , is_homed_status , is_fl

Properties There are properties that can be changed:

halpin_option
Adds an output pin that reflects selected state.

invert_state_status
: Invert the LED state compared to the LinuxCNC state.

diameter
: Diámetro del LED.

color
: Color of the LED when on.

off_color
: Color of the LED when off.

alignment
: Guía de alineación Qt.

state
: Current state of LED (for testing in Qt Designer).

flashing
: Enciende y apaga la opción de parpadeo.

flashRate
: Sets the flash rate.

The LED properties can be defined in a stylesheet with the following code added to the `.qss` file.

```
State_LED #name_of_led{          (1)
  qproperty-color: red;
  qproperty-diameter: 20;
  qproperty-flashRate: 150;
}
```

(1) `name_of_led` would be the name defined in Qt Designer's editor.

It is based on the *LED* widget.

12.7.2.28. `StatusAdjustmentBar` - Controller Value Setting Widget

This widget allows **setting values using buttons while displaying a bar**.

It also has an *optional hi/low toggle button* that can be held down to set the **levels**.

The widget can adjust:

- Jog rate
- Angular jog rate
- Velocidad de alimentación
- Spindle override rate
- Rapid override rate

It is based on PyQt's *QProgressBar*.

12.7.2.29. `SystemToolButton` - User System Selection Widget

This widget allows you to **manually select a G5x user system by pressing and holding**.
If you don't set the button text it will automatically update to the current system.
It is based on PyQt's *QToolButton*.

12.7.2.30. `StateEnableGridlayout` - Controller State Enabled Container Widget

`_disable the widgets inside it depending on LinuxCNC's current state_.`

This is a **container that other widgets can be placed in**.
Embedded widgets are be greyed-out when the `StateEnableGridlayout` is disabled.
It can selectably react to:

- Machine on
- Interpreter idle
- E-stop off
- All-homed

It is based on PyQt's *QGridLayout*.

12.7.2.31. `StatusImageSwitcher` - Controller Status Image Switching Widget

This widget will **display images based on LinuxCNC status**.
You can watch:

- el estado del husillo,
- the state of all homed,
- the state of a certain axis homed,
- the state of hard limits.

It is based on PyQt's FIXME

12.7.2.32. `ToolOffsetView` - Tools Offsets View And Edit Widget

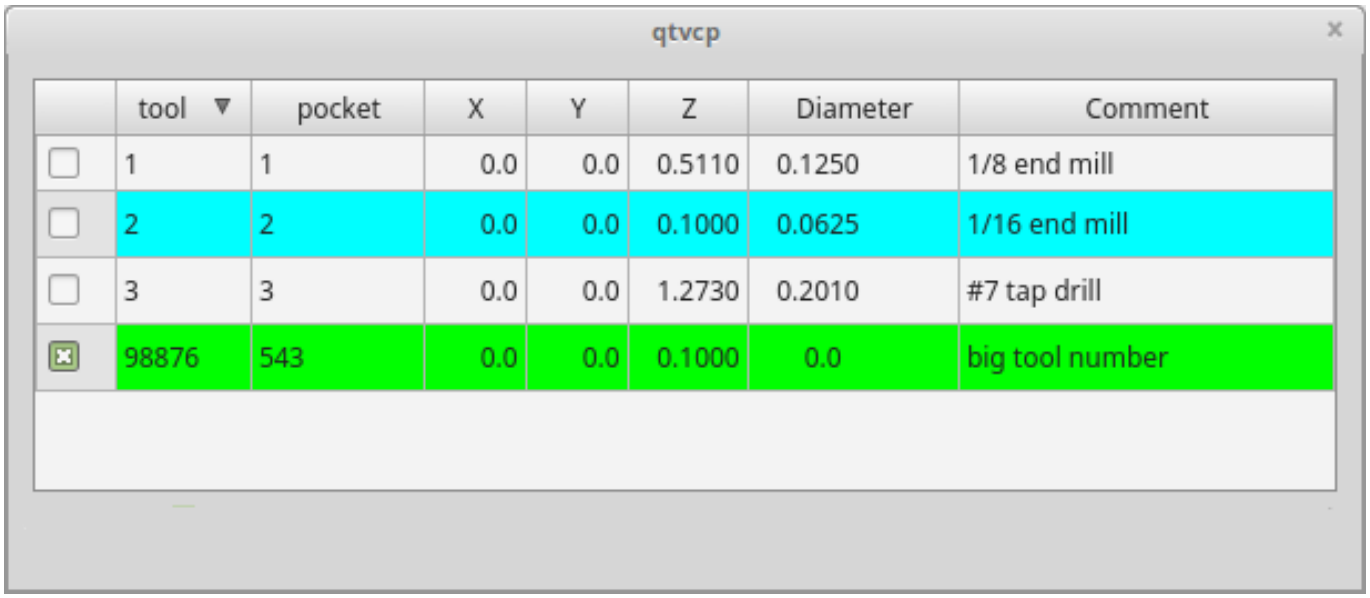


Figura 12.95: QtVCP `ToolOffsetView`: Tools Offsets View And Edit Widget

This widget **displays and allows one to modify tools offsets**.

It will *update LinuxCNC's tool table* for changes made or found.

The tool settings can only be changed in LinuxCNC after homing and when the motion controller is idle.

The display and entry will change between metric and imperial based on LinuxCNC's *current* `G20/G21` setting.

The current in-use tool will be highlighted, and the current selected tool will be highlighted in a different color.

The checkbox beside each tool can be used to select too for an *action* that depends on extra code added either to a combined widget, like the `toolOffsetView` dialog or the screens handler code. Typical actions are *load selected tool*, *delete selected tools*, etc.

Clicking on the columns and rows allows one to adjust the settings.

A dialog can be made to popup for data or text entry.

The comments section will typically be displayed in the manual tool change dialog.

If using a *lathe configuration*, there can be columns for X and Z wear.

To use these columns to adjust the *tool wear*, it requires a remapped tool change routine.

It is based on PyQt's *QTableView*, *QAbstractTableModel*, and *ItemEditorFactory*.
Properties, functions and styles of the PyQt base objects are always available.

Properties `ToolOffsetView` has properties that can be set in Qt Designer, in Python handler code or (if appropriate) in stylesheets:

`dialog_code_string`
: Sets which dialog will pop up with numerical entry.

`text_dialog_code_string`
: Sets which dialog will pop up with text entry.

metric_template
Metric numerical data format.

imperial_template
Imperial numerical data format.

styleCodeHighlight
Current tool-in-use highlight color.

styleCodeSelected
Selected highlight color.

En un archivo manejador:

```
self.w.tooloffsetview.setProperty('dialog_code','CALCULATOR')
self.w.tooloffsetview.setProperty('metric_template','%10.3f')
```

y en hojas de estilo:

```
ToolOffsetView{
  qproperty-styleColorHighlist: lightblue;
  qproperty-styleColorSelected: #444;
}
```

Funciones `ToolOffsetView` has some functions useful for screen builders to add actions:

add_tool()
Adds a blank dummy tool (99) that the user can edit to suit.

delete_tools()
Elimina la herramientas seleccionadas con casillas de verificación.

get_checked_list()
Returns a list of tools selected by checkboxs.

set_all_unchecked()
De-selecciona todas las herramientas seleccionadas.

Example for handler file executing aforementioned functions.

```
self.w.tooloffsetview.add_tool()
self.w.tooloffsetview.delete_tools()
toolList = self.w.tooloffsetview.get_checked_list()
self.w.tooloffsetview.set_all_unchecked()
```

12.7.2.33. `VersaProbe` - Mill Probing Widget

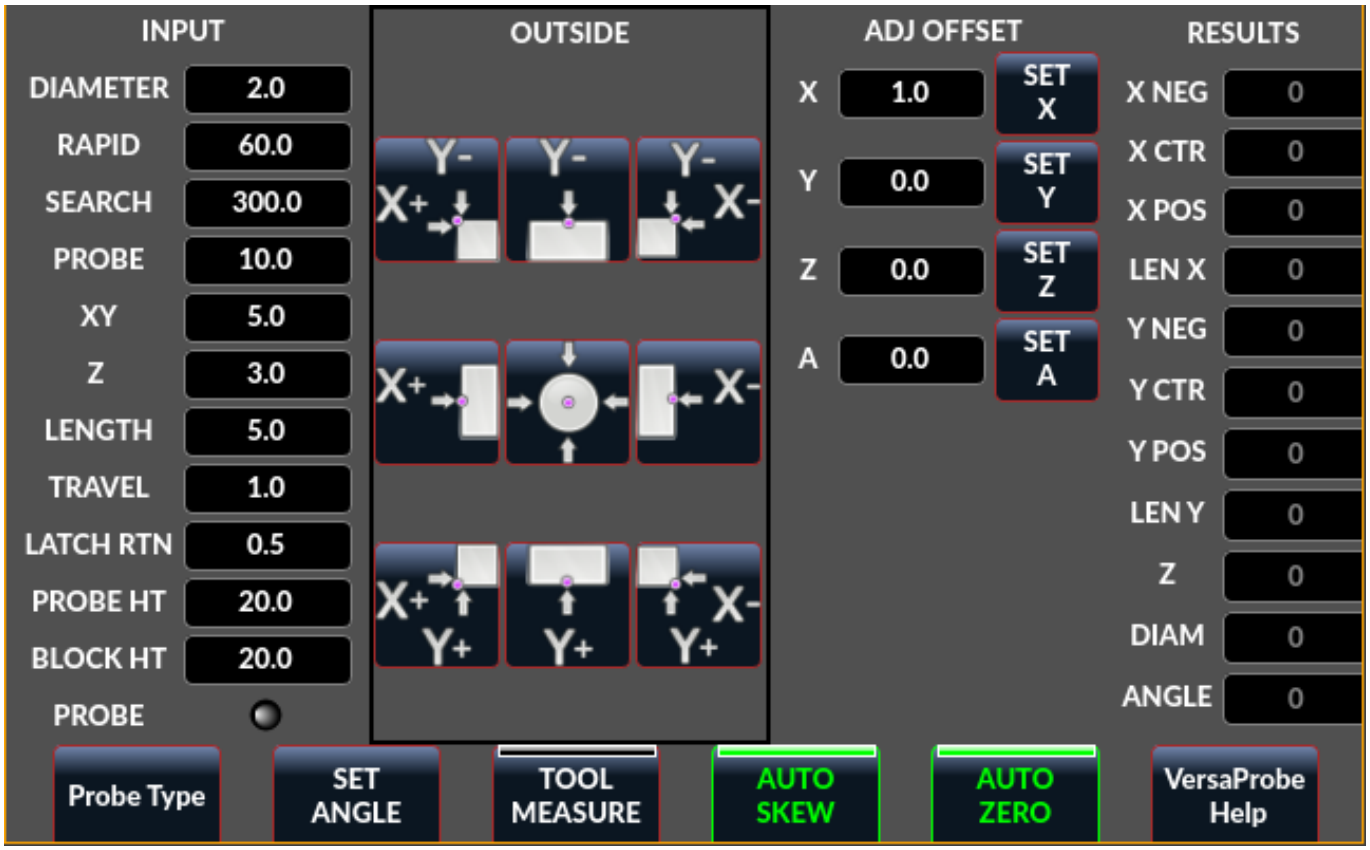


Figura 12.96: QtVCP `VersaProbe`: Mill Probing Widget

Widget for **probing on a mill**. Used by the *QtDragon* screen.

12.7.3. Dialog Widgets

Dialogs are used to **present or request immediately required information** in a focused way.

The typical used dialogs can be loaded using the `ScreenOptions` *widget*.

You can also add them directly to the *UI* - but each dialog must have a unique launch name or you will see multiple dialogs displayed, one after another.

Use dialogs from Python Code You can show dialogs directly with *Python code*, but a safer way is to **use `STATUS` messages** to request the dialog to launch and to return the gathered information.

- **Register to `STATUS` channel**:
  To set this up, first register to catch the `general` message from `STATUS`:

```
STATUS.connect('general',self.return_value)
```

- **Add a function to call a dialog**:
  This function must *build a message `dict` to send to the dialog*.
  This message will be passed back in the general message with the addition of the *`return` variable*.
  It is possible to add *extra user information* to the message. The dialog will ignore these and pass them back.

NAME
Launches code name of dialog to show.

ID
A unique id so we process only a dialog that we requested.

TITLE
El título a usar en el diálogo.

```
def show_dialog(self):
    mess = {'NAME':'ENTRY','ID':'__test1__',
            'TITLE':'Test Entry'}
    ACTION.CALL_DIALOG, mess)
```

- **Add a callback function that processes the general message**:
  Keep in mind this function will *get all general messages* so the `dict` keynames are not guaranteed to be there. Using the `.get()` function and/or using `try`/`except` is advisable. This function should:
  - check the name and id is the same as we sent,
  - then extract the return value and any user variables.

```
# process the STATUS return message
def return_value(self, w, message):
    rtn = message.get('RETURN')
    code = bool(message.get('ID') == '__test1__')
    name = bool(message.get('NAME') == 'ENTRY')
    if code and name and not rtn is None:
        print('Entry return value from {} = {}'.format(code, rtn))
```

12.7.3.1. `LcncDialog` - General Message Dialog Widget

This is a **general message dialog widget**.

If there is a Focus Overlay widget present, it can signal it to display.

If the sound library is set up it can *play sounds*.

There are *options* that can be set when requesting a dialog, these would be added to the message `dict`.

TITLE
Título de la ventana de diálogo.

MESSAGE
Title message text in bold.

MORE
Standard text under the heading.

DETAILS
Texto inicial oculto.

TYPE *(OK|YESNO|OKCANCEL)* , ICON (QUESTION|INFO|CRITICAL|WARNING) , PINNAME
Aún no implementado.

FOCUSTEXT *(overlay text|None)*
Text to display if focus overlay is used. Use `None` for no text.

FOCUSCOLOR (QColor(_R, G, B, A_))
Color to use if focus overlay is used.

PLAYALERT
Sound to play if sound is available, i.e., `SPEAK` <*spoken_message*> .

When using `STATUS` 's `request-dialog` function, the *default launch name* is **`MESSAGE`**.

It is based on PyQt's *QMessagebox*.

12.7.3.2. `ToolDialog` - Manual Tool Change Dialog Widget

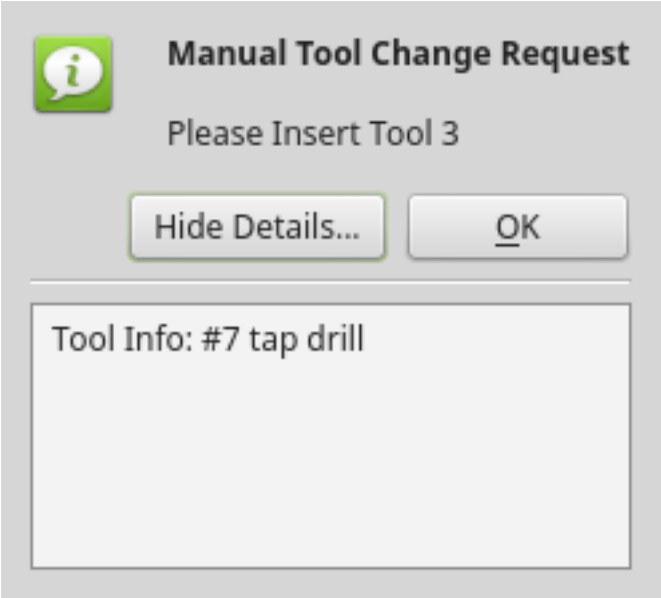


Figura 12.97: QtVCP `ToolDialog`: Manual Tool Change Dialog

This is used as a **manual tool change prompt**.

It has *HAL pins to connect to the machine controller*. The pins are named the same as the original AXIS manual tool prompt and works the same.

The tool change dialog *can only be launched by HAL pins*.

If there is a Focus Overlay widget present, it will signal it to display.

It is based on PyQt's *QMessagebox*.

12.7.3.3. `FileDialog` - Load and Save File Chooser Dialog Widget

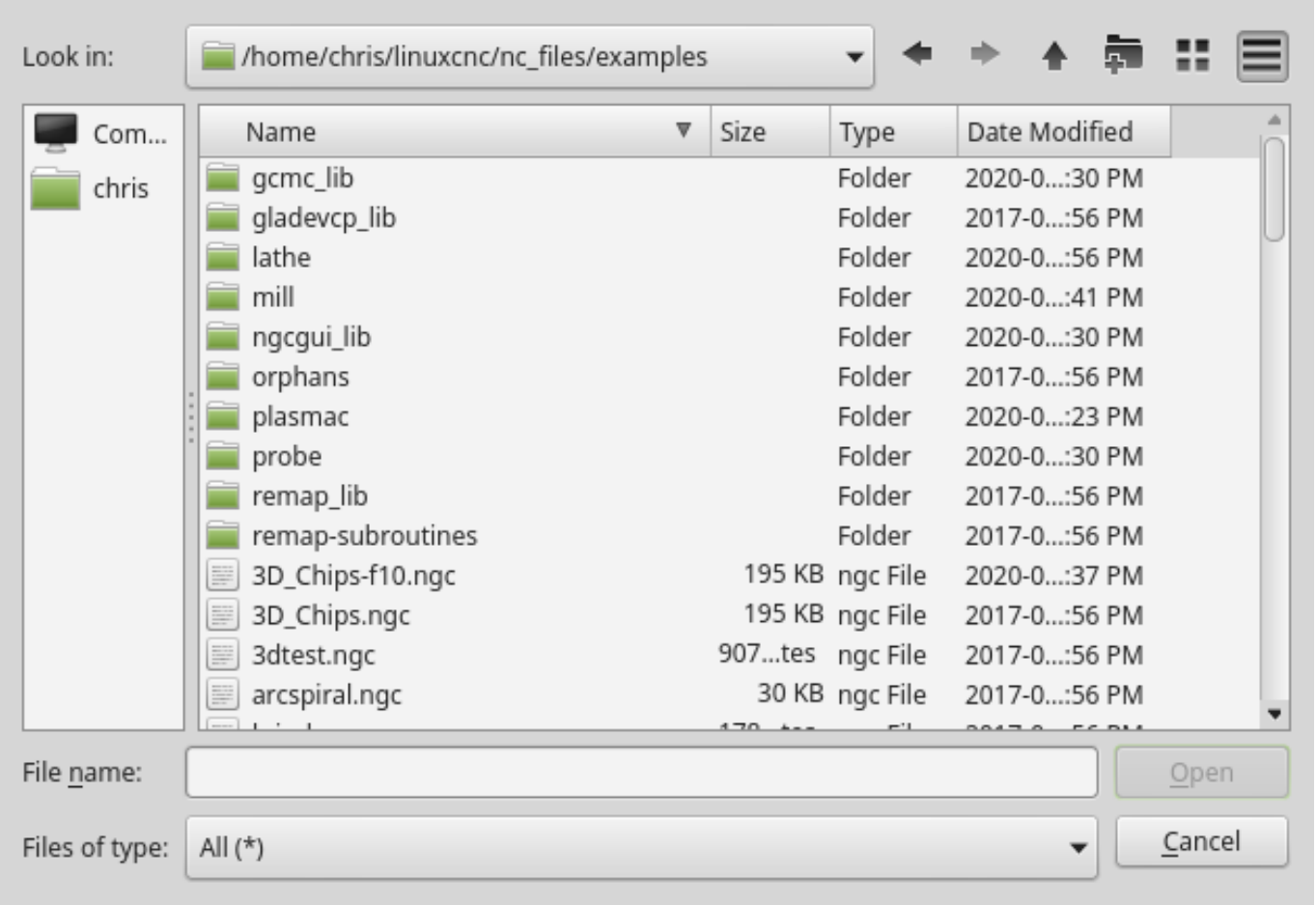


Figura 12.98: QtVCP `FileDialog`: Load and Save File Chooser Widget

Se usa para **cargar archivos de código G**.

If there is a Focus Overlay widget present, it will signal it to display.

When using `STATUS` 's `request-dialog` function, the default launch names are **`LOAD`** or **`SAVE`**.

There are *options* that can be set when requesting a dialog, these would be added to the message dict:

`EXTENSIONS`, **`FILENAME`**, **`DIRECTORY`**

An example Python call, for a *load dialog*:

```
mess = {'NAME':'LOAD','ID':'_MY_DIALOG_',
            'TITLE':'Load Some text File',
            'FILENAME':'~/linuxcnc/nc_files/someprogram.txt',
            'EXTENSIONS':'Text Files (*.txt);;ALL Files (*.*)'
            }
ACTION.CALL_DIALOG(mess)
```

And for a *save dialog*

```
mess = {'NAME':'SAVE','ID':'_MY_DIALOG_',
            'TITLE':'Save Some text File',
            'FILENAME':'~/linuxcnc/nc_files/someprogram.txt',
            'EXTENSIONS':'Text Files (*.txt);;ALL Files (*.*)'
            }
ACTION.CALL_DIALOG(mess)
```

It is based on PyQt's *QMessagebox*.

12.7.3.4. `OriginOffsetDialog` - Origin Offset Setting Dialog Widget

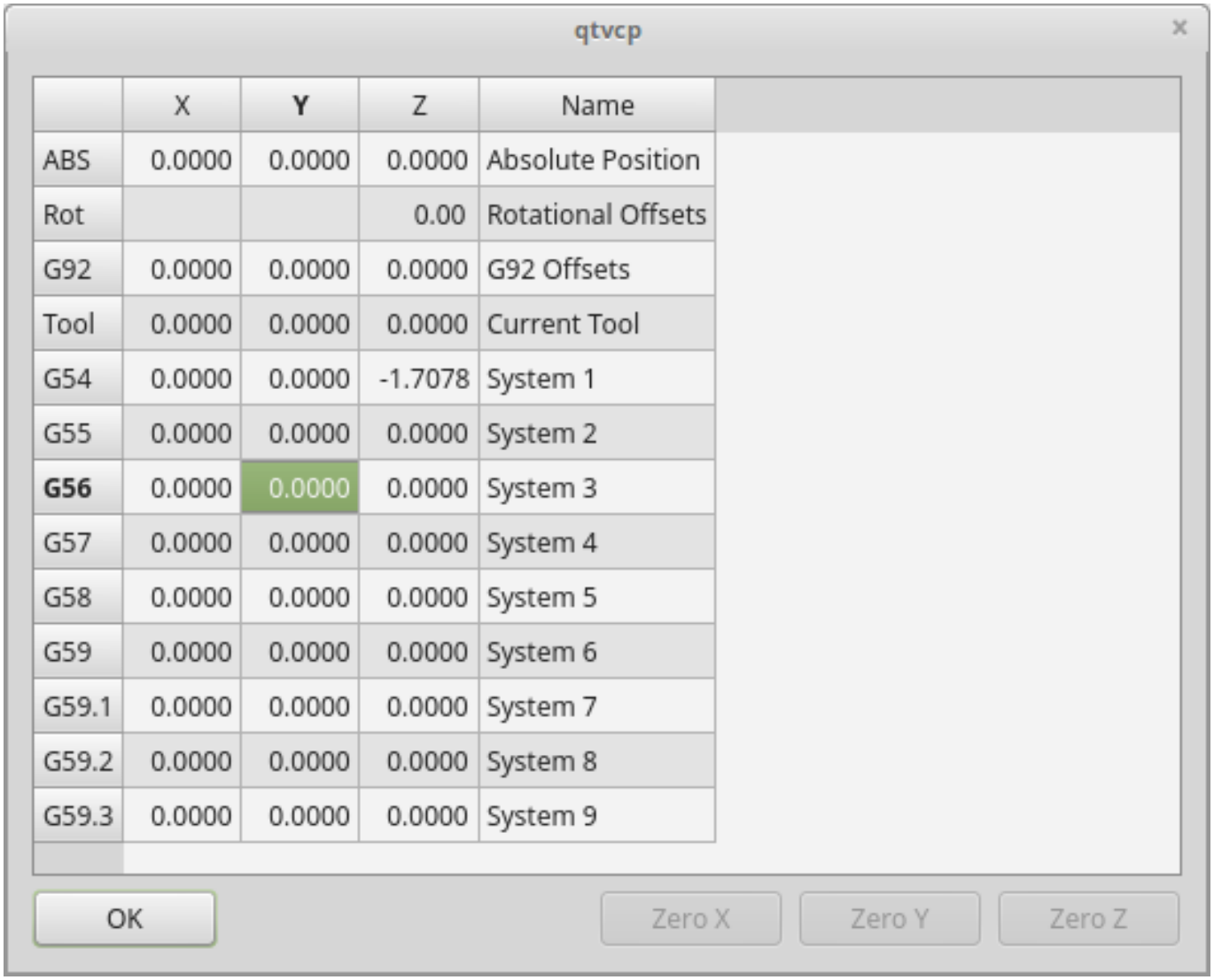


Figura 12.99: QtVCP `OriginOffsetDialog`: Origin Offset Setting Widget

This widget allows one to **modify User System origin offsets directly** in a dialog form.
If there is an Focus Overlay widget present, it will signal it to display.
When using `STATUS` 's `request-dialog` function, the default launch name is **`ORIGINOFFSET`**.
It is based on PyQt's *QDialog*.

12.7.3.5. `ToolOffsetDialog` - Tool Offset Setting Dialog Widget

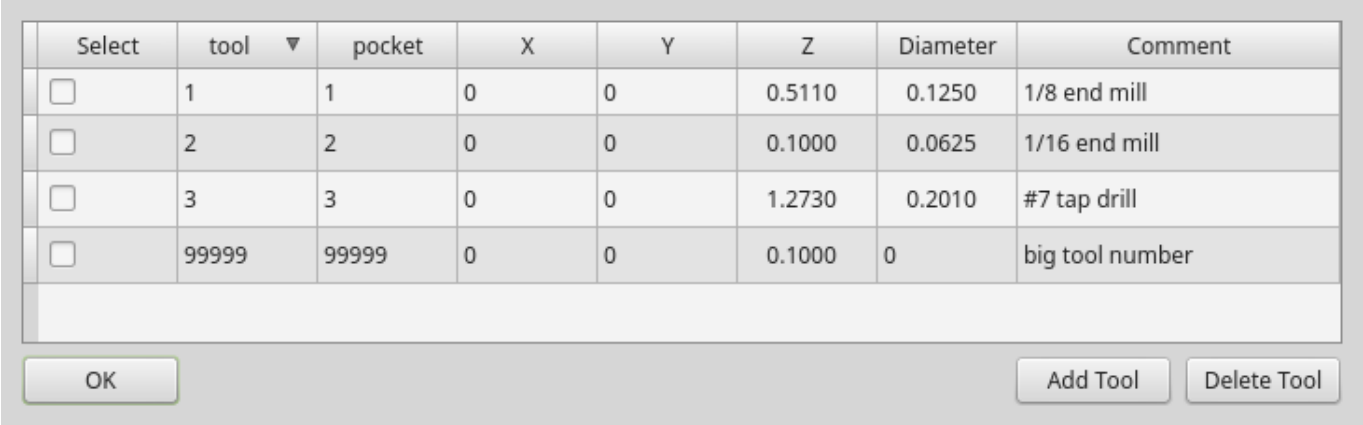


Figura 12.100: QtVCP `ToolOffsetDialog`: Tool Offset Setting Dialog Widget

This widget allows one to **modify Tool offsets directly** in a dialog form.
If there is an Focus Overlay widget present, it will signal it to display.
When using `STATUS` 's `request-dialog` function, the default launch name is **`TOOLOFFSET`**.
It is based on PyQt's *QDialog*.

12.7.3.6. `ToolChooserDialog` - Tool Chooser Dialog Widget

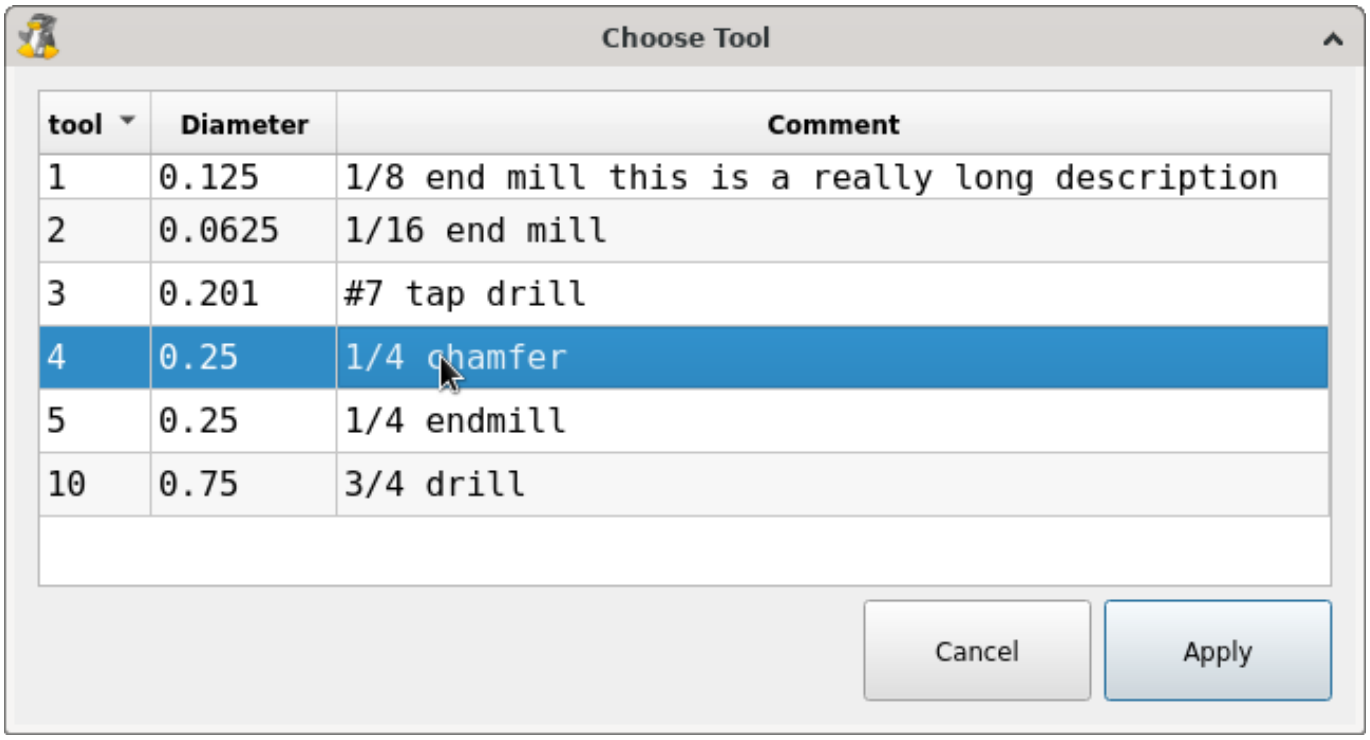


Figura 12.101: QtVCP `ToolChooserDialog`: Tool Chooser Dialog Widget

This widget allows the operator to select one of the tools defined in the tool table. If a tool is selected and *Apply* is pressed or the tool is double-clicked, the dialog will return the tool number selected. This can be used in conjunction with the `OperatorValueLine` widget to create a tool change widget, for example.

If there is an Focus Overlay widget present, it will signal it to display.

When using `STATUS` 's `request-dialog` function, the default launch name is **`TOOLCHOOSER`**.

It is based on PyQt's *QDialog*.

12.7.3.7. `MachineLog` - Machine Events Journal Display Widget

This widget displays various **log event messages** that have been output by the system during the current session. This includes informational messages as well as errors.

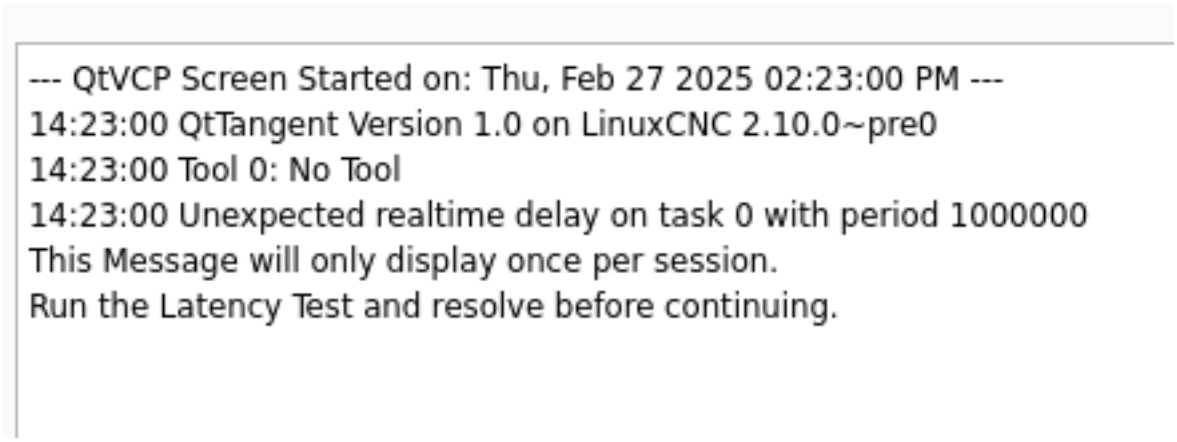


Figura 12.102: QtVCP `MachineLog`: Machine Events Log in `machine_log` (plain) mode

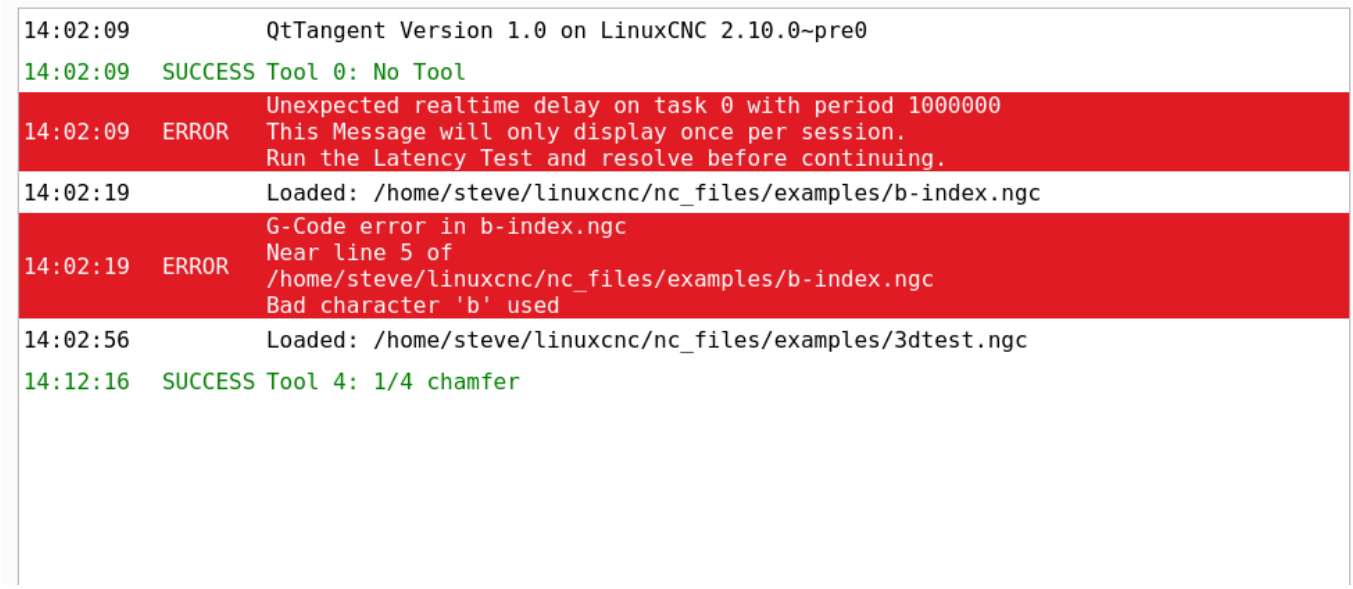


Figura 12.103: QtVCP `MachineLog`: Machine Events Log in `machine_log_severity` mode

Two distinct types of logs may be displayed:

- machine log (plain text or severity highlighted)
- integrator log (plain text only)

The type of log shown by the widget is controlled by the option properties of the widget. By selecting `machine_log_option` or `integrator_log_option` the appropriate log will be displayed. These options will display plain styled logs in a Qt `QTextEdit` widget.

Additionally, there is a `machine_log_severity_option` property that may be chosen that will display the machine log in a variety of colors depending on the severity of the message, by using a `QTableWidget`. The colors may be configured with the properties of the widget.

Severity is conveyed via the `option` value sent along with the `STATUS` signal called `update-machine-log`. The `option` parameter is a comma-delimited list, containing typically

``` text = *an error has occurred.* STATUS.emit(*update-machine-log*, text, *TIME,ERROR*) ```

The log may be cleared by calling the `clear()` method of the widget.

12.7.3.8. `MacroTabDialog` - Macro Launch Dialog Widget

This is a dialog to **display the macrotab widget**.

`MacroTab` displays a *choice of macro programs to run using icons*.

If there is a Focus Overlay widget present, it will signal it to display.

When using ``STATUS``'s `request-dialog` function, the default launch name is **`MACROTAB`**.

It is based on PyQt's *QDialog*.

12.7.3.9. `CamViewDialog` - WebCam Part Alignment Dialog Widget

This is a dialog to **display the CamView widget for Webcam part alignment**.

When using ``STATUS``'s `request-dialog` function, the default launch name is **`CAMVIEW`**.

It is based on PyQt's *QDialog*.

12.7.3.10. `EntryDialog` - Edit Line Dialog Widget

This is a dialog to **display an edit line for information entry**, such as origin offset.

It returns the entry via `STATUS` messages using a Python `DICT`.

The `DICT` contains at minimum, the name of the dialog requested and an ID code.

When using ``STATUS``'s `request-dialog` function, the default launch name is **`ENTRY`**.

It is based on PyQt's *QDialog*.

12.7.3.11. `CalculatorDialog` - Calculator Dialog Widget

This is a dialog to **display a calculator for numeric entry**, such as origin offset, spindle RPM, etc. It is primarily intended for touchscreen use, but it has support for physical keyboard input as well.
```

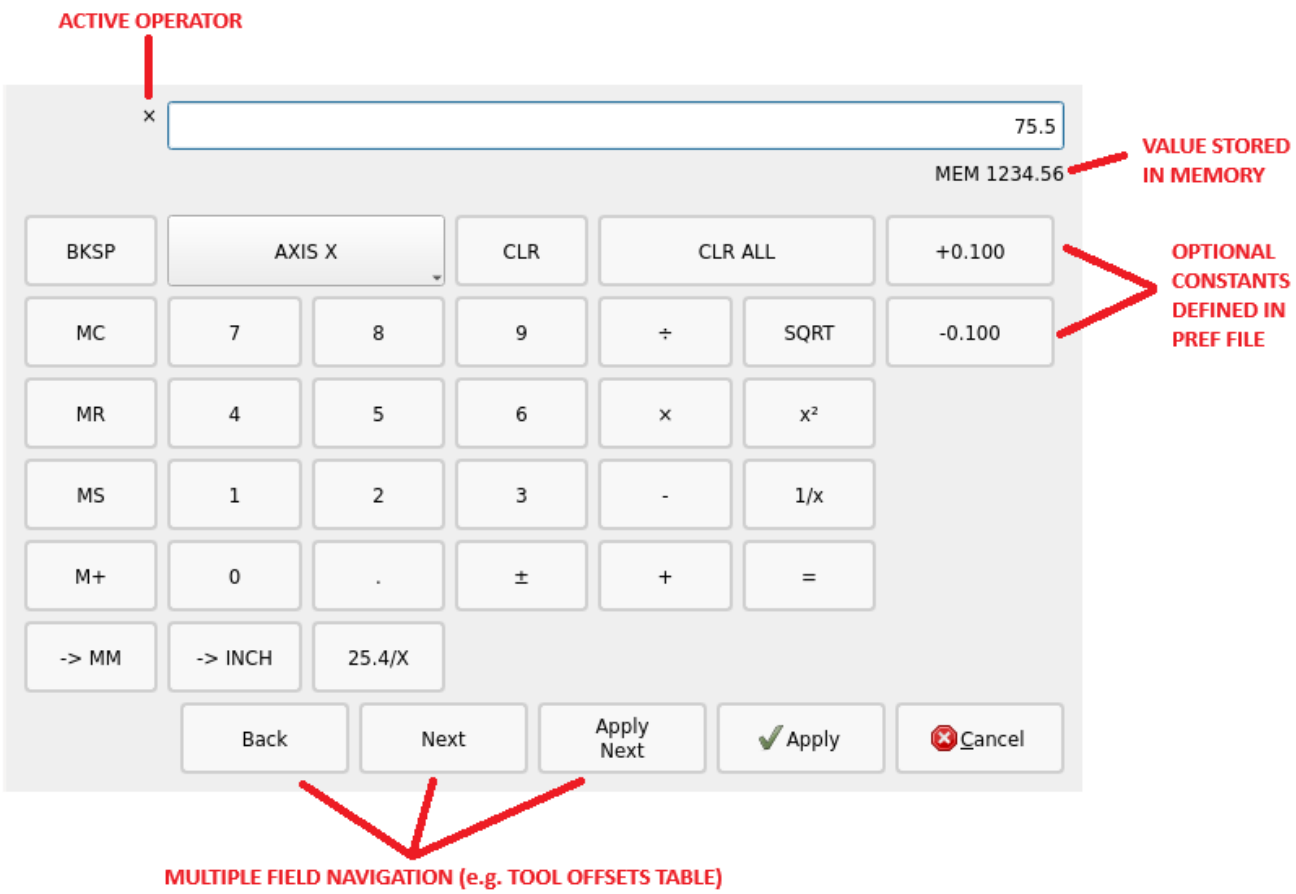


Figura 12.104: QtVCP `CalculatorDialog`: Calculator Dialog Widget

It returns the entry via `STATUS` messages using a Python `DICT`.

The `DICT` contains at minimum, the name of the dialog requested and an ID code.

When using ``STATUS``'s `request-dialog` function, the default launch name is **`CALCULATOR`**.

It is based on PyQt's *QDialog*.

In the `CALCULATOR` section of the preferences file the following options may be set:

- `constValuesList` - A comma-delimited list of common values you might enter, that will appear on a dedicated row of buttons at the bottom of the calculator. e.g. setting to `0.100, -0.100` would provide two buttons for +0.100 and -0.100 which are commonly used when edge-finding on inch mills. Up to six (6) values may be entered, beyond that the list will be truncated. Values must be valid floating point or integer.
- `onShowBehavior` - A list of optional behaviors that will be triggered when the calculator dialog is shown. Each option must be separated by a comma.
  - `CLEAR_ALL` to issue a **Clear All** each time the calculator is shown. This will clear any previously entered values from the last time the calculator was used and open with the display value set to `0`
  - `FORCE_FOCUS` will force the focus to the calculator input field when the widget is shown. This will allow a physical keyboard to provide input to the widget properly without additional clicks. Also, it has the side-effect of selecting the current value, such that typing from a physical keyboard will replace the existing value unless the text selection is changed.

- `acceptOnReturnKey` - If set to `True`, the calculator will accept the current value and close the dialog when the keyboard return/enter key is pressed. If set to `False`, the return key will be ignored and the **Apply** button must be clicked. For cases where the **Apply Next** button is available, the return key will perform this action instead, and the dialog will remain open.

While the intent of this widget is to provide a touchscreen-friendly interface, it is possible to use a physical keyboard to enter values, typically via a numeric keypad. The keys perform mostly as one would expect, but a few special key functions also exist:

- the **Enter** or **Return** key is equivalent to the **Equal** (=) operator in the cases where a calculation is pending. Otherwise it will perform the **Apply** function if it is enabled via the `acceptOnReturnKey` preference.
- the **minus key** (-) will toggle the sign of the current number in the calculator display when it is hit twice in a row. Otherwise when hit only once it will perform a subtraction operator as expected.
- **Alt+Left Arrow** will perform **Back** function, moving to the previous field for cases where this is supported.
- **Alt+Right Arrow** will perform a **Next** function, moving to the next field for cases where this is supported.
- **Alt+Backspace** will cancel out of the calculator and not provide a return value to the calling widget.

12.7.3.12. `RunFromLine` - Run-From-Line Dialog Widget



Figura 12.105: QtVCP `RunFromLine`: Run-From-Line Dialog Widget

Dialog to **preset spindle settings before running a program from a specific line**.

It is based on PyQt's *QDialog*.

12.7.3.13. `VersaProbeDialog` - Part Touch Probing Dialog Widget

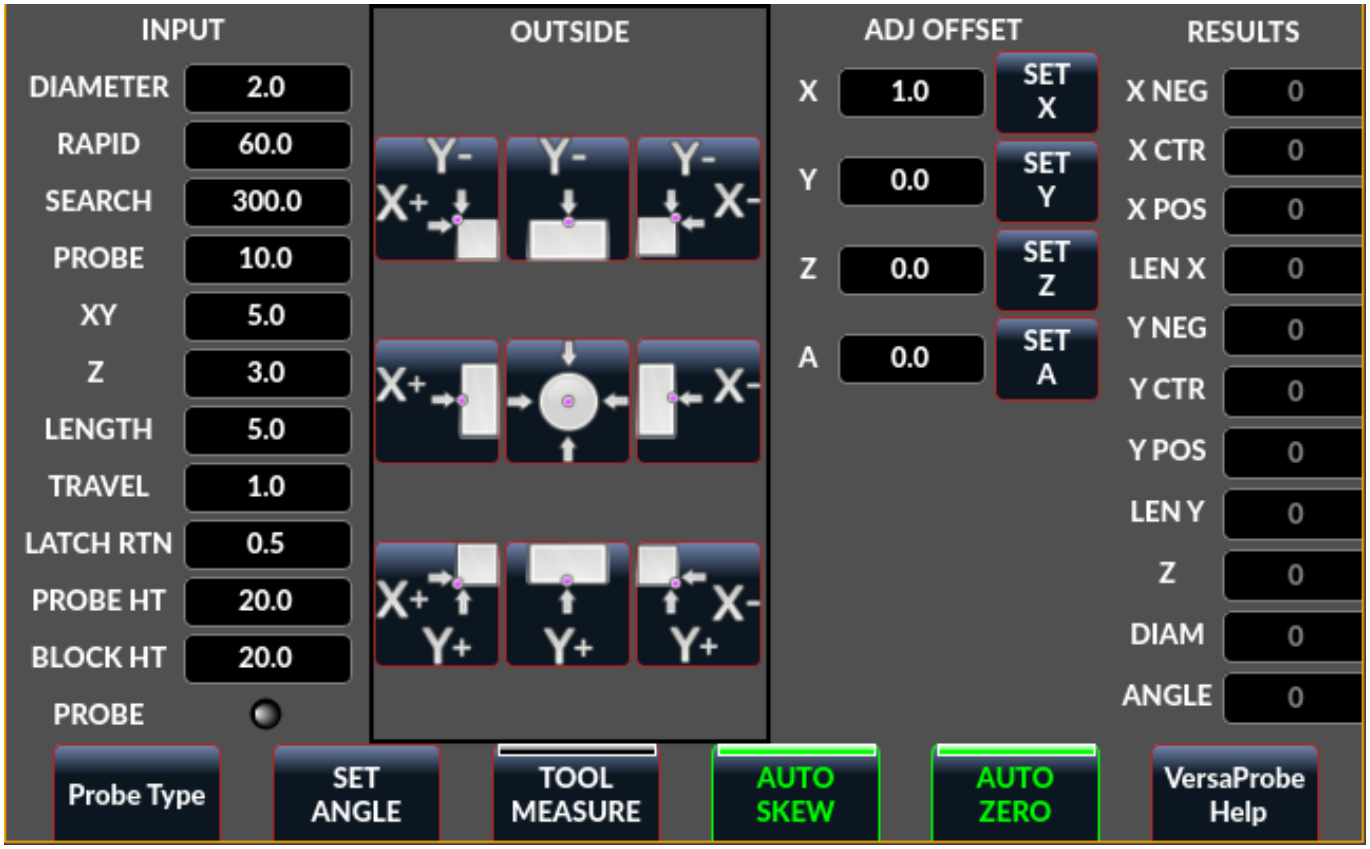


Figura 12.106: QtVCP `VersaProbeDialog`: Part Touch Probing Dialog Widget

This is a dialog to display a **part probing screen based on Verser Probe v2**.
It is based on PyQt's *QDialog*.

12.7.3.14. `MachineLogDialog` - Machine and Debugging Logs Dialog Widget

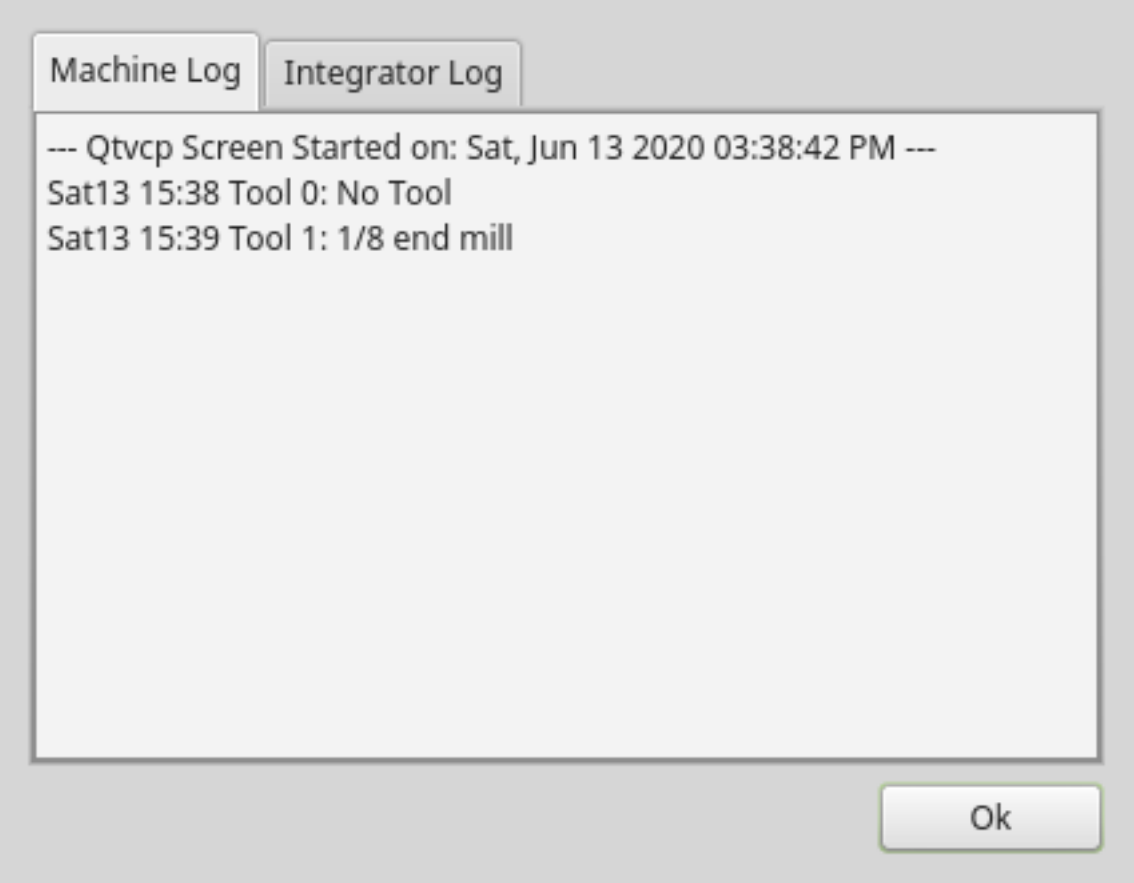


Figura 12.107: QtVCP `MachineLogDialog`: Machine and Debugging Logs Dialog Widget

This is a dialog to **display the machine log and QtVCP's debugging log**.
It is based on PyQt's *QDialog*.

12.7.4. Other Widgets

Other available widgets:

12.7.4.1. `NurbsEditor` - NURBS Editing Widget

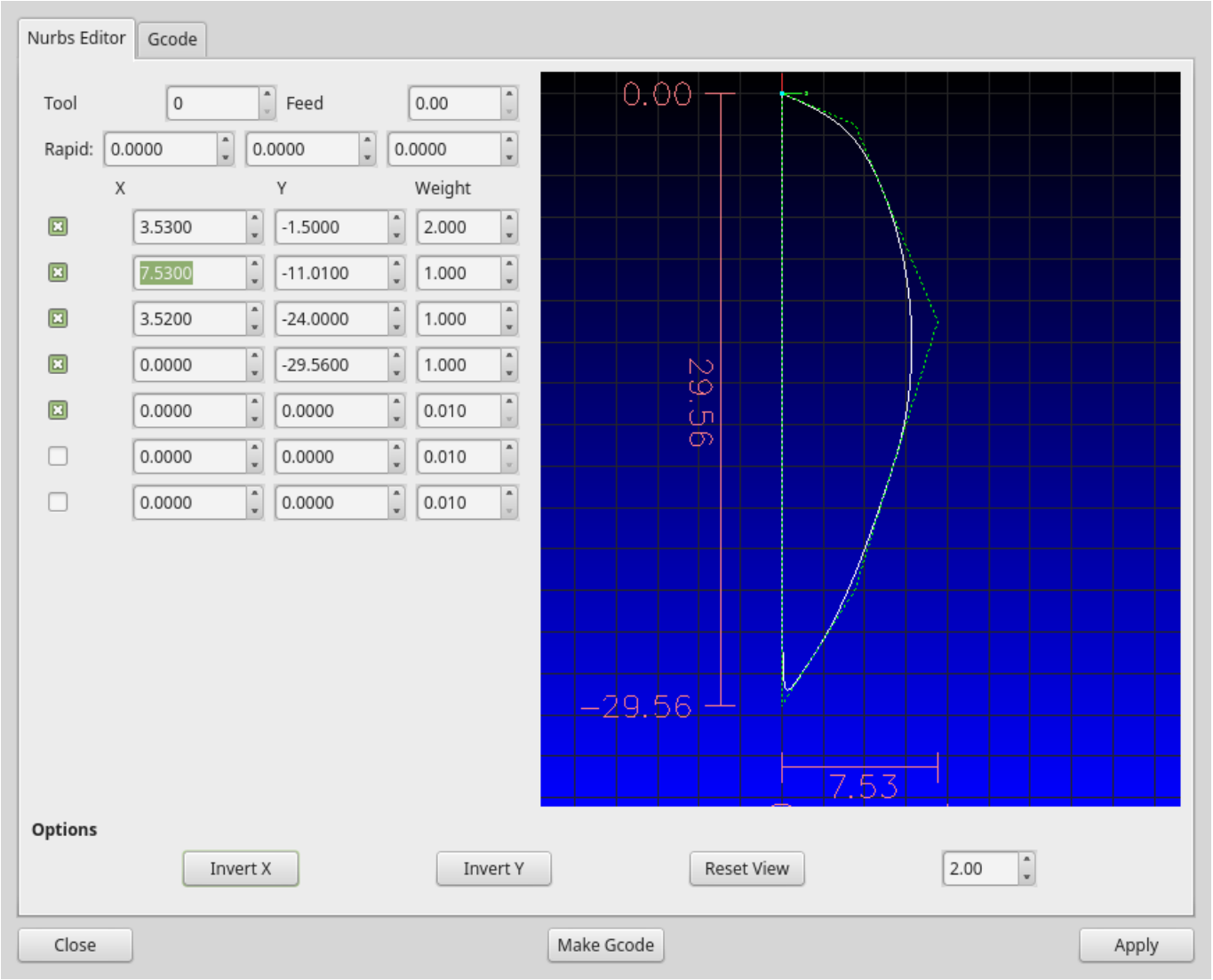


Figura 12.108: QtVCP `NurbsEditor`: NURBS Editing Widget

The Nurbs editor allows you to **manipulate a NURBS based geometry** on screen and then **convert NURBS to G-code**.

You can edit the G-code on screen and then send it to LinuxCNC.

It is based on PyQt's *QDialog*.

12.7.4.2. `JoyPad` - 5 button D-pad Widget

Es la clase base para el widget `HAL Pad`.

This widget looks and acts like a **5 button D-pad, with a LED like indicators in a ring**.

You can put text or icons in each of the button positions.

You can *connect to output signals* when the buttons are pressed.

There are also *input slots* to change the color of the indicator(s).

ENUMS There are **enumerated constants used to reference indicator positions**.
They are used in Qt Designer editor's property editor or in Python code.

NONE , LEFT, L , RIGHT, R , CENTER, C , TOP, T , BOTTOM, B , LEFTRIGHT, X , TOPBOTTOM, A
For Python handler code, you use the widget name in Qt Designer plus the reference constant:

```
self.w.joypadname.set_highlight(self.w.joypadname.LEFT)
```

Useful Override-able Functions

```
def _pressedOutput(self, btncode):
    self.joy_btn_pressed.emit(btncode)
    self['''.format(btncode.lower())].emit(True)

def _releasedOutput(self, btncode):
    self.joy_btn_released.emit(btncode)
    self['joy_{}_pressed'.format(btncode.lower())].emit(False)
```

As coded these function *issue (emit) PyQt5 signals (joy_btn_pressed and joy*<letter>_pressed) for the any button pressed or released_.

Signal *joy_btn_pressed* outputs a string code for the button.
Signal *joy_<letter>_pressed* outputs a bool value.

You could override the functions to do something else if making a custom widget:

Callable Functions

reset_highlight()
Clears the highlight indicator.

set_highlight(_button_, state=_True_)
Set the highlight indicator in position `button` to state `state`.
You can use *strings letters* (`LRCTBXA`) or *position* `ENUMS` for the button argument.

set_button_icon(_button_, _pixmap_)
Sets the button's icon pixmap.

set_button_text(_button_, _text_)
Sets the button's icon text.

set_tooltip(_button_, _text_)
Sets the buttons pop-up tooltip descriptive text.

setLight(_state_)
Sets the highlight indicator to the `True` color or `False` color.
The `set_highlight()` function must be used prior to set the indicator to use.

Signals These signals will be **sent when buttons are pressed**.

They can be connected to in Qt Designer editor or Python code.

The first two output a string that indicates the button pressed:

joy_btn_pressed *(string)* , joy_btn_released *(string)* , joy_l_pressed *(bool)* , joy_l_released *(boo*

They are based on PyQt's *Signal* (`QtCore.pyqtSignal()`)

Slots Slots can be connected to in Qt Designer editor or Python code:

`set_colorStateTrue()` , **`set_colorStateFalse()`** , **`set_colorState(_bool_)`** , **`set_true_color(_str_)`**

Properties These can be set in stylesheets or Python code:

`highlightPosition`
: Establece la posición del indicador.

`setColorState`
: Select the color state of the indicator.

`left_image_path` , **`right_image_path`** , **`center_image_path`** , **`top_image_path`** , **`bottom_image_path`**

: A file path or resource path to an image to display in the described button location.
If the reset button is pressed in Qt Designer editor property, the image will not be displayed (allowing optionally text).

`left_text` , **`right_text`** , **`center_text`** , **`top_text`** , **`bottom_text`**
: A text string to be displayed in the described button location.
If left blank an image can be designated to be displayed.

`true_color` , **`false_color`**
: Color selection for the center LED ring to be displayed, when the `BASENAME.light.center` *HAL pin* is `True` or `False`.

`text_color`
: Color selection for the button text.

`button_font`
: Font selection for the button text.

The above properties could be set in:

- **Stylesheets**:
  You would usually use the Qt Designer widget name with *# prefix* to set individual widget properties, otherwise you would use the `JoyPad` *class name* to set all `JoyPad` widgets the same:

```
#joypadname{
  qproperty-true_color: #000;
  qproperty-false_color: #444;
}
```

- **In Python handler code**:

```
self.w.joypadename.setProperty('true_color','green')
self.w.joypadename.setProperty('false_color','red')
```

12.7.4.3. *WebWidget*

This widget will create a html/pdf viewing page using the QtWebKit or QtWebEngine libraries. The newer QtWebEngine is preferred if both are on the system.
If the QtWebEngine library is used with the Qt Designer editor, a placeholder QWidget will show in Qesigner. This will be replaced with the QtWebEngine widget at run time.

12.7.5. BaseClass/Mixin Widgets

These widgets are used to **combine different properties and behaviours into other widgets**.

You will see them as a collapsible header in the Qt Designer properties column.

12.7.5.1. `IndicatedPushButtons`

This class **modifies `QPushButton` behaviour**.

`indicator_option` *puts a* LED *on the top of the button*.

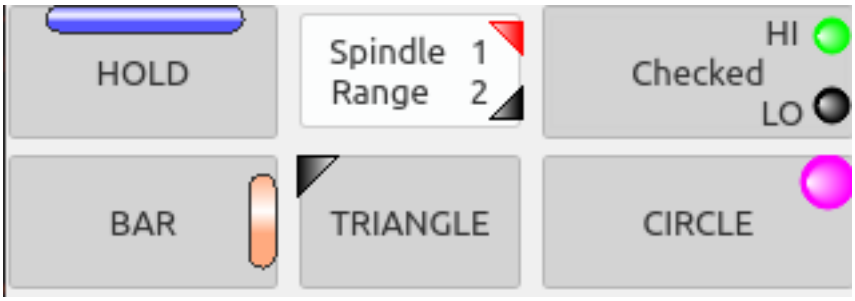


Figura 12.109: QtVCP `PushButton`: IndicatedPushButton Button, LED Indicator Option

It can be a *triangle*, *circle*, *top bar*, or *side bar*.
The *triangle* and *circle* LEDs can display double vertical LEDs optionally each with it's own HAL pin.
+ The *size* and *position* can be adjusted.

It will indicate:

- the **current state of the button**, or
- el **estado de un pin HAL**, o
- **estado de LinuxCNC**.

Properties These properties are available to customize the indicator (not all are applicable to every LED shape):

`doubleIndicator`
: With triangle or round LEDs add a second vertical LED.

`on_color` , **`off_color`** , **`flashIndicator`**
: When the indicator is true, flash the LED on and off.

`flashRate`
: Tasa de flasheo

`indicator_size`
: Size of triangle LED

`circle_diameter`
: Diameter of round LED

`shape_option`
: 0-4 LED shape type

right_edge_offset
: Space from right edge

top_edge_offset
: Space from top edge

height_fraction
: Used for bar Leds

width_fraction
: Used for bar Leds

corner_radius
: Indicador de radio de esquina.

The LED indicator color can be defined in a *stylesheet* with the following code added to the `.qss` file:

```
Indicated_PushButton{
  qproperty-on_color: #000;
  qproperty-off_color: #444;
}
```

Or for a particular button:

```
Indicated_PushButton #button_estop{
  qproperty-on_color: black;
  qproperty-off_color: yellow;
}
```

Opciones `IndicatedPushButton` have **exclusive options**:

indicator_HAL_pin_option
: Adds a `halpin`, named `<buttonname>-led` that controls the button indicator state.

indicator_status_option
: Makes the LED indicate the state of these selectable LinuxCNC status:

- *Is Estopped*
- *Is On*
- *All Homed*
- *Is Joint Homed*
- *Idle*
- *Paused*
- *Flood*
- *Mist*
- *Block Delete*
- *Parada opcional*
- *Manual*
- *MDI*
- *Auto*
- *Spindle Stopped*
- *Spindle Forward*
- *Spindle Reverse*
- *On Limits*

Some `indicator_status_options` holds a property that can be used with a *stylesheet* to change the color of the button based on the state of the property in LinuxCNC.
Currently these status properties can be used to auto style buttons:

- `is_estopped_status` will toggle the `isEstop` property
- `is_on_status` will toggle the `isStateOn` property
- `is_manual_status`, `is_mdi_status`, `is_auto_status` will toggle the `isManual`, `isMDI`, `isAuto` properties.
- `is_homed_status` will toggle the *isAllHomed* property

Here is a sample stylesheet entry setting the background of mode button widgets when LinuxCNC is in that mode:

```
ActionButton[isManual=true] {
    background: red;
}
ActionButton[isMdi=true] {
    background: blue;
}
ActionButton[isAuto=true] {
    background: green;
}
```

Here is how you specify a particular widget by its objectName in Qt Designer:

```
ActionButton #estop button [isEstopped=false] {
    color: yellow;
}
```

Often, having the button disabled and enabled based on the state of LinuxCNC's motion controller is necessary.

There are several properties that can be selected to aid with this:

`isAllHomedSentive`, **`isOnSensitive`**, **`isIdleSensitive`**, **`isRunSensitive`**, **`isRunPausedSensitive`**, **`is`**

You can select multiple properties for combined requirements.

Choosing the **`checked_state_text_option`** allows a *checkable* button to *change the text based on its checked state*.

It uses the following properties to specify the text for each state:

`true_state_string`, **`false_state_string`**
\\n will be converted to a newline.

Puedes configurar/modificar esas hojas de estilo:

```
ActionButton #action_aux{
  qproperty-true_state_string: ”Air\\nOn”;
  qproperty-false_state_string: ”Air\\nOff”;
}
```

The **`python_command_option`** allow small snippets of Python code to be run from the push of a button, without having to edit the handler file. Though, it can call functions in the handler file.

Cuando se usan la propiedades de `command_string`.

true_python_cmd_string
A Python command that will be called when the button is toggled `True`.

false_python_cmd_string
A Python command that will be called when the button is toggled `False`.

Special capitalized words will give access to the following:

INSTANCE
Will give access to the widgets instances and handler functions.
E.g., `INSTANCE.my_handler_function_call(True)`

ACTION
Will give access to QtVCP's `ACTION` library.
E.g., `ACTION.TOGGLE_FLOOD()`

PROGRAM_LOADER
Will give access to QtVCP's `PROGRAM_LOADER` library.
E.g., `PROGRAM_LOADER.load_halshow()`

HAL
Will give access to HAL's Python module.
E.g., `HAL.set_p('motion.probe-input','1')`

12.7.6. Import-Only Widgets

These widgets are usually the **base class widget for other QtVCP widgets**.

They are *not available directly from the Qt Designer editor* but could be **imported and manually inserted**.

They could also be **subclassed** to make a similar widget with new features.

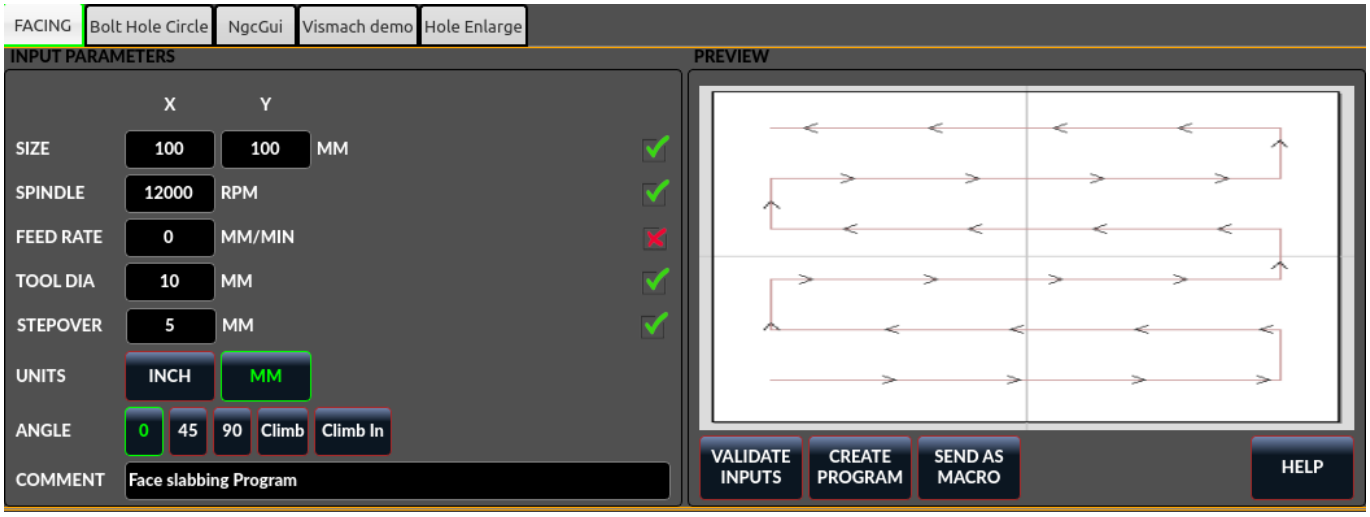
12.7.6.1. Auto Height

Widget for measuring two heights with a probe.
For setup.

12.7.6.2. G-code Utility

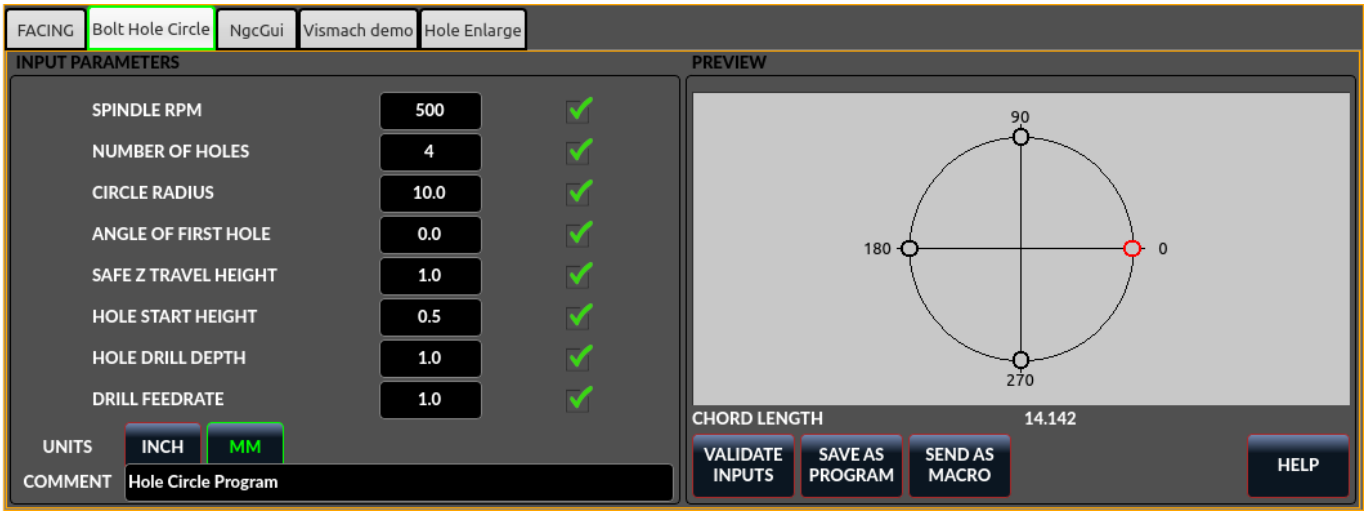
Widgets for performing common machining processes.

12.7.6.3. Facing



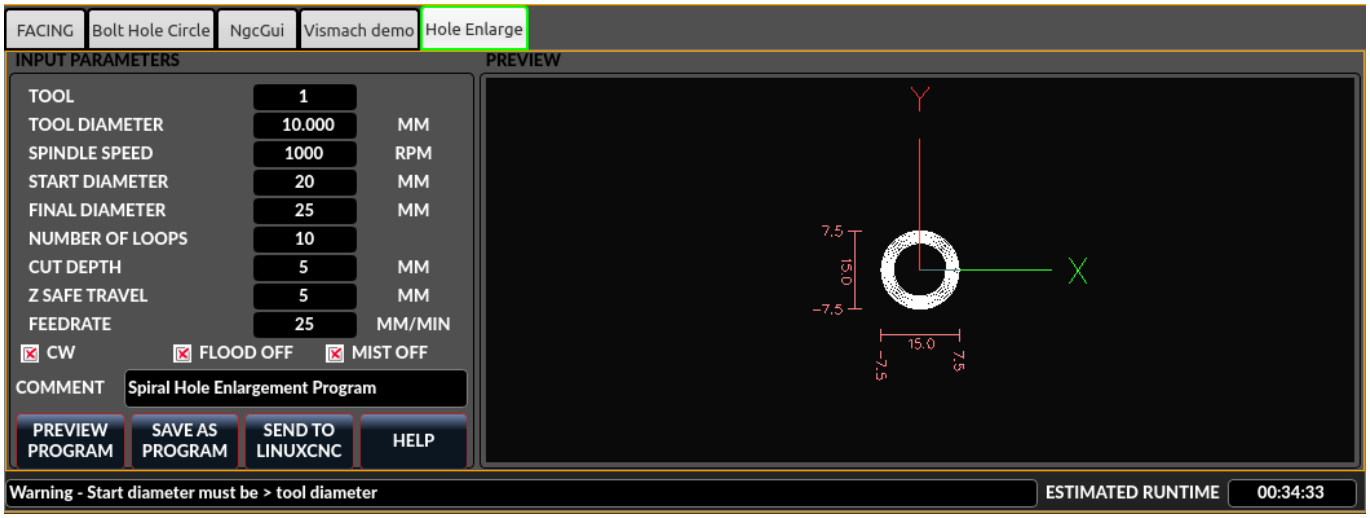
Slab or face a definable area with different strategies.

12.7.6.4. Hole Circle



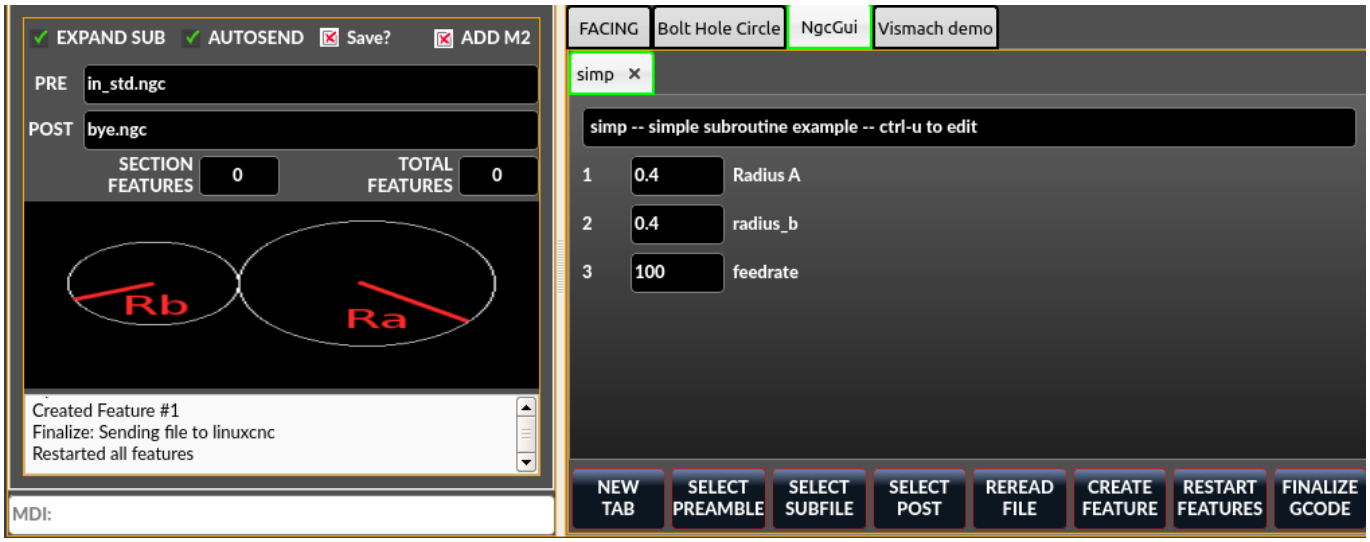
Drill multiple holes on a bolt hole circle.

12.7.6.5. Hole Enlarge



Use an end mill to enlarge a drilled hole.

12.7.6.6. Qt NGCGUI



QtVCP's version of NGC subroutine selector (Shown as used in QtDragon).

LinuxCNC needs to know where to look to run the subroutines.
If the subroutine calls other subroutines or custom M codes, those paths must be added too.

```
[RS274NGC]
SUBROUTINE_PATH = ~/linuxcnc/nc_files/examples/ngcgui_lib:~/linuxcnc/nc_files/examples/ ↩
    ngcgui_lib/utilitysubs
```

QtVCP needs to know where to open subroutines from.
You can also specify subroutines to be pre-opened in tabs.

```
[DISPLAY]
# NGCGUI subroutine path.
# This path must also be in [RS274NGC] SUBROUTINE_PATH
NGCGUI_SUBFILE_PATH = ~/linuxcnc/nc_files/examples/ngcgui_lib
```

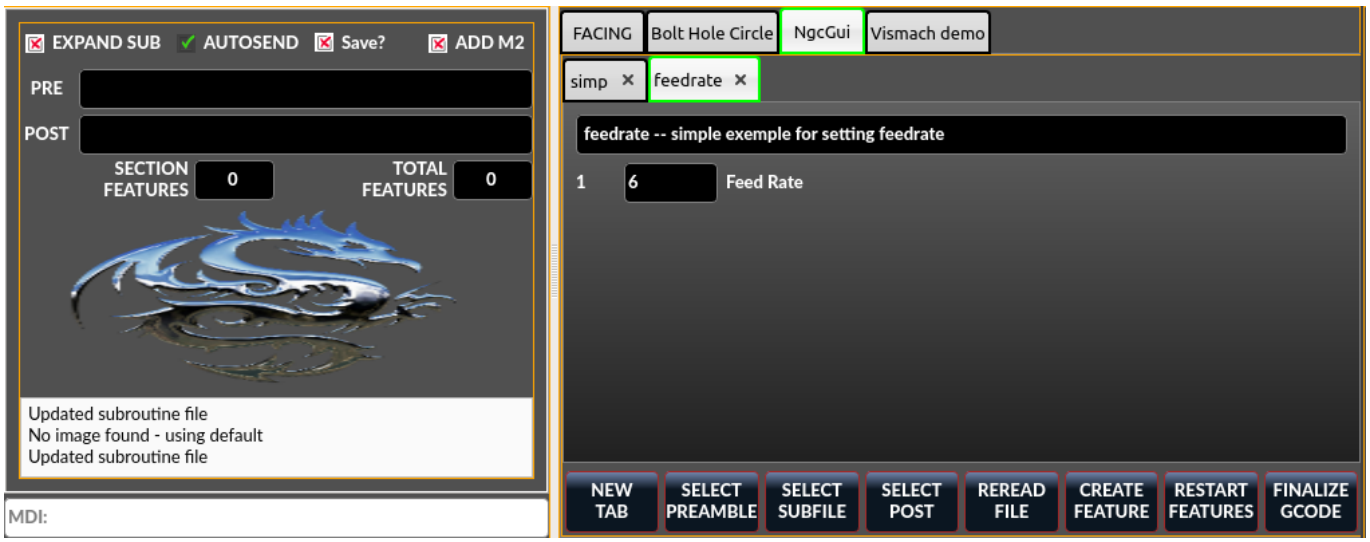
```
# pre selected programs tabs
# specify filenames only, files must be in the NGCGUI_SUBFILE_PATH
NGCGUI_SUBFILE = slot.ngc
NGCGUI_SUBFILE = qpocket.ngc
```

- *NEW TAB* - add new blank tab to NGCGUI
- *SELECT PREAMBLE* - select a file that add preamble G-code
- *SELECT SUBFILE* - select a NGCGUI subroutine file
- *SELECT POST* - select a file that add post G-code
- *REREAD FILE* - reload the subroutine file
- *CREATE FEATURE* - add feature to the list
- *RESTART FEATURE* - remove all features from the list
- *FINALIZE GCODE* - create the full G-code and send it to LinuxCNC/a file

You can create your own subroutines for use with NGCGUI. They must follow these rules:

- For creating a subroutine for use with NGCGUI, the filename and the subroutine name must be the same.
- The subroutine must be in a folder within LinuxCNC's INI designated search path.
- En la primera línea puede estar un comentario de tipo información:
- The subroutine must be surrounded by the sub and endsub tags.
- The variables used must be numbered variables and must not skip number.
- Comments and presets may be included.
- If an image file of the same name is in the folder, it will be shown.

```
(info: feedrate -- simple example for setting feedrate)
o<feedrate> sub
    #<feedrate>       = #1 (= 6 Feed Rate) ; comments in brackets will be shown in ngcui
    f#<feedrate>
o<feedrate> endsub
```



12.7.6.7. Qt PDF

Allows adding loadable PDFs to a screen.

12.7.6.8. Qt Vismach

Use this to build/add OpenGl simulated machines.

12.7.6.9. Hal Selection Box

This widget is combobox that will allows selection of a pin or signal on the system.

```
from qtvcp.widgets.hal_selectionbox import HALSelectionBox

    def buildComboBox(self):
        # combo box for HAL pin selection
        combobox = HALSelectionBox()
        combobox.setShowTypes([combobox.PINS,combobox.SIGNALS])
        combobox.setPinTypes([combobox.HAL_BIT], direction = [combobox.HAL_IN])
        combobox.setSignalTypes([combobox.HAL_BIT], driven = [False,True])
        combobox.hal_init()
        combobox.selectionUpdated.connect(lambda w: self.signalSelected(w))

    def signalSelected(self, sig):
        print('Watching:',sig)
```

Hay llamadas a función

```
# set the list of types to show from: PINS SIGNALS
combobox.setShowTypes([combobox.PINS])

# set the pin types to show: HAL_BIT,HAL_FLOAT,HAL_S32,HAL_U32
# and a list of directions: HAL_IN HAL_OUT
combobox.setPinTypes(types=[combobox.HAL_BIT], direction = [HAL_IN])

# set the signal types to show: HAL_BIT,HAL_FLOAT,HAL_S32,HAL_U32
# and a list of driven/undriven (by a connected pin) to show
combobox.setSignalTypes( types=[combobox.HAL_BIT], driven = [True,True])
```

12.8. QtVCP Libraries modules

Las bibliotecas son **módulos de Python preconstruidos que le dan características adicionales a QtVCP**. De esta manera, puede seleccionar qué características desea, sin embargo, no tiene que crear las comunes usted mismo.

12.8.1. Status

Status is a library that **sends GObject messages based on LinuxCNC's current state**. It is an *extension of GladeVCP's GStat object*.

It also has some functions to report status on such things as internal jog rate.

You *connect a function* call to the STATUS message you are interested in, and QtVCP will call this function when the message is sent from STATUS.

12.8.1.1. Utilización

- **Import `Status` modules**
  Agrega este código Python a tu sección de importación:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.core import Status
```

- **Instantiate `Status` module**
  Agrega este código Python a tu sección de instanciación:

```
STATUS = Status()
```

- **Connect to `STATUS` messages**
  Use *GObject syntax*.

12.8.1.2. Ejemplo

For example, you can catch machine on and off messages.

---

nota
The example below shows the *two common ways of connecting signals*, one of them *using lambda*. **`lambda`** is *used to strip off or manipulate arguments* from the status message before calling the function. You can see the difference in the called function signature: The one that uses lambda does not accept the status object - lambda did not pass it to the function.

- Place these commands into the `[INITIALIZE]` section of the Python handler file:

```
STATUS.connect('state-on', self.on_state_on)
STATUS.connect('state-off', lambda: w, self.on_state_off())
```

  In this example code, when LinuxCNC is in *"machine on" state* the function `self.on_state_on` will be called.
  When LinuxCNC is in *"machine off" state* the function `self.on_state_off` will be called.

- These would call functions that looks like these:

```
def on_state_on(self, status_object):
    print('LinuxCNC machine is on')
def on_state_off(self):
    print('LinuxCNC machine is off')
```

12.8.2. `Info`

`Info` is a library to **collect and filter data from the INI file**.

12.8.2.1. Available data and defaults

```
LINUXCNC_IS_RUNNING
LINUXCNC_VERSION
INIPATH
INI = linuxcnc.ini(INIPATH)
MDI_HISTORY_PATH = '~/.axis_mdi_history'
QTVCP_LOG_HISTORY_PATH = '~/qtvcp.log'
MACHINE_LOG_HISTORY_PATH = '~/.machine_log_history'
PREFERENCE_PATH = '~/.Preferences'
SUB_PATH = None
SUB_PATH_LIST = []
self.MACRO_PATH = None
MACRO_PATH_LIST = []
INI_MACROS = self.INI.findall("DISPLAY", "MACRO")

IMAGE_PATH = IMAGEDIR
LIB_PATH = os.path.join(HOME, "share","qtvcp")

PROGRAM_FILTERS = None
PARAMETER_FILE = None
MACHINE_IS_LATHE = False
MACHINE_IS_METRIC = False
MACHINE_UNIT_CONVERSION = 1
MACHINE_UNIT_CONVERSION_9 = [1]*9
TRAJ_COORDINATES =
JOINT_COUNT = self.INI.getint("KINS","JOINTS", fallback=0)
AVAILABLE_AXES = ['X','Y','Z']
AVAILABLE_JOINTS = [0,1,2]
GET_NAME_FROM_JOINT = {0:'X',1:'Y',2:'Z'}
GET_JOG_FROM_NAME = {'X':0,'Y':1,'Z':2}
NO_HOME_REQUIRED = False
HOME_ALL_FLAG
JOINT_TYPE = self.INI.getstring(section, "TYPE", fallback="LINEAR")
JOINT_SEQUENCE_LIST
JOINT_SYNC_LIST

JOG_INCREMENTS = None
ANGULAR_INCREMENTS = None
GRID_INCREMENTS

DEFAULT_LINEAR_JOG_VEL = 15 units per minute
MIN_LINEAR_JOG_VEL = 60 units per minute
MAX_LINEAR_JOG_VEL = 300 units per minute

DEFAULT_ANGULAR_JOG_VEL =
MIN_ANGULAR_JOG_VEL =
MAX_ANGULAR_JOG_VEL =

MAX_FEED_OVERRIDE =
MAX_TRAJ_VELOCITY =

AVAILABLE_SPINDLES = self.INI.getint("TRAJ", "SPINDLES", fallback=1)
DEFAULT_SPINDLE_0_SPEED = 200
MAX_SPINDLE_0_SPEED = 2500
MAX_SPINDLE_0_OVERRIDE = 100
MIN_SPINDLE_0_OVERRIDE = 50

MAX_FEED_OVERRIDE = 1.5
MAX_TRAJ_VELOCITY
```

12.8.2.2. User message dialog info

```
USRMESS_BOLDTEXT = self.INI.findall("DISPLAY", "MESSAGE_BOLDTEXT")
USRMESS_TEXT = self.INI.findall("DISPLAY", "MESSAGE_TEXT")
USRMESS_TYPE = self.INI.findall("DISPLAY", "MESSAGE_TYPE")
USRMESS_PINNAME = self.INI.findall("DISPLAY", "MESSAGE_PINNAME")
USRMESS_DETAILS = self.INI.findall("DISPLAY", "MESSAGE_DETAILS")
USRMESS_ICON = self.INI.findall("DISPLAY", "MESSAGE_ICON")
ZIPPED_USRMESS =

self.GLADEVCP = self.INI.find("DISPLAY", "GLADEVCP")
```

12.8.2.3. Información del programa embebido

```
TAB_NAMES = (self.INI.findall("DISPLAY", "EMBED_TAB_NAME")) or None
TAB_LOCATION = (self.INI.findall("DISPLAY", "EMBED_TAB_LOCATION")) or []
TAB_CMD = (self.INI.findall("DISPLAY", "EMBED_TAB_COMMAND")) or None
ZIPPED_TABS =

MDI_COMMAND_LIST =      (heading: [MDI_COMMAND_LIST], title: MDI_COMMAND")
TOOL_FILE_PATH =        (heading: [EMCIO], title:TOOL_TABLE)
POSTGUI_HALFILE_PATH =  (heading: [HAL], title: POSTGUI_HALFILE)
```

12.8.2.4. Helpers

There are some *helper functions* - mostly used for widget support:

get_error_safe_setting(_self_, _heading_, _detail_, default=_None_), convert_metric_to_mac

Get filter extensions in Qt format.

12.8.2.5. Utilización

- **Import `Info` module**
  Agrega este código Python a tu sección de importación:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.core import Info
```

- **Instantiate `Info` module**.
  Agrega este código Python a tu sección de instanciación:

```
###########################################
# **** INSTANTIATE LIBRARIES SECTION **** #
###########################################

INFO = Info()
```

- **Access `INFO` data** Use this general syntax:

```
home_state = INFO.NO_HOME_REQUIRED
if INFO.MACHINE_IS_METRIC is True:
    print('Metric based')
```

12.8.3. Action

Action la librería se usa para **ordenar al controlador de movimiento de LinuxCNC**.

It tries to hide incidental details and add convenience methods for developers.

12.8.3.1. Helpers

There are some **helper functions**, mostly used for this library's support:

get_jog_info (_num_) , jnum_check(_num_) , ensure_mode(_modes_) , open_filter_program(_filena

Open G-code filter program.

12.8.3.2. Utilización

- **Import Action module**
  Agrega este código Python a tu sección de importación:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.core import Action
```

- **Instantiate Action module**
  Agrega este código Python a tu sección de instanciación:

```
###########################################
# **** INSTANTIATE LIBRARIES SECTION **** #
###########################################

ACTION = Action()
```

- **Access ACTION commands**
  Use general syntax such as these:

```
ACTION.SET_ESTOP_STATE(state)
ACTION.SET_MACHINE_STATE(state)

ACTION.SET_MACHINE_HOMING(joint)
ACTION.SET_MACHINE_UNHOMED(joint)

ACTION.SET_LIMITS_OVERRIDE()

ACTION.SET_MDI_MODE()
ACTION.SET_MANUAL_MODE()
ACTION.SET_AUTO_MODE()

ACTION.SET_LIMITS_OVERRIDE()

ACTION.CALL_MDI(code)
ACTION.CALL_MDI_WAIT(code)
ACTION.CALL_INI_MDI(number)

ACTION.CALL_OWORD()

ACTION.OPEN_PROGRAM(filename)
```

```
ACTION.SAVE_PROGRAM(text_source, fname):

ACTION.SET_AXIS_ORIGIN(axis,value)
ACTION.SET_TOOL_OFFSET(axis,value,fixture = False)

ACTION.RUN()
ACTION.ABORT()
ACTION.PAUSE()          # Toggles pause/resume
ACTION.PAUSE_MACHINE()
ACTION.RESUME()

ACTION.SET_MAX_VELOCITY_RATE(rate)
ACTION.SET_RAPID_RATE(rate)
ACTION.SET_FEED_RATE(rate)
ACTION.SET_SPINDLE_RATE(rate)

ACTION.SET_JOG_RATE(rate)
ACTION.SET_JOG_INCR(incr)
ACTION.SET_JOG_RATE_ANGULAR(rate)
ACTION.SET_JOG_INCR_ANGULAR(incr, text)

ACTION.SET_SPINDLE_ROTATION(direction = 1, rpm = 100, number = 0)
ACTION.SET_SPINDLE_FASTER(number = 0)
ACTION.SET_SPINDLE_SLOWER(number = 0)
ACTION.SET_SPINDLE_STOP(number = 0)

ACTION.SET_USER_SYSTEM(system)

ACTION.ZERO_G92_OFFSET()
ACTION.ZERO_ROTATIONAL_OFFSET()
ACTION.ZERO_G5X_OFFSET(num)

ACTION.RECORD_CURRENT_MODE()
ACTION.RESTORE_RECORDED_MODE()

ACTION.SET_SELECTED_AXIS(jointnum)

ACTION.DO_JOG(jointnum, direction)
ACTION.JOG(jointnum, direction, rate, distance=0)

ACTION.TOGGLE_FLOOD()
ACTION.SET_FLOOD_ON()
ACTION.SET_FLOOD_OFF()

ACTION.TOGGLE_MIST()
ACTION.SET_MIST_ON()
ACTION.SET_MIST_OFF()

ACTION.RELOAD_TOOLTABLE()
ACTION.UPDATE_VAR_FILE()

ACTION.TOGGLE_OPTIONAL_STOP()
ACTION.SET_OPTIONAL_STOP_ON()
ACTION.SET_OPTIONAL_STOP_OFF()

ACTION.TOGGLE_BLOCK_DELETE()
ACTION.SET_BLOCK_DELETE_ON()
ACTION.SET_BLOCK_DELETE_OFF()

ACTION.RELOAD_DISPLAY()
ACTION.SET_GRAPHICS_VIEW(view)
```

```
ACTION.UPDATE_MACHINE_LOG(text, option=None):

ACTION.CALL_DIALOG(command):

ACTION.HIDE_POINTER(state):

ACTION.PLAY_SOUND(path):
ACTION.PLAY_ERROR():
ACTION.PLAY_DONE():
ACTION.PLAY_READY():
ACTION.PLAY_ATTENTION():
ACTION.PLAY_LOGIN():
ACTION.PLAY_LOGOUT():
ACTION.SPEAK(speech):

ACTION.BEEP():
ACTION.BEEP_RING():
ACTION.BEEP_START():

ACTION.SET_DISPLAY_MESSAGE(string)
ACTION.SET_ERROR_MESSAGE(string)

ACTION.TOUCHPLATE_TOUCHOFF(search_vel, probe_vel, max_probe,
       z_offset, retract_distance, z_safe_travel, rtn_method=None, error_rtn = None)
```

12.8.4. Qhal

A library for HAL component/system interactions.

12.8.4.1. Attributes

These are the functions that can be called on the Qhal object:

setUpdateRate(cyclerate)
: Set cycle rate in ms

newPin(name, pin type constant, pin direction constant)
: returns a new QPin object

getPinObject(name)
: returns an existing named QPin object

getValue(name)
: returns the named pin, signal, or parameter's value, use the full component.pin name.

setPin(name,value)
: sets the named pin's value, use the full component.pin name.

setSignal(name,value)
: sets the named signal's value, use the full component.pin name.

makeUniqueName(name)
: returns an unique HAL pin name string by adding *-x* (a number) to the given pin name string

exit()
: kills the component

12.8.4.2. Constantes

Here are the available constants:

- **HAL_BIT**
- **HAL_FLOAT**
- **HAL_S32**
- **HAL_U32**
- **HAL_IN**
- **HAL_OUT**
- **HAL_IO**
- **HAL_RO**
- **HAL_RW**

12.8.4.3. References

Available object references:

- **comp** the component object
- **hal** the hal library object

12.8.5. QPin

A wrapper class around HAL pins

12.8.5.1. Signals

There are 3 Qt signals that the QPin pin can be connect to:

- **value_changed** will call a named function with an argument of the current value (depreciated)
- **pinValueChanged** will call a named function with arguments of the pin object and the current value
- **isDrivenChanged** will call a named function with arguments of the pin object and current state when the pin is (un)connected to a driving pin

12.8.5.2. Attributes

These are the functions that can be called on a QPin object:

- **<objeto pin>.get()** devuelve el valor actual del objeto pin
- **<objeto pin>.set(X)** asigna X al valor del objeto pin
- **<objeto pin>.text()** devuelve la cadena del nombre del pin

12.8.5.3. References

Available object references:

- **hal** the hal library object

12.8.5.4. Ejemplo

Add a function that gets called when the pin state changes

```
from qtvcp.core import Qhal
QHAL = Qhal()

########################################
# Special Functions called from QtVCP
########################################

# en este punto:
# los widgets se han instanciado.
# los pines HAL se han construido pero HAL no esta listo
def initialized__(self):
    self.pin_button_in = QHAL.newpin('cycle-start-in',QHAL.HAL_BIT, QHAL.HAL_IN)
    self.pin_button_in.pinValuechanged.connect(self.buttonChanged)
    self.pin_button_in.isDrivenChanged.connect(lambda p,s: self.buttonDriven(p,s))

def buttonChanged(self, pinObject, value):
    print('Pin name:{} changed value to {}'.format(pinObject.text(), value))

def buttonDriven(self, pinObject, state):
    message = 'not driven by an output pin'
    if state:
        message = 'is driven by an output pin'
    print('Pin name:{} is {}'.format(pinObject.text(), message))
```

12.8.6. Tool

This library **handles tool offset file changes**.



aviso
LinuxCNC doesn't handle third party manipulation of the tool file well.

12.8.6.1. Helpers

GET_TOOL_INFO(_toolnumber_)
: This will return a Python **list of information on the requested tool number**.

GET_TOOL_ARRAY()
: This return a single Python **list of Python lists of tool information**.

  This is a raw list formed *from the system tool file*.

ADD_TOOL(_newtool_ = [_-99, 0,'0','0','0','0','0','0','0','0','0','0','0','0', 0,'New Too

This will return a Python **tuple of two Python lists of Python lists of tool information**:

- **[0]** will be *real tools information*
- **[1]** will be *wear tools information* (tool numbers will be over 10000; Fanuc style tool wear)

By default, adds a blank tool entry with tool number -99.
You can preload the `newtool` array with tool information.

DELETE_TOOLS(_toolnumber_)
Eliminar la herramienta numerada.

SAVE_TOOLFILE(_toolarray_)
This will **parse the toolarray and save it to the tool file** specified in the *INI file* as the tool path.

This tool *array must contain all the available tools information*.

This array is expected to use the LinuxCNC *raw tool array*, i.e. it does not feature tool wear entries.

It will return True if there was an error.

CONVERT_TO_WEAR_TYPE(_toolarray_)
This function **converts a LinuxCNC raw tool array to a QtVCP tool array**.

QtVCP's tool array includes entries for X and Z axis tool wear.

LinuxCNC supports tool wear by adding **tool wear information into tool entries above 10000**.

---

nota
This also **requires remap code to add the wear offsets at tool change time**.

CONVERT_TO_STANDARD_TYPE(_toolarray_)
This function **converts QtVCP's tool array into a LinuxCNC raw tool array**.

QtVCP's array includes entries for X and Z axis tool wear.

LinuxCNC supports tool wear by adding **tool wear information into tool entries above 10000**.

---

nota
This also **requires remap code to add the wear offsets t tool change time**.

12.8.7. Path

Path module gives **reference to important files paths**.

12.8.7.1. Referenced Paths

PATH.PREFS_FILENAME
La preferencia de ruta de archivo.

PATH.WORKINGDIR
The directory QtVCP was launched from.

PATH.IS_SCREEN
Is this a screen or a VCP?

PATH.CONFIGPATH
Carpeta de configuración lanzada.

PATH.RIPCONFIGDIR
The Run-in-place config folder for QtVCP screens.

PATH.BASEDIR
Base folder for LinuxCNC.

PATH.BASENAME
The Qt Designer files name (no ending).

PATH.IMAGEDIR
La carpeta de imágenes de QtVCP.

PATH.SCREENDIR
The QtVCP builtin Screen folder.

PATH.PANELDIR
The QtVCP builtin VCP folder.

PATH.HANDLER
Ruta del archivo de controlador.

PATH.HANDLERDIR
Directory where the Python handler file was found.

PATH.XML
Ruta del archivo de interfaz de usuario QtVCP.

PATH.HANDLERDIR
Directory where the UI file was found.

PATH.QSS
Ruta del archivo QSS de QtVCP.

PATH.PYDIR
Biblioteca Python de LinuxCNC.

PATH.LIBDIR
The QtVCP library folder.

PATH.WIDGET
The QtVCP widget folder.

PATH.PLUGIN
The QtVCP widget plugin folder.

PATH.VISMACHDIR
Directory where prebuilt Vismach files are found.

Actualmente en desuso:

PATH.LOCALEDIR
Locale translation folder.

PATH.DOMAIN
Dominio de traducciones.

12.8.7.2. Helpers

There are some helper functions available:

```
file_list = PATH.find_vismach_files()
directory_list = PATH.find_screen_dirs()
directory_list = PATH.find_panel_dirs()
```

12.8.7.3. Utilización

- **Import `Path` module**
  Agrega este código Python a tu sección de importación:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.core import Path
```

- **Instantiate `Path` module**
  Agrega este código Python a tu sección de instanciación:

```
###########################################
# **** INSTANTIATE LIBRARIES SECTION **** #
###########################################

PATH = Path()
```

12.8.8. `VCPWindow`

`VCPWindow` module gives **reference to the `MainWindow` and widgets**.

Typically this would be used for a library (e.g., the toolbar library uses it) as the widgets get a reference to the `MainWindow` from the `_hal_init()` function.

12.8.8.1. Utilización

- **Import `VCPWindow` module**
  Agrega este código Python a tu sección de importación:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.qt_makegui import VCPWindow
```

- **Instantiate `VCPWindow` module**
  Agrega este código Python a tu sección de instanciación:

```
###########################################
# **** INSTANTIATE LIBRARIES SECTION **** #
###########################################

WIDGETS = VCPWindow()
```

12.8.9. `Aux_program_loader`

`Aux_program_loader` module allows an easy way to **load auxiliary programs LinuxCNC often uses**.

12.8.9.1. Helpers

`load_halmeter()`
Halmeter is used to **display one HAL pin data**.
Load a `halmeter` with:

```
AUX_PRGM.load_halmeter()
```

`load_ladder()`
Carga el programa PLC *ClassicLadder*:

```
AUX_PRGM.load_ladder()
```

`load_status()`
Carga el programa LinuxCNC `status`:

```
AUX_PRGM.load_status()
```

`load_halshow()`
Load *HALshow,* configure display program:

```
AUX_PRGM.load_halshow()
```

`load_halscope()`
Load *HALscope* program:

```
AUX_PRGM.load_halscope()
```

`load_tooledit()`
Carga el programa *Tooledit*:

```
AUX_PRGM.load_tooledit(<TOOLEFILE_PATH>)
```

`load_calibration()`
Load *Calibration* program:

```
AUX_PRGM.load_calibration()
```

`keyboard_onboard()`
Load *onboard/Matchbox keyboard*

```
AUX_PRGM.keyboard_onboard(<ARGS>)
```

12.8.9.2. Utilización

- **Import `Aux_program_loader` module**
  Agrega este código Python a tu sección de importación:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.lib.aux_program_loader import Aux_program_loader
```

- **Instantiate `Aux_program_loader` module**
  Agrega este código Python a tu sección de instanciación:

```
########################################
# **** INSTANTIATE LIBRARIES SECTION **** #
########################################

AUX_PRGM = Aux_program_loader()
```

12.8.10. `Keylookup`

`Keylookup` module is used to **allow keypresses to control behaviors** such as jogging.

It's used inside the handler file to facilitate creation of **key bindings** such as keyboard jogging, etc.

12.8.10.1. Utilización

Import Keylookup module To import this modules add this Python code to your import section:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.lib.keybindings import Keylookup
```

Instantiate Keylookup module To instantiate `Keylookup` module* so you can use it, add this Python code to your instantiate section:

```
########################################
# **** INSTANTIATE LIBRARIES SECTION **** #
########################################

KEYBIND = Keylookup()
```

---

nota
`Keylookup` requires code under the `processed_key_event` function to call `KEYBIND.call()`.
Most handler files already have this code.

In the handler file, under the *initialized function* use this general syntax to **create keybindings**:

```
KEYBIND.add_call("DEFINED_KEY","FUNCTION TO CALL", USER DATA)
```

Here we add a keybinding for `F10`, `F11` and `F12`:

```
########################################
# Special Functions called from QtVCP
########################################

# en este punto:
# los widgets se han instanciado.
# los pines HAL se han construido pero HAL no esta listo
def initialized__(self):
    KEYBIND.add_call('Key_F10','on_keycall_F10',None)
    KEYBIND.add_call('Key_F11','on_keycall_override',10)
    KEYBIND.add_call('Key_F12','on_keycall_override',20)
```

And then we need to **add the functions that get called**.
In the handler file, under the `KEY BINDING CALLS` section, add this:

```
####################
# KEY BINDING CALLS #
####################

def on_keycall_F12(self,event,state,shift,cntrl,value):
    if state:
        print('F12 pressed')

def on_keycall_override(self,event,state,shift,cntrl,value):
    if state:
        print('value = {}'.format(value))
```

12.8.10.2. Key Defines

Here is a list of recognized key words. Use the quoted text.
Letter keys use *Key_* with the upper or lower letter added.
e.g., *Key_a* and *Key_A*.

```
keys = {
    Qt.Key_Escape: "Key_Escape",
    Qt.Key_Tab: "Key_Tab",
    Qt.Key_Backtab: "Key_Backtab",
    Qt.Key_Backspace: "Key_Backspace",
    Qt.Key_Return: "Key_Return",
    Qt.Key_Enter: "Key_Enter",
    Qt.Key_Insert: "Key_Insert",
    Qt.Key_Delete: "Key_Delete",
    Qt.Key_Pause: "Key_Pause",
    Qt.Key_Print: "Key_Print",
    Qt.Key_SysReq: "Key_SysReq",
    Qt.Key_Clear: "Key_Clear",
    Qt.Key_Home: "Key_Home",
    Qt.Key_End: "Key_End",
    Qt.Key_Left: "Key_Left",
    Qt.Key_Up: "Key_Up",
    Qt.Key_Right: "Key_Right",
    Qt.Key_Down: "Key_Down",
    Qt.Key_PageUp: "Key_PageUp",
    Qt.Key_PageDown: "Key_PageDown",
    Qt.Key_Shift: "Key_Shift",
    Qt.Key_Control: "Key_Control",
    Qt.Key_Meta: "Key_Meta",
    # Qt.Key_Alt: "Key_Alt",
    Qt.Key_AltGr: "Key_AltGr",
```

```
    Qt.Key_CapsLock: "Key_CapsLock",
    Qt.Key_NumLock: "Key_NumLock",
    Qt.Key_ScrollLock: "Key_ScrollLock",
    Qt.Key_F1: "Key_F1",
    Qt.Key_F2: "Key_F2",
    Qt.Key_F3: "Key_F3",
    Qt.Key_F4: "Key_F4",
    Qt.Key_F5: "Key_F5",
    Qt.Key_F6: "Key_F6",
    Qt.Key_F7: "Key_F7",
    Qt.Key_F8: "Key_F8",
    Qt.Key_F9: "Key_F9",
    Qt.Key_F10: "Key_F10",
    Qt.Key_F11: "Key_F11",
    Qt.Key_F12: "Key_F12",
    Qt.Key_F13: "Key_F13",
    Qt.Key_F14: "Key_F14",
    Qt.Key_F15: "Key_F15",
    Qt.Key_F16: "Key_F16",
    Qt.Key_F17: "Key_F17",
    Qt.Key_F18: "Key_F18",
    Qt.Key_F19: "Key_F19",
    Qt.Key_F20: "Key_F20",
    Qt.Key_F21: "Key_F21",
    Qt.Key_F22: "Key_F22",
    Qt.Key_F23: "Key_F23",
    Qt.Key_F24: "Key_F24",
    Qt.Key_F25: "Key_F25",
    Qt.Key_F26: "Key_F26",
    Qt.Key_F27: "Key_F27",
    Qt.Key_F28: "Key_F28",
    Qt.Key_F29: "Key_F29",
    Qt.Key_F30: "Key_F30",
    Qt.Key_F31: "Key_F31",
    Qt.Key_F32: "Key_F32",
    Qt.Key_F33: "Key_F33",
    Qt.Key_F34: "Key_F34",
    Qt.Key_F35: "Key_F35",
    Qt.Key_Super_L: "Key_Super_L",
    Qt.Key_Super_R: "Key_Super_R",
    Qt.Key_Menu: "Key_Menu",
    Qt.Key_Hyper_L: "Key_HYPER_L",
    Qt.Key_Hyper_R: "Key_Hyper_R",
    Qt.Key_Help: "Key_Help",
    Qt.Key_Direction_L: "Key_Direction_L",
    Qt.Key_Direction_R: "Key_Direction_R",
    Qt.Key_Space: "Key_Space",
    Qt.Key_Any: "Key_Any",
    Qt.Key_Exclam: "Key_Exclam",
    Qt.Key_QuoteDbl: "Key_QuoteDdl",
    Qt.Key_NumberSign: "Key_NumberSign",
    Qt.Key_Dollar: "Key_Dollar",
    Qt.Key_Percent: "Key_Percent",
    Qt.Key_Ampersand: "Key_Ampersand",
    Qt.Key_Apostrophe: "Key_Apostrophe",
    Qt.Key_ParenLeft: "Key_ParenLeft",
    Qt.Key_ParenRight: "Key_ParenRight",
    Qt.Key_Asterisk: "Key_Asterisk",
    Qt.Key_Plus: "Key_Plus",
    Qt.Key_Comma: "Key_Comma",
    Qt.Key_Minus: "Key_Minus",
    Qt.Key_Period: "Key_Period",
```

```
    Qt.Key_Slash: "Key_Slash",
    Qt.Key_0: "Key_0",
    Qt.Key_1: "Key_1",
    Qt.Key_2: "Key_2",
    Qt.Key_3: "Key_3",
    Qt.Key_4: "Key_4",
    Qt.Key_5: "Key_5",
    Qt.Key_6: "Key_6",
    Qt.Key_7: "Key_7",
    Qt.Key_8: "Key_8",
    Qt.Key_9: "Key_9",
    Qt.Key_Colon: "Key_Colon",
    Qt.Key_Semicolon: "Key_Semicolon",
    Qt.Key_Less: "Key_Less",
    Qt.Key_Equal: "Key_Equal",
    Qt.Key_Greater: "Key_Greater",
    Qt.Key_Question: "Key_Question",
    Qt.Key_At: "Key_At",
    Qt.Key_BracketLeft: "Key_BracketLeft",
    Qt.Key_Backslash: "Key_Backslash",
    Qt.Key_BracketRight: "Key_BracketRight",
    Qt.Key_AsciiCircum: "Key_AsciiCircum",
    Qt.Key_Underscore: "Key_Underscore",
    Qt.Key_QuoteLeft: "Key_QuoteLeft",
    Qt.Key_BraceLeft: "Key_BraceLeft",
    Qt.Key_Bar: "Key_Bar",
    Qt.Key_BraceRight: "Key_BraceRight",
    Qt.Key_AsciiTilde: "Key_AsciiTilde",

}
```

12.8.11. Messages

Messages module is used to **display pop up dialog messages on the screen**.

These messages are:

- *defined in the INI file under the `[DISPLAY]` heading,* and
- *controlled by HAL pins*.

Use this style if you need independent HAL pins for each dialog message.

12.8.11.1. Properties

BOLDTEXT
: Generally is a title.

TEXT
: Text below title, and usually longer.

DETAIL
: Text hidden unless clicked on.

PINNAME
: Nombre base de los pines HAL.

TYPE
Specifies whether it is a (can have dialog and status options together): **status** - shown in the *status bar and the notify dialog*.
Requires no user intervention.

- **nonedialog** - specifically does not show a dialog.
  okdialog - *requiring the user to click OK to close the dialog*.
  OK messages have *two HAL pins*:
  - One HAL pin to launch the dialog, and
  - one to signify it is waiting for a response. **yesnodialog** - *requiring the user to select yes or no buttons to close the dialog*.
    Yes/No messages have *three HAL pins*:
  - One to show the dialog,
  - Uno para esperar, y
  - one for the answer. ***okcanceldialog*** - *requiring the user to select ok or cancel*
    Ok/Cancel messages have _three HAL pins:
  - One to show the dialog,
  - Uno para esperar, y
  - one for the answer.
- ***closepromptdialog*** - *requiring the user to select*

By default, `STATUS` messages for `focus_overlay` and alert sound will be sent when the dialog shows. This allows screen *focus* dimming/blurring and sounds to be added to alerts.

12.8.11.2. HAL Pins

The HAL pin names would use these patterns:

<SCREEN BASENAME>.<PINNAME>
invoking s32 pin

<SCREEN BASENAME>.<PINNAME>-waiting
Waiting for the user's response output bit pin

<SCREEN BASENAME>.<PINNAME>-response
The user response output bit pin

<SCREEN BASENAME>.<PINNAME>-response-s32
The user response output s32 pin

12.8.11.3. Ejemplos

Aquí hay una muestra de un bloque de código de definición de mensaje INI que deberá encontrarse bajo el encabezado `[DISPLAY]`:

- Status bar and desktop notify pop up message:

```
MESSAGE_BOLDTEXT = NONE
MESSAGE_TEXT = Esta es una prueba de barra de estado
MESSAGE_DETAILS = STATUS DETAILS
MESSAGE_TYPE = status
MESSAGE_PINNAME = statustest
```

- Pop up dialog asking a Yes/No question:

```
MESSAGE_BOLDTEXT = NONE
MESSAGE_TEXT = Esta es una prueba de diálogo si no
MESSAGE_DETAILS = Y/N DETAILS
MESSAGE_TYPE = yesnodialog
MESSAGE_PINNAME = yndialogtest
```

- Pop up dialog asking an OK answer + Status bar and desktop notification:

```
MESSAGE_BOLDTEXT = Este es el texto corto
MESSAGE_TEXT = Este es el texto mas largo de los dos tipos de prueba. Puede ser mas largo ↩
     que el texto de la barra de estado
MESSAGE_DETAILS = BOTH DETAILS
MESSAGE_TYPE = okdialog status
MESSAGE_PINNAME = bothtest
```

The `ScreenOptions` widget can automatically set up the message system.

12.8.12. multimessages

`Messages` module is used to **display pop up dialog messages on the screen**.

These messages are:

- *defined in the INI file under the `[DISPLAY]` heading,* and
- *controlled by one s32 HAL pin per defined id*.
- *each message is called by a corresponding number on the s32 pin.*

Use this style of user messages for instance when a VFD sends error messages encoded as numbers. It uses common invoke/response/wait HAL pins for all (per ID name) multimessage dialogs. The HAL pin names would use these patterns:

<SCREEN BASENAME>.<ID NAME>
: invoking s32 pin

<SCREEN BASENAME>.<ID NAME>-waiting
: *Waiting for the user's response* output bit pin

<SCREEN BASENAME>.<ID NAME>-response
: *The user response* output bit pin

<SCREEN BASENAME>.<ID NAME>-response-s32
: *The user response* output s32 pin

12.8.12.1. Properties

`TITLE`
: Este es el título mostrado en la ventana de diálogo.

`TEXT`
: Text below title, and usually longer.

`DETAIL`
: Text hidden unless clicked on.

TYPE
Specifies type of message the user sees (can have dialog and status options together): **status** - shown in the *status bar and the notify dialog*.
Requires no user intervention.

- **nonedialog** - specifically does not show a dialog.
  okdialog - *requiring the user to click OK to close the dialog*.
  OK messages use *two HAL pins*:
  - One HAL pin to launch the dialog, and
  - one to signify it's waiting for response. **yesnodialog** - *requiring the user to select yes or no buttons to close the dialog*.
    Yes/No messages use *three HAL pins*:
  - One to show the dialog,
  - Uno para esperar, y
  - one for the answer.

By default, `STATUS` messages for `focus_overlay` and alert sound will be sent when the dialog shows. This allows screen *focus* dimming/blurring and sounds to be added to alerts.

12.8.12.2. Ejemplos

Aquí hay una muestra de un bloque de código de definición de mensaje INI que deberá encontrarse bajo el encabezado `[DISPLAY]`:

```
[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
```

12.8.13. Notify

`Notify` module is used to **send messages that are integrated into the desktop**.

It uses the `pynotify` library.

Ubuntu/Mint does not follow the standard so you can't set how long the message stays up for.
I suggest fixing this with the `notify-osd` package available from this PPA (DISCONTINUED due to move of Ubuntu to Gnome).

Notify *keeps a list of all the alarm messages since starting* in **`self.alarmpage`**.
If you click '*Show all messages*' in the notify popup, it will print them to the terminal.

The `ScreenOptions` widget can automatically set up the notify system.

Typically `STATUS` *messages* are used to sent notify messages.

12.8.13.1. Properties

You can set the:

`title`
: Notification message title text.

`message`
: Notification message content text.

`icon`
: Notification message icon.

`timeout`
: How long the message stays up for.

12.8.14. `Preferences`

`Preferences` module allows one to **load and save preference data permanently to storage media**.

The `ScreenOptions` widget can automatically set up the preference system.

QtVCP searches for the `ScreenOptions` widget first and, if found, calls **`_pref_init()`**.
This will *create the preferences object* and return it to QtVCP to pass to all the widgets and add it to the window object attributes.
In this case the preferences object would be accessible from the handler file's `initialized_` method as **`self.w.PREFS_`**. Also all widgets can have access to a specific preferences file at initialization time.
The `ScreenOptions` widget can automatically set up the preference file.

12.8.15. `Player`

This module **allows playing sounds using Gstreamer, beep and Espeak**.

It can:

- **play sound/music files** using *Gstreamer* (non blocking),
- **play sounds** using the `beep` library (currently blocks while beeping),
- **speak words** using the `espeak` library (non blocking while speaking).

There are *default alert sounds* using Mint or FreeDesktop default sounds.

You can play arbitrary sounds or even songs by specifying the path.

`STATUS` has *messages to control `Player` module*.

The `ScreenOptions` widget can automatically set up the audio system.

12.8.15.1. Sounds

Alerts There are default **alerts** to choose from:

- `ERROR`
- `READY`
- `ATTENTION`

- RING
- DONE
- LOGIN
- LOGOUT

Beeps There are three **beeps**:

- BEEP_RING
- BEEP_START
- BEEP

12.8.15.2. Utilización

- **Import `Player` module**
  Agrega este código Python a tu sección de importación:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.lib.audio_player import Player
```

- **Instantiate `Player` module**
  Agrega este código Python a tu sección de instanciación:

```
########################################
# **** INSTANTIATE LIBRARIES SECTION **** #
########################################

SOUND = Player()
SOUND._register_messages()
```

  The **`_register_messages()`** function connects the audio player to the STATUS library so sounds can be played with the STATUS message system.

12.8.15.3. Ejemplo

To play sounds upon STATUS messages, use these general syntaxes:

```
STATUS.emit('play-alert','LOGOUT')
STATUS.emit('play-alert','BEEP')
STATUS.emit('play-alert','SPEAK This is a test screen for Q t V C P')
STATUS.emit('play-sound', 'PATH TO SOUND')
```

12.8.16. Virtual Keyboard

This library allows you to **use STATUS messages to launch a virtual keyboard**.

It uses Onboard or Matchbox libraries for the keyboard.

12.8.17. Toolbar Actions

This library supplies **prebuilt submenus and actions for toolbar menus and toolbar buttons**. Toolbuttons, menu and toolbar menus are:

- *built in Qt Designer,* and
- *acciones asignadas/submenus en el archivo manejador*.

12.8.17.1. Actions

estop , power , load , reload , gcode_properties , run , pause , abort , block_delete , optional_stop

Toggles dimensions display.

12.8.17.2. Submenus

recent_submenu , home_submenu , unhome_submenu , zero_systems_submenu , grid_size_submenu

Menu to set graphic grid size

12.8.17.3. Utilización

Here is the typical code to add to the relevant *handler file* sections:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.lib.toolbar_actions import ToolBarActions

#########################################
# **** instantiate libraries section **** #
#########################################

TOOLBAR = ToolBarActions()
```

12.8.17.4. Ejemplos

- Asignando acciones de herramienta a botones de la barra de herramientas

```
########################################
# Special Functions called from QtVCP
########################################

# At this point:
#   * the widgets are instantiated,
#   * the HAL pins are built but HAL is not set ready.
def initialized__(self):
    TOOLBAR.configure_submenu(self.w.menuHoming, 'home_submenu')
    TOOLBAR.configure_action(self.w.actionEstop, 'estop')
    TOOLBAR.configure_action(self.w.actionQuit, 'quit', lambda d:self.w.close())
    TOOLBAR.configure_action(self.w.actionEdit, 'edit', self.edit)
    # Add a custom function
    TOOLBAR.configure_action(self.w.actionMyFunction, 'my_Function', self.my_function)
```

- Add a custom toolbar function:

```
####################
# GENERAL FUNCTIONS #
####################

def my_function(self, widget, state):
    print('My function State = ()'.format(state))
```

12.8.18. Qt Vismach Machine Graphics library

`Qt_vismach` is a *set of Python functions* that can be **used to create and animate models of machines**.

Vismach:

- *displays the model* in a **3D viewport**
- *animates the model parts* as the values of associated HAL pins change.

This is the *Qt based version* of the library, there is also a tkinter version available in LinuxCNC.

The Qt version *allows embedding the simulation in other screens*.

12.8.18.1. Builtin Samples

There are included *sample panels* in QtVCP for:

- a 3-Axis XYZ mill,
- a 5-Axis gantry mill,
- a 3-Axis mill with an A axis/spindle, and
- a scara mill.

Most of these samples, if loaded after a running LinuxCNC configuration (including non-QtVCP based screens), will react to machine movement.
Some require HAL pins to be connected for movement.

From a terminal (pick one):

```
qtvcp vismach_mill_xyz
qtvcp vismach_scara
qtvcp vismach_millturn
qtvcp vismach_5axis_gantry
```

12.8.18.2. Primitives Library

Provides the **basic building blocks of a simulated machine**.

`Collection`
: A `collection` is an **object of individual machine parts**.

  This holds a **hierarchical list** of primitive shapes or *STL objects* that operations can be applied to.

`Translate`
This object will perform an **OpenGL translation** calculation *on a Collection object*.

Translation refers to *moving an object in straight line* to a different position on screen.

`Scale`
This object will perform an **OpenGL scale** function *on a collection object*.

`HalTranslate`
This object will perform an **OpenGL translation** calculation *on a Collection object,* **offset by the HAL pin value**.

Translation refers to moving an object in straight line to a different position on screen.

You can either:

- *read a pin from a component owned by the Vismach object,* or
- *read a HAL system pin directly* if the component argument is set to `None`.

`Rotate`
This object will perform an **OpenGL rotation** calculation *on a Collection object*.

`HalRotate`
This object will perform an **OpenGL rotation** calculation *on a Collection object,* **offset by the HAL pin value**.

You can either:

- *read a pin from a component owned by the vismach* object, or
- *read a HAL system pin directly* if the component argument is set to `None`.

`HalToolCylinder`
This object will build a *CylinderZ object* that will **change size and length based on loaded tool dimensition** (from the tool table)

It reads the `halui.tool.diameter` and `motion.tooloffset.z` *HAL pins*.

Ejemplo de mill_xyz:

```
toolshape = CylinderZ(0)
toolshape = Color([1, .5, .5, .5], [toolshape])
tool = Collection([
    Translate([HalTranslate([tooltip], None, "motion.tooloffset.z", 0, 0, - ↩
        MODEL_SCALING)], 0, 0, 0),
    HalToolCylinder(toolshape)
])
```

`Track`
Move and rotate an object to point from one `capture()` 'd coordinate system to another.

Base object to *hold coordinates for primitive shapes*.

`CylinderX`, `CylinderY`, `CylinderZ`
Build a cylinder on the X, Y or Z axis by giving *endpoint* (X, Y, or Z) and *radii* coordinates.

`Sphere`
Build a sphere from *center* and *radius* coordinates.

`TriangleXY`, `TriangleXZ`, `TriangleYZ`
Build a triangle in the *specified plane* by giving the *corners Z coordinates* for each side.

`ArcX`
Build an arc by specifying

`Box`
Build a box specified by the *6 vertex coordinates*.

`BoxCentered`
Build a box centered on origin by specifying the *width in X and Y*, and the *height in Z*.

`BoxCenteredXY`
Build a box centered in X and Y, and running from Z=0, by specifying the *width in X and Y*, and running up or down to the specified *height in Z*.

`Capture`
Capture current transformation matrix of a collection.

---

nota
This *transforms from the current coordinate system to the viewport system*, NOT to the world system.

`Hud`
Heads up display draws a *semi-transparent text box*.

Use:

- `HUD.strs` for things that must be *updated constantly,*
- `HUD.show("stuff")` for one-shot things like error messages.

`Color`
Applies a color to the *parts of a collection*.

`AsciiSTL, AsciiOBJ`
Loads a STL or OBJ data file as a *Vismach part*.

12.8.18.3. Utilización

Import a simulation Here is how one might import the XYZ_mill simulation in a QtVCP panel or screen handler file.

```
############################
# **** IMPORT SECTION **** #
############################

import mill_xyz as MILL
```

Instantiate and use the simulation widget Instantiate the simulation widget and add it to the screen's main layout:

```
########################################
# Special Functions called from QtVCP
########################################

# At this point:
#   * the widgets are instantiated,
#   * the HAL pins are built but HAL is not set ready.
def initialized__(self):
    machine = MILL.Window()
    self.w.mainLayout.addWidget(machine)
```

12.8.18.4. More Information

More information on how to build a custom machine simulation in the Qt Vismach chapter.

12.9. QtVismach

Vismach is a set of **Python functions that can be used to create and animate models of machines**.

This chapter is about the Qt embedded version of Vismach, also see: https://sa-cnc.com/linuxcnc-vismach/ .

12.9.1. Introducción

Vismach displays the model in a **3D viewport** and the **model parts are animated as the values of associated HAL pins change**.

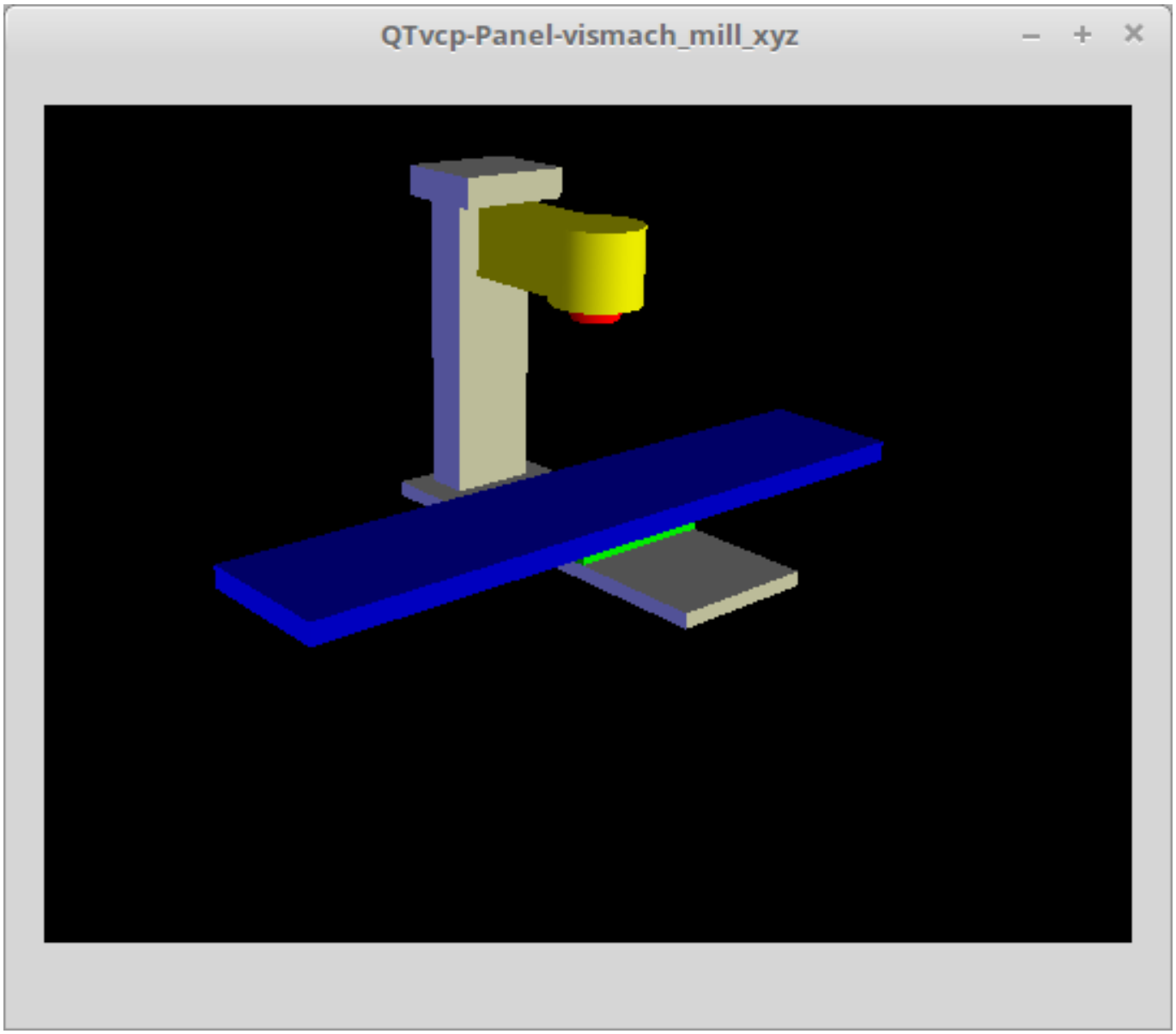


Figura 12.110: QtVismach 3D Viewport

The Vismach 3D viewport view can be manipulated as follows:

- **zoom** by *scroll wheel*
- **pan** by *middle button drag*
- **rotate** by *right-button drag*
- **tilt** by *left button drag*

A **Vismach model** takes the form of a *Python script* and can use standard Python syntax.

This means that there is more than one way to lay out the script, but in the examples given in this document the simplest and most basic of them will be used.

The basic sequence in creating the Vismach model is:

1. Create the parts
2. Define how they move
3. Assemble into movement groups

12.9.2. Hierarchy of Machine Design

The model follows **logical tree design**.

Picture the tree, with root/trunk, branches and smaller branches off it. If you move the larger branch, smaller branches will move with it, but if you move the smaller branch, the larger will not.

Machine design follows that conceptual design.

Taking the mill shown in the 3D Viewport picture above for example:

- If you move X, it can move on its own,
- but if you move Y, it will also move X assembly, as it is attached to Y assembly.

So for this machine, the tree looks like this:

```
model
  |
  |---frame
  |     |
  |     |---base
  |     |
  |     |---column
  |     |
  |     |---top
  |
  |---yassembly
  |       |
  |       |---xassembly
  |       |       |
  |       |       |---xbase
  |       |       |
  |       |       |---work
  |       |
  |       |---ybase
  |
  |---zassembly
          |
          |---zframe
          |     |
          |     |---zbody
          |     |
          |     |---spindle
          |
          |---toolassembly
                    |
                    |---cat30
                    |
                    |---tool
                          |
                          |---tooltip
                          |
                          |---(tool cylinder function)
```

As you can see, the *lowest parts must exist* first *before those can be grouped with others into an assembly*. So you *build upwards* from lowest point in tree and assemble them together.

The same is applicable for any design of machine: look at the machine arm example and you will see that it starts with the tip and adds to the larger part of the arm, then it finally groups with the base.

12.9.3. Iniciar el script

It is useful for testing to include the `#!/usr/bin/env python3` shebang line to _allow the file to be executed directly from the command line.

The first thing to do is to *import the required libraries*.

```
#!/usr/bin/env python3

import hal
import math
import sys

from qtvcp.lib.qt_vismach.qt_vismach import *
```

12.9.4. Pines HAL.

Originally the vismach library required creating a component and connecting HAL pins to control the simulation.

`qt_vismach` can read the HAL system pins directly or if you wish, to use separate HAL pins that you must define in a HAL component:

```
c = hal.component("samplegui")
c.newpin("joint0", hal.HAL_FLOAT, hal.HAL_IN)
c.newpin("joint1", hal.HAL_FLOAT, hal.HAL_IN)
c.ready()
```

You can select between the two options in the functions that take these entries:

hal_comp
: The *HAL component Object* or None.
In QtVCP if you are reading *system pins* directly, then the component argument is set to `None`.

hal_pin
: The *name of the BIT HAL IN pin* that will change the color.
If hal_comp is *None* then this must be the full name of a system pin otherwise this is the pin name excluding the component name.

12.9.5. Creating Parts

12.9.5.1. Import STL or OBJ Files

It is probably easiest to:

- *create geometry in a CAD package*
- *import into the model script using the `AsciiSTL()` or `AsciiOBJ()` functions*.

Both functions can take one of two named arguments, either a *filename* or *data*:

```
part = AsciiSTL(filename="path/to/file.stl")
part = AsciiSTL(data="solid part1 facet normal ...")
part = AsciiOBJ(filename="path/to/file.obj")
part = AsciiOBJ(data="v 0.123 0.234 0.345 1.0 ...")
```

- STL model parts are added to the Vismach space in the *same locations as they were created in the STL or OBJ space,* i.e. ideally with a rotational point at their origin.

---

nota
It is much easier to move while building if the origin of the model is at a rotational pivot point.

12.9.5.2. Build from Geometric Primitives

Alternatively parts can be *created inside the model script from a range of shape primitives*.

assembly = collction([part1,part2,part3])
Collection is a general container of related parts

Many shapes are *created at the origin* and need to be *moved to the required location* after creation.

cylinder = CylinderX(x1, r1, x2, r2) , cylinder = CylinderY(y1, r1, y2, r2) , cylinder = Cyl

Creates a *(optionally tapered) cylinder on the given axis* with the given radii at the given points on the axis.

sphere = Sphere(x, y, z, r)
Creates a *sphere of radius r at (x,y,z)*.

triangle = TriangleXY(x1, y1, x2, y2, x3, y3, z1, z2) , triangle = TriangleXZ(x1, z1, x2, z

Creates a *triangular plate between planes* defined by the last two values parallel to the specified plane, with vertices given by the three coordinate pairs.

arc = ArcX(x1, x2, r1, r2, a1, a2)
Crear una *forma de arco*.

box = Box(x1, y1, z1, x2, y2, z2)
Creates a *rectangular prism with opposite corners* at the specified positions and edges parallel to the XYZ axes.

box = BoxCentered(xw, yw, zw)
Creates an xw by yw by zw *box centred on the origin*.

box = BoxCenteredXY(xw, yw, z)
Creates a *box ground on WY plane* of width xw / yw and height z.

Composite parts may be created by assembling these primitives either at creation time or subsequently:

```
part1 = Collection([Sphere(100,100,100,50), CylinderX(100,40,150,30)])
part2 = Box(50,40,75,100,75,100)
part3 = Collection([part2, TriangleXY(10,10,20,10,15,20,100,101)])
part4 = Collection([part1, part2])
```

12.9.6. Moving Model Parts

Parts may need to be moved in the Vismach space to assemble the model. The origin does not move - Translate() and Rotate() move the Collection as you add parts, relative to a stationary origin.

12.9.6.1. Translating Model parts

`part1 = Translate([part1], x, y, z)`
Move part1 the specified distances in x, y and z.

12.9.6.2. Rotating Model Parts

`part1 = Rotate([part1], theta, x, y, z)`
Rotate the part by angle theta [degrees] about an axis between the origin and x, y, z.

12.9.7. Animating Parts

To **animate the model controlled by the values of HAL pins** there are four functions: `HalTranslate`, `HalRotate`, `HalToolCylinder` and `HalToolTriangle`.

For parts to move inside an assembly they need to have their HAL motions defined before being assembled with the "Collection" command.

The **rotation axis and translation vector move with the part**:

- as it is moved by the Vismach script during model assembly, or
- as it moves in response to the HAL pins as the model is animated.

12.9.7.1. `HalTranslate`

`part = HalTranslate([part], hal_comp, hal_pin, xs, ys, zs)`

`part`
A *collection or part*.
It can be pre-created earlier in the script, or could be created at this point if preferred, e.g.,

```
'part1 = HalTranslate([Box(....)], ...)'. +
```

`hal_comp`
The *HAL component* is the next argument.
In QtVCP if you are reading *system pins* directly then the component argument is set to `None`.

`hal_pin`
The *name of the HAL pin* that will animate the motion.
This needs to match an existing HAL pin that describes the joint position such as:

```
"joint.2.pos-fb"
```

Otherwise the component instance would be specified and the pin name of that component would be specified. `xs, ys, zs`;; The *X, Y, Z scales*.
For a Cartesian machine created at 1:1 scale this would typically be `1,0,0` for a motion in the positive X direction.
However if the STL file happened to be in cm and the machine was in inches, this could be fixed at this point by using 0.3937 (= 1 cm/1 inch = 1 cm /2.54 cm) as the scale.

12.9.7.2. HalRotate

part = HalRotate([part], hal_comp, hal_pin, angle_scale, x, y, z)
This command is similar in its operation to `HalTranslate`, except that it is typically necessary to move the part to the origin first to define the axis.

x, y, z
Defines the *axis of rotation* from the origin the point of coordinates (x,y,z).
When the part is moved back away from the origin to its correct location, the axis of rotation can be considered to remain "embedded" in the part.

angle_scale
Rotation angles are in degrees, so for a rotary joint with a 0-1 scaling you would need to use an angle scale of 360.

12.9.7.3. *HalToolCylinder*

tool = HalToolCylinder()
Make a cylinder to represent a cylindrical mill tool, based on the tool table and current loaded tool.

```
tool = HalToolCylinder()
toolshape = Color([1, .5, .5, .5],[tool])

# or more compact:
toolshape = Color([1, .5, .5, .5], [HalToolCylinder()])
```

12.9.7.4. *HalToolTriangle*

tool = HalToolTriangle()
Make a triangle to represent a triangular lathe tool, based on the tool table and current loaded tool.

```
tool = HalToolTriangle()
toolshape = Color([1, 1, 0, 1],[tool])

# or more compact:
toolshape = Color([1, 1, 0, 1],[HalToolTriangle()])
```

12.9.7.5. HAL Adjustable Primitives

All shape primitives can have HAL pin names substituted for coordinates.
Either by adding the component object as the first variable and substituting the pinname string for a coordinate, or
by substituting the full component/pinname string for a coordinate.

This example creates a *rectangular prism with opposite corners* at the specified positions and edges parallel to the XYZ axes.
The Z start coordinate will be controlled by the HAL pin *Zstart*.

```
box = Box(component, x1, y1, 'Zstart', x2, y2, z2)
box = Box(x1, y1, 'componentName.Zstart', x2, y2, z2)
```

12.9.8. Assembling the model

In order for parts to move together they need to be assembled with the **`Collection()` command**.

It is important to **assemble the parts and define their motions in the correct sequence**.

For example to create a moving head milling machine with a rotating spindle and an animated draw bar you would:

- Create the head main body.
- Create the spindle at the origin.
- Define the rotation.
- Move the head to the spindle or spindle to the head.
- Crear la barra de dibujo.
- Define the motion of the draw bar.
- Assemble the three parts into a head assembly.
- Define the motion of the head assembly.

In this example the spindle rotation is indicated by rotation of a set of drive dogs:

```
#Drive dogs
dogs = Box(-6,-3,94,6,3,100)
dogs = Color([1,1,1,1],[dogs])
dogs = HalRotate([dogs],c,"spindle",360,0,0,1)
dogs = Translate([dogs],-1,49,0)

#Drawbar
draw = CylinderZ(120,3,125,3)
draw = Color([1,0,.5,1],[draw])
draw = Translate([draw],-1,49,0)
draw = HalTranslate([draw],c,"drawbar",0,0,1)

# head/spindle
head = AsciiSTL(filename="./head.stl")
head = Color([0.3,0.3,0.3,1],[head])
head = Translate([head],0,0,4)
head = Collection([head, tool, dogs, draw])
head = HalTranslate([head],c,"Z",0,0,0.1)

# base
base = AsciiSTL(filename="./base.stl")
base = Color([0.5,0.5,0.5,1],[base])
# mount head on it
base = Collection([head, base])
```

Finally a **single collection of all the machine parts, floor and work** (if any) needs to be created.

For a *serial machine* each new part will be added to the collection of the previous part.

For a *parallel machine* there may be several "base" parts.

Thus, for example, in `scaragui.py` link3 is added to link2, link2 to link1 and link1 to link0, so the final model is created by:

```
model = Collection([link0, floor, table])
```

Whereas a VMC model with separate parts moving on the base might have

```
model = Collection([base, saddle, head, carousel])
```

12.9.9. Other functions

12.9.9.1. Color

Establece el *color en pantalla de la pieza*.

part = Color([_colorspec_], [_part_])
Note that unlike the other functions, the part definition comes second in this case.

colorspec
Three (0-1.0) RGB values and opacity. [R,G,B,A]
For example [1.0,0,0,0.5] for a 50 % opacity red.

12.9.9.2. HALColorFlip

Sets the *display color of the part based on a designated HAL bit pin state*.

part = HALColorFlip([_colorspec_], [_colorspec_], [_part_], hal_comp, hal_pin)
Note that unlike the other functions, the part definition comes second in this case.

colorspec
Three (0-1.0) RGB values and opacity.
For example [1.0,0,0,0.5] for a 50 % opacity red.

hal_comp
The *HAL component Object* or None.
In QtVCP if you are reading *system pins* directly, then the component argument is set to `None`.

hal_pin
The *name of the BIT HAL IN pin* that will change the color.
if hal_comp is *None* then this must be the full name of a system pin other wise this is the pin name excluding the component name.

12.9.9.3. HALColorRGB

Sets the *display color of the part based on a designated HAL U32 pin value*.
The color is decoded from the U32 value. each color is a 0-255 decimal value (shown here in hex)
red = 0xXXXXXXRR
green = 0xXXXXGGXX
blue = 0xXXBBXXXX
combined as 0xXXBBGGRR

part = HALColorRGB([_part_], hal_comp, hal_pin, alpha=1.0)

hal_comp
The *HAL component Object* or None.
In QtVCP if you are reading *system pins* directly, then the component argument is set to `None`.

hal_pin
The *name of the U32 HAL IN pin* that will change the color.
if hal_comp is *None* then this must be the full name of a system pin other wise this is the pin name excluding the component name.

alpha=
Sets the opacity. (0-1.0)

12.9.9.4. Heads Up Display

Creates a *heads-up display* in the Vismach GUI to display items such as axis positions, titles, or messages.

myhud = Hud()

```
myhud = Hud()
myhud.show("Mill_XYZ")‘
```

12.9.9.5. HAL Heads Up Display

A more advanced version of the Hud that allows HAL pins to be displayed:

myhud = HalHud()

```
myhud = HalHud()
myhud.display_on_right()
myhud.set_background_color(0,.1,.2,0)
myHud.set_text_color(1,1,1)
myhud.show_top("Mill_XYZ")
myhud.show_top("------------")
myhud.add_pin('axis-x: ',"{:10.4f}","axis.x.pos-cmd")
myhud.add_pin('axis-y: ',"{:10.4f}","axis.y.pos-cmd")
myhud.add_pin('axis-z: ',"{:10.4f}","axis.z.pos-cmd")
myhud.show("------------")
```

Some of the available HalHUD function:

- set_background_color(red, green, blue, alpha)
- add_pin(text, format, pinname)
- set_text_color(red, green, blue)

12.9.9.6. HideCollection

```
 HideCollection is a container that uses a HAL pin to control display of the ↩
    contained parts. +
A logic high on the HAL pin will hide the contained parts.
```

```
comp.newpin("hide-chuck", hal.HAL_BIT, hal.HAL_IN)
# <snip make a machine with an A axis chuck assembly>
chuckassembly = HideCollection([chuckassembly],comp,'hide-chuck')
# also can be used like this
chuckassembly = HideCollection([chuckassembly],None,'myvismach.hide-chuck')
```

12.9.9.7. Plot Color Based on Motion Type

If you wish to plot different colors for different motions you need to add some more Python code.

Add this at the top of the file:

```
from qtvcp.core import Status
STATUS = Status()
```

and this to the Window class:

```
STATUS.connect('motion-type-changed', lambda w, data: v.choosePlotColor(data))

# uncomment to change feed color and to see all colors printed to the terminal
#v.setColorsAttribute('FEED',(0,1,0))
#print(v.colors)
```

You can set DEFAULT, FEED, TRAVERSE, ARC, PROBE, ROTARYINDEX, TOOLCHANGE colors with setColorsAttribute().

12.9.9.8. Capture

- `tooltip = Capture()`
  Think of this as an invisible part that needs to be attached to the tooltip to track the position and orientation of the tool coordinate system. It is actually a transformation matrix that is constantly updated as the model moves.
- `work = Capture()`
  Same as above but attached to the work table to track the work coordinate system.

12.9.9.9. main

This is the command that makes it all happen, creates the display, etc. if invoked directly from Python. Usually this file is imported by QtVCP and the `window()` object is instantiated and embedded into another screen.

`main(model, tooltip, work, size=10, hud=myhud, rotation_vectors=None, lat=0, lon=0)`

model
: Should be a collection that contains all the machine parts.

tooltip and _work_
: Need to be created by `Capture()` to draw the backplot which is basically the tooltip position drawn in the work coordinate system. See `mill_xyz.py` for an example of how to connect the tool tip to a tool and the tool to the model.

size
: Sets the extent of the volume visualized in the initial view.

hud
: refers to a head-up display.

_rotation_vectors_ or _lat, lon_
: Can be used to set the original viewpoint. It is advisable to do as the default initial viewpoint is rather unhelpful from immediately overhead.

12.9.10. Tips

Create an axes origin marker to be able to see parts relative to it, for construction purposes. You can remove it when you are done.

```
# build axis origin markers
X = CylinderX(-500,1,500,1)
X = Color([1, 0, 0, 1], [X])
Y = CylinderY(-500,1,500,1)
Y = Color([0, 1, 0, 1], [Y])
Z = CylinderZ(-500,1,500,1)
Z = Color([0, 0, 1, 1], [Z])
origin = Collection([X,Y,Z])
```

Add it to the Window class Collection so it is never moved from the origin.

```
v.model = Collection([origin, model, world])
```

Start from the cutting tip and work your way back. Add each collection to the model at the origin and run the script to confirm the location, then rotate/translate and run the script to confirm again.

12.9.11. Basic structure of a `QtVismach` script

```
# imports
import hal
from qtvcp.lib.qt_vismach.qt_vismach import *

# importa Status para mensajes de tipo de movimiento
from qtvcp.core import Status
STATUS = Status()

# create HAL pins here if needed
#c = hal.component("samplegui")
#c.newpin("joint0", hal.HAL_FLOAT, hal.HAL_IN)

# create the floor, tool and work
floor = Box(-50, -50, -3, 50, 50, 0)
work = Capture()
tooltip = Capture()

# Build and assemble the model
part1 = Collection([Box(-6,-3,94,6,3,100)])
part1 = Color([1,1,1,1],[part1])
part1 = HalRotate([part1],None,"joint.0.pos-fb",360,0,0,1)
part1 = Translate([dogs],-1,49,0)

# create a top-level model
model = Collection([base, saddle, head, carousel])

# we want to either embed into qtvcp or display directly with PyQt5
# so build a window to display the model

class Window(QWidget):

    def __init__(self):
        super(Window, self).__init__()
        self.glWidget = GLWidget()
        v = self.glWidget
        v.set_latitudelimits(-180, 180)
```

```
        world = Capture()

        # uncomment if there is a HUD
        # HUD needs to know where to draw
        #v.hud = myhud
        #v.hud.app = v

        # update plot color based on motion type
        STATUS.connect('motion-type-changed', lambda w, data: v.choosePlotColor(data))

        # uncomment to change feed color
        #v.setColorsAttribute('FEED',(0,1,0))
        # and to see all colors printed to the terminal
        #print(v.colors)

        v.model = Collection([model, world])
        size = 600
        v.distance = size * 3
        v.near = size * 0.01
        v.far = size * 10.0
        v.tool2view = tooltip
        v.world2view = world
        v.work2view = work

        mainLayout = QHBoxLayout()
        mainLayout.addWidget(self.glWidget)
        self.setLayout(mainLayout)

# if you call this file directly from python3, it will display a PyQt5 window
# good for confirming the parts of the assembly.

if __name__ == '__main__':
    main(model, tooltip, work, size=600, hud=None, lat=-75, lon=215)
```

12.9.12. Builtin Vismach Sample Panels

QtVCP builtin Vismach Panels

12.10. QtVCP: Building Custom Widgets

12.10.1. Descripción general

Building custom widgets allows one to **use the Qt Designer editor to place a custom widget** *rather than doing it manually in a handler file*.

A useful custom widgets would be a great way to contribute back to LinuxCNC.

12.10.1.1. Widgets

Widget is the *general name for the UI objects* such as buttons and labels in PyQt.

There are also **special widgets made for LinuxCNC** that make integration easier.

Todos estos widgets pueden *colocarse con el editor Qt Designer* - permitiéndonos *ver el resultado* antes de cargar realmente el panel en LinuxCNC.

12.10.1.2. Qt Designer

Qt Designer is a *WYSIWYG (What You See is What You Get) editor for placing PyQt widgets*.

It's original intend was for building the graphic widgets for programs.
We leverage it to **build screens and panels for LinuxCNC**.

In Qt Designer, on the left side of the editor, you find **three categories of LinuxCNC widgets**:

- *Solo widgets HAL*.
- *Widgets de controlador LinuxCNC*.
- *Widgets de diálogo*.

For Qt Designer to *add custom widgets* to it's editor it must have a **plugin** added to the right folder.

12.10.1.3. Initialization Process

QtVCP does *extra setup* for **widgets subclassed from `_HALWidgetBase`**, aka "HAL-ified" widgets.
This includes:

- Injecting *important variables*,
- Calling an *extra setup function*
- Calling a *closing cleanup function* at shutdown.

These functions are not called when the Qt Designer editor displays the widgets.
Cuando QtVCP construye una pantalla a partir de un archivo `.ui`:

1. It searches for all the HAL-ified widgets.
2. It finds the `ScreenOptions` widget, to collect information it needs to inject into the other widgets
3. It instantiates each widget and if it is a HAL-ified widget, calls the `hal_init()` function.
   `hal_init()` is defined in the base class and it:
   a. Adds variables such as the preference file to every HAL-ified widget.
   b. Call `+_hal_init()+` on the widget.
      `+_hal_init()+` allows the widget designer to do setup that requires access to the extra variables.

Here is a description of the extra variables injected into "HAL-ified" widgets:

`self.HAL_GCOMP`
The *HAL component instance*

`self.HAL_NAME`
This *widget's name* as a string

`self.QT_OBJECT_`
This *widget's object instance*

`self.QTVCP_INSTANCE_`
The *very top level parent* of the screen

`self.PATHS_`
The *QtVCP's path library instance*

`self.PREFS_`
The *optional preference file instance*

`self.SETTINGS_`
The `Qsettings` *object instance*

12.10.1.4. proceso de limpieza

When QtVCP closes, it calls the **+_hal_cleanup()+** function *on all HAL-ified widgets*.

The base class creates an empty +_hal_cleanup()+ function, which can be redefined in the custom widget subclass.

This can be used to do such things as record preferences, etc.

This function is not called when the Qt Designer editor displays the widgets.

12.10.2. Custom HAL Widgets

HAL widgets are the simplest to show example of.
`qtvcp/widgets/simple_widgets.py` holds many HAL only widgets.

Lets look at a snippet of `simple_widgets.py`:

In the Imports section This is where we import libraries that our widget class needs.

```
#!/usr/bin/env python3

##############################
# Imports
##############################
from PyQt5 import QtWidgets                     # ❶
from qtvcp.widgets.widget_baseclass \
    import _HalWidgetBase, _HalSensitiveBase    # ❷
import hal                                      # ❸
```

In this case we need access to:

❶ PyQt's QtWidgets library,

❷ LinuxCNC's HAL library, and

❸ QtVCP's widget `baseclass` 's **`_HalSensitiveBase`** for *automatic HAL pin setup* and to *disable/enable the widget* (also known as input sensitivity).
There is also `_HalToggleBase`, and `_HalScaleBase` functions available in the library.`_HalToggleBase`, and `_HalScaleBase`.

In the WIDGET section Here is a *custom widget* based on PyQt's **`QGridLayout`** widget.

`QGridLayout` allows one to:

- *Place objects in a grid* fashion.
- *Enable/disable all widgets inside it* based on a **HAL pin state**.

```
######################
# WIDGET
######################

class Lcnc_GridLayout(QtWidgets.QWidget, _HalSensitiveBase):    # ❶
    def __init__(self, parent = None):                          # ❷
        super(GridLayout, self).__init__(parent)                # ❸
```

Line by Line:

1. This defines the *class name* and the *libraries it inherits from*.
   This class, named `Lcnc_GridLayout`, inherits the functions of `QWidget` and `+_HalSensitiveBase+`.
   `+_HalSensitiveBase+` is *subclass* of **`+_HalWidgetBase+`**, the *base class of most QtVCP widgets*, meaning it has all the functions of `+_HalWidgetBase+` plus the functions of `+_HalSensitiveBase+`.
   It adds the function to make the widget be enabled or disabled based on a HAL input BIT pin.

2. This is the function *called when the widget is first made* (said instantiated) - this is pretty standard.

3. This function initializes our widget's **`Super` classes**.
   `Super` just means the *inherited baseclasses*, that is `QWidget` and `_HalSensitiveBase`.
   Pretty standard other than the widget name will change.

12.10.3. Custom Controller Widgets Using `STATUS`

Widget that interact with LinuxCNC's controller are only a little more complicated and they require some *extra libraries*.

In this cut down example we will add properties that can be changed in Qt Designer.

This LED indicator widget will respond to selectable LinuxCNC controller states.

```
#!/usr/bin/env python3

##############################
# Imports
##############################
from PyQt5.QtCore import pyqtProperty
from qtvcp.widgets.led_widget import LED
from qtvcp.core import Status

############################################
# **** instantiate libraries section **** #
############################################
STATUS = Status()

########################################
# custom widget class definition
########################################
class StateLED(LED):
    def __init__(self, parent=None):
        super(StateLED, self).__init__(parent)
        self.has_hal_pins = False
        self.setState(False)
        self.is_estopped = False
        self.is_on = False
        self.invert_state = False

    def _hal_init(self):
        if self.is_estopped:
            STATUS.connect('state-estop', lambda w:self._flip_state(True))
            STATUS.connect('state-estop-reset', lambda w:self._flip_state(False))
        elif self.is_on:
            STATUS.connect('state-on', lambda w:self._flip_state(True))
            STATUS.connect('state-off', lambda w:self._flip_state(False))

    def _flip_state(self, data):
            if self.invert_state:
                data = not data
            self.change_state(data)
```

```
    ########################################################################
    # Qt Designer properties setter/getters/resetters
    ########################################################################

    # invert status
    def set_invert_state(self, data):
        self.invert_state = data
    def get_invert_state(self):
        return self.invert_state
    def reset_invert_state(self):
        self.invert_state = False

    # machine is estopped status
    def set_is_estopped(self, data):
        self.is_estopped = data
    def get_is_estopped(self):
        return self.is_estopped
    def reset_is_estopped(self):
        self.is_estopped = False

    # machine is on status
    def set_is_on(self, data):
        self.is_on = data
    def get_is_on(self):
        return self.is_on
    def reset_is_on(self):
        self.is_on = False

    ######################################
    # Qt Designer properties
    ######################################
    invert_state_status = pyqtProperty(bool, get_invert_state, set_invert_state,  ↩
        reset_invert_state)
    is_estopped_status = pyqtProperty(bool, get_is_estopped, set_is_estopped,  ↩
        reset_is_estopped)
    is_on_status = pyqtProperty(bool, get_is_on, set_is_on, reset_is_on)
```

12.10.3.1. En la sección de importaciones

This is where we import libraries that our widget class needs.

```
#!/usr/bin/env python3

################################
# Imports
################################
from PyQt5.QtCore import pyqtProperty       # ❶
from qtvcp.widgets.led_widget import LED    # ❷
from qtvcp.core import Status               # ❸
```

We import

❶ `pyqtProperty` so we can interact with the Qt Designer editor,

❷ `LED` because our custom widget is based on it,

❸ `Status` because it gives us status messages from LinuxCNC.

12.10.3.2. In The *Instantiate Libraries* Section

Here we create the `Status` library instance:

```
########################################
# **** instantiate libraries section **** #
########################################
STATUS = Status()
```

Típicamente instanciamos la librería *fuera de la clase del widget* de tal manera que la referencia a él es **global** - significando que no necesitas anteponer `self.`.

By convention we use *all capital* letters in the name for global references.

12.10.3.3. In The *Custom Widget Class Definition* Section

This is the meat and potatoes of our custom widget.

Class definition and instance initialization function

```
class StateLed(LED):                              # ❶
    def __init__(self, parent=None):              # ❷
        super(StateLed, self).__init__(parent)    # ❸
        self.has_hal_pins = False                 # ❹
        self.setState(False)                      # ❺
        self.is_estopped = False
        self.is_on = False
        self.invert_state = False
```

❶ Defines the **name** *of our custom widget* and what other *class it inherits from*.
In this case we inherit `LED` - a QtVCP widget that represents a status light.

❷ Typical of most widgets - called when the widget is first made.

❸ Typical of most widgets - calls the parent (super) widget initialization code.
Entonces asignamos algunos atributos:

❹ Inherited from `Lcnc_Led` - we set it here so no HAL pin is made.

❺ Inherited from `Lcnc_led` - we set it to make sure the LED is off.

The other attributes are for the selectable options of our widget.

Widget's HAL initialization function

```
    def _hal_init(self):
        if self.is_estopped:
            STATUS.connect('state-estop', lambda w:self._flip_state(True))
            STATUS.connect('state-estop-reset', lambda w:self._flip_state(False))
        elif self.is_on:
            STATUS.connect('state-on', lambda w:self._flip_state(True))
            STATUS.connect('state-off', lambda w:self._flip_state(False))
```

This function connects `STATUS` (LinuxCNC status message library) to our widget, so that the LED will on or off based on the selected state of the controller.

We have two states we can choose from `is_estopped` or `is_on`.
Depending on which is active our widget get connected to the appropriate STATUS messages.

+_hal_init()+ is *called on each widget that inherits* +_HalWidgetBase+, when QtVCP first builds the screen.
You might wonder why it's called on this widget since we didn't have +_HalWidgetBase+ in our class definition (`class Lcnc_State_Led(Lcnc_Led):`) - it's called because `Lcnc_Led` inherits +_HalWidgetBase+.

In this function you have access to some extra information (though we don't use them in this example):

self.HAL_GCOMP
: the *HAL component* instance

self.HAL_NAME
: This *widget's name* as a string

self.QT_OBJECT_
: This *widget's PyQt object instance*

self.QTVCP_INSTANCE_
: The *very top level parent* of the screen

self.PATHS_
: The *instance of QtVCP's path* library

self.PREFS_
: the *instance of an optional preference file*

self.SETTINGS_
: the `Qsettings` *object*

We could use this information to create HAL pins or look up image paths etc.

```
STATUS.connect('state-estop', lambda w:self._flip_state(True))
```

Lets look at this line more closely:

- `STATUS` is very common theme is widget building.
  `STATUS` uses `GObject` message system to send messages to widgets that register to it.
  This line is the registering process.
- `state-estop` is the message we wish to listen for and act on. There are many messages available.
- `lambda w:self._flip_state(True)` is what happens when the message is caught.
  The lambda function accepts the widget instance (`w`) that GObject sends it and then calls the function `self._flip_state(True)`.
  Lambda was used to strip the (`w`) object before calling the `self._flip_state` function.
  It also allowed use to send `self._flip_state()` the True state.

```
    def _flip_state(self, data):
            if self.invert_state:
                data = not data
            self.change_state(data)
```

This is the function that actually flips the state of the LED.
It is what gets called when the appropriate STATUS message is accepted.

```
STATUS.connect('current-feed-rate', self._set_feedrate_text)
```

The function called looks like this:

```
    def _set_feedrate_text(self, widget, data):
```

in which the widget and any data must be accepted by the function.

```
    ########################################################################
    # Qt Designer properties setter/getters/resetters
    ########################################################################

    # invert status
    def set_invert_state(self, data):
        self.invert_state = data
    def get_invert_state(self):
        return self.invert_state
    def reset_invert_state(self):
        self.invert_state = False

    # machine is estopped status
    def set_is_estopped(self, data):
        self.is_estopped = data
    def get_is_estopped(self):
        return self.is_estopped
    def reset_is_estopped(self):
        self.is_estopped = False

    # machine is on status
    def set_is_on(self, data):
        self.is_on = data
    def get_is_on(self):
        return self.is_on
    def reset_is_on(self):
        self.is_on = False
```

This is *how Qt Designer sets the attributes of the widget*.
This *can also be called directly* in the widget.

```
    ######################################
    # Qt Designer properties
    ######################################
    invert_state_status = pyqtProperty(bool, get_invert_state, set_invert_state,  ↩
        reset_invert_state)
    is_estopped_status = pyqtProperty(bool, get_is_estopped, set_is_estopped,  ↩
        reset_is_estopped)
    is_on_status = pyqtProperty(bool, get_is_on, set_is_on, reset_is_on)
```

This is the **registering of properties in Qt Designer**.

El **nombre de la propiedad**:

- is the *text used in Qt Designer,*
- *cannot be the same as the attributes* they represent.

These properties show in Qt Designer in the order they appear here.

12.10.4. Custom Controller Widgets with Actions

Here is an example of a widget that sets the user reference system.

It changes:

- the machine controller state using the `ACTION` library,
- whether the button can be clicked or not using the `STATUS` library.

```
import os
import hal

from PyQt5.QtWidgets import QWidget, QToolButton, QMenu, QAction
from PyQt5.QtCore import Qt, QEvent, pyqtProperty, QBasicTimer, pyqtSignal
from PyQt5.QtGui import QIcon

from qtvcp.widgets.widget_baseclass import _HalWidgetBase
from qtvcp.widgets.dialog_widget import EntryDialog
from qtvcp.core import Status, Action, Info

# Instantiate the libraries with global reference
# STATUS gives us status messages from LinuxCNC
# INFO holds INI details
# ACTION gives commands to LinuxCNC
STATUS = Status()
INFO = Info()
ACTION = Action()

class SystemToolButton(QToolButton, _HalWidgetBase):
    def __init__(self, parent=None):
        super(SystemToolButton, self).__init__(parent)
        self._joint = 0
        self._last = 0
        self._block_signal = False
        self._auto_label_flag = True
        SettingMenu = QMenu()
        for system in('G54', 'G55', 'G56', 'G57', 'G58', 'G59', 'G59.1', 'G59.2', 'G59.3'):

            Button = QAction(QIcon('exit24.png'), system, self)
            Button.triggered.connect(self[system.replace('.','_')])
            SettingMenu.addAction(Button)

        self.setMenu(SettingMenu)
        self.dialog = EntryDialog()

    def _hal_init(self):
        if not self.text() == '':
            self._auto_label_flag = False
        def homed_on_test():
            return (STATUS.machine_is_on()
                    and (STATUS.is_all_homed() or INFO.NO_HOME_REQUIRED))

        STATUS.connect('state-off', lambda w: self.setEnabled(False))
        STATUS.connect('state-estop', lambda w: self.setEnabled(False))
        STATUS.connect('interp-idle', lambda w: self.setEnabled(homed_on_test()))
        STATUS.connect('interp-run', lambda w: self.setEnabled(False))
        STATUS.connect('all-homed', lambda w: self.setEnabled(True))
        STATUS.connect('not-all-homed', lambda w, data: self.setEnabled(False))
        STATUS.connect('interp-paused', lambda w: self.setEnabled(True))
        STATUS.connect('user-system-changed', self._set_user_system_text)

    def G54(self):
        ACTION.SET_USER_SYSTEM('54')

    def G55(self):
        ACTION.SET_USER_SYSTEM('55')

    def G56(self):
        ACTION.SET_USER_SYSTEM('56')
```

```
    def G57(self):
        ACTION.SET_USER_SYSTEM('57')

    def G58(self):
        ACTION.SET_USER_SYSTEM('58')

    def G59(self):
        ACTION.SET_USER_SYSTEM('59')

    def G59_1(self):
        ACTION.SET_USER_SYSTEM('59.1')

    def G59_2(self):
        ACTION.SET_USER_SYSTEM('59.2')

    def G59_3(self):
        ACTION.SET_USER_SYSTEM('59.3')

    def _set_user_system_text(self, w, data):
        convert = { 1:"G54", 2:"G55", 3:"G56", 4:"G57", 5:"G58", 6:"G59", 7:"G59.1", 8:"G59 ↩
            .2", 9:"G59.3"}
        if self._auto_label_flag:
            self.setText(convert[int(data)])

    def ChangeState(self, joint):
        if int(joint) != self._joint:
            self._block_signal = True
            self.setChecked(False)
            self._block_signal = False
            self.hal_pin.set(False)

    ##############################
    # required class boiler code #
    ##############################

    def __getitem__(self, item):
        return getattr(self, item)
    def __setitem__(self, item, value):
        return setattr(self, item, value)
```

12.10.5. Stylesheet Property Changes Based On Events

It's possible to **have widgets restyled when events change**. You must explicitly *"polish" the widget* to have PyQt redo the style.
This is a relatively expensive function so should be used sparingly.

This example sets an `isHomed` property based on LinuxCNC's homed state and in turn uses it to change stylesheet properties:

This example will set the property isHomed based on LinuxCNC's homed state.

```
class HomeLabel(QLabel, _HalWidgetBase):
    def __init__(self, parent=None):
        super(HomeLabel, self).__init__(parent)
        self.joint_number = 0
        # for stylesheet reading
        self._isHomed = False

    def _hal_init(self):
        super(HomeLabel, self)._hal_init()
```

```
        STATUS.connect('homed', lambda w,d: self._home_status_polish(int(d), True))
        STATUS.connect('unhomed', lambda w,d: self._home_status_polish(int(d), False))

    # update ishomed property
    # polish widget so stylesheet sees the property change
    # some stylesheets color the text on home/unhome
    def _home_status_polish(self, d, state):
        if self.joint_number = d:
            self.setProperty('isHomed', state)
            self.style().unpolish(self)
            self.style().polish(self)

    # Qproperty getter and setter
    def getisHomed(self):
        return self._isHomed
    def setisHomed(self, data):
        self._isHomed = data

    # Qproperty
    isHomed = QtCore.pyqtProperty(bool, getisHomed, setisHomed)
```

Here is a sample stylesheet to change text color based on home state.

In this case any widget based on the HomeLabel widget above will change text color.
You would usually pick specific widgets using `HomeLabel #specific_widget_name[homed=true]`:

```
HomeLabel[homed=true] {
    color: green;
}
HomeLabel[homed=false] {
    color: red;
}
```

12.10.6. Use Stylesheets To Change Custom Widget Properties

```
class Label(QLabel):
    def __init__(self, parent=None):
        super(Label, self).__init__(parent)
        alternateFont0 = self.font

    # Qproperty getter and setter
    def getFont0(self):
        return self.aleternateFont0
    def setFont0(self, value):
        self.alternateFont0(value)
    # Qproperty
    styleFont0 = pyqtProperty(QFont, getFont0, setFont0)
```

Sample stylesheet that sets a custom widget property.

```
Label{
    qproperty-styleFont0: "Times,12,-1,0,90,0,0,0,0,0";
}
```

12.10.7. Widget Plugins

We must *register our custom widget* for Qt Designer to use them.

Here are a typical samples.
They would need to be added to `qtvcp/plugins/`
Then `qtvcp/plugins/qtvcp_plugin.py` would need to be adjusted to *import* them.

12.10.7.1. Ejemplo de Gridlayout

```
#!/usr/bin/env python3

from PyQt5 import QtCore, QtGui
from PyQt5.QtDesigner import QPyDesignerCustomWidgetPlugin
from qtvcp.widgets.simple_widgets import Lcnc_GridLayout
from qtvcp.widgets.qtvcp_icons import Icon
ICON = Icon()

#####################################
# GridLayout
#####################################
class LcncGridLayoutPlugin(QPyDesignerCustomWidgetPlugin):
    def __init__(self, parent = None):
        QPyDesignerCustomWidgetPlugin.__init__(self)
        self.initialized = False
    def initialize(self, formEditor):
        if self.initialized:
            return
        self.initialized = True
    def isInitialized(self):
        return self.initialized
    def createWidget(self, parent):
        return Lcnc_GridLayout(parent)
    def name(self):
        return "Lcnc_GridLayout"
    def group(self):
        return "LinuxCNC - HAL"
    def icon(self):
        return QtGui.QIcon(QtGui.QPixmap(ICON.get_path('lcnc_gridlayout')))
    def toolTip(self):
        return "HAL enable/disable GridLayout widget"
    def whatsThis(self):
        return ""
    def isContainer(self):
        return True
    def domXml(self):
        return '<widget class="Lcnc_GridLayout" name="lcnc_gridlayout" />\n'
    def includeFile(self):
        return "qtvcp.widgets.simple_widgets"
```

12.10.7.2. SystemToolbutton Example

```
#!/usr/bin/env python3

from PyQt5 import QtCore, QtGui
from PyQt5.QtDesigner import QPyDesignerCustomWidgetPlugin
from qtvcp.widgets.system_tool_button import SystemToolButton
from qtvcp.widgets.qtvcp_icons import Icon
ICON = Icon()

#####################################
# SystemToolButton
```

```
##################################
class SystemToolButtonPlugin(QPyDesignerCustomWidgetPlugin):
    def __init__(self, parent = None):
        super(SystemToolButtonPlugin, self).__init__(parent)
        self.initialized = False
    def initialize(self, formEditor):
        if self.initialized:
            return
        self.initialized = True
    def isInitialized(self):
        return self.initialized
    def createWidget(self, parent):
        return SystemToolButton(parent)
    def name(self):
        return "SystemToolButton"
    def group(self):
        return "LinuxCNC - Controller"
    def icon(self):
        return QtGui.QIcon(QtGui.QPixmap(ICON.get_path('systemtoolbutton')))
    def toolTip(self):
        return "Button for selecting a User Coordinate System"
    def whatsThis(self):
        return ""
    def isContainer(self):
        return False
    def domXml(self):
        return '<widget class="SystemToolButton" name="systemtoolbutton" />\n'
    def includeFile(self):
        return "qtvcp.widgets.system_tool_button"
```

12.10.7.3. Making a plugin with a MenuEntry dialog box

It possible to add an entry to the dialog that pops up when you right click the widget in the layout.

This can do things such as selecting options in a more convenient way.

This is the plugin used for *action buttons*.

```
#!/usr/bin/env python3

import sip
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtDesigner import QPyDesignerCustomWidgetPlugin, \
                QPyDesignerTaskMenuExtension, QExtensionFactory, \
                QDesignerFormWindowInterface, QPyDesignerMemberSheetExtension
from qtvcp.widgets.action_button import ActionButton
from qtvcp.widgets.qtvcp_icons import Icon
ICON = Icon()

Q_TYPEID = {
    'QDesignerContainerExtension':     'org.qt-project.Qt.Designer.Container',
    'QDesignerPropertySheetExtension': 'org.qt-project.Qt.Designer.PropertySheet',
    'QDesignerTaskMenuExtension': 'org.qt-project.Qt.Designer.TaskMenu',
    'QDesignerMemberSheetExtension': 'org.qt-project.Qt.Designer.MemberSheet'
}

##################################
# ActionBUTTON
##################################
class ActionButtonPlugin(QPyDesignerCustomWidgetPlugin):
```

```
    # The __init__() method is only used to set up the plugin and define its
    # initialized variable.
    def __init__(self, parent=None):
        super(ActionButtonPlugin, self).__init__(parent)
        self.initialized = False

    # The initialize() and isInitialized() methods allow the plugin to set up
    # any required resources, ensuring that this can only happen once for each
    # plugin.
    def initialize(self, formEditor):

        if self.initialized:
            return
        manager = formEditor.extensionManager()
        if manager:
            self.factory = ActionButtonTaskMenuFactory(manager)
            manager.registerExtensions(self.factory, Q_TYPEID['QDesignerTaskMenuExtension' ↩
                ])
        self.initialized = True

    def isInitialized(self):
        return self.initialized

    # This factory method creates new instances of our custom widget
    def createWidget(self, parent):
        return ActionButton(parent)

    # This method returns the name of the custom widget class
    def name(self):
        return "ActionButton"

    # Returns the name of the group in Qt Designer's widget box
    def group(self):
        return "LinuxCNC - Controller"

    # Returns the icon
    def icon(self):
        return QtGui.QIcon(QtGui.QPixmap(ICON.get_path('actionbutton')))

    # Returns a tool tip short description
    def toolTip(self):
        return "Action button widget"

    # Returns a short description of the custom widget for use in a "What's
    # This?" help message for the widget.
    def whatsThis(self):
        return ""

    # Returns True if the custom widget acts as a container for other widgets;
    def isContainer(self):
        return False

    # Returns an XML description of a custom widget instance that describes
    # default values for its properties.
    def domXml(self):
        return '<widget class="ActionButton" name="actionbutton" />\n'

    # Returns the module containing the custom widget class. It may include
    # a module path.
    def includeFile(self):
        return "qtvcp.widgets.action_button"
```

```
class ActionButtonDialog(QtWidgets.QDialog):

   def __init__(self, widget, parent = None):

      QtWidgets.QDialog.__init__(self, parent)

      self.widget = widget

      self.previewWidget = ActionButton()

      buttonBox = QtWidgets.QDialogButtonBox()
      okButton = buttonBox.addButton(buttonBox.Ok)
      cancelButton = buttonBox.addButton(buttonBox.Cancel)

      okButton.clicked.connect(self.updateWidget)
      cancelButton.clicked.connect(self.reject)

      layout = QtWidgets.QGridLayout()
      self.c_estop = QtWidgets.QCheckBox("Estop Action")
      self.c_estop.setChecked(widget.estop )
      layout.addWidget(self.c_estop)

      layout.addWidget(buttonBox, 5, 0, 1, 2)
      self.setLayout(layout)

      self.setWindowTitle(self.tr("Set Options"))

   def updateWidget(self):

      formWindow = QDesignerFormWindowInterface.findFormWindow(self.widget)
      if formWindow:
          formWindow.cursor().setProperty("estop_action",
              QtCore.QVariant(self.c_estop.isChecked()))
      self.accept()

class ActionButtonMenuEntry(QPyDesignerTaskMenuExtension):

    def __init__(self, widget, parent):
        super(QPyDesignerTaskMenuExtension, self).__init__(parent)
        self.widget = widget
        self.editStateAction = QtWidgets.QAction(
          self.tr("Set Options..."), self)
        self.editStateAction.triggered.connect(self.updateOptions)

    def preferredEditAction(self):
        return self.editStateAction

    def taskActions(self):
        return [self.editStateAction]

    def updateOptions(self):
        dialog = ActionButtonDialog(self.widget)
        dialog.exec_()

class ActionButtonTaskMenuFactory(QExtensionFactory):
    def __init__(self, parent = None):
        QExtensionFactory.__init__(self, parent)

    def createExtension(self, obj, iid, parent):

        if not isinstance(obj, ActionButton):
```

```
            return None
        if iid == Q_TYPEID['QDesignerTaskMenuExtension']:
            return ActionButtonMenuEntry(obj, parent)
        elif iid == Q_TYPEID['QDesignerMemberSheetExtension']:
            return ActionButtonMemberSheet(obj, parent)
        return None
```

12.11. QtVCP Handler File Code Snippets

12.11.1. Preference File Loading/Saving

Here is how to **load and save preferences at launch and closing time**.

Prerequisites

- *Preference file option must be set* in the `ScreenOptions` widget.
- *Preference file path must be set* in the `INI` configuration.

Reading preferences at launch time Under the **`def initialized__(self):`** function add:

```
if self.w.PREFS_:
    # variable name (entry name, default value, type, section name)
    self.int_value = self.w.PREFS_.getpref('Integer_value', 75, int, 'CUSTOM_FORM_ENTRIES')
    self.string_value = self.w.PREFS_.getpref('String_value', 'on', str, ' ↩
        CUSTOM_FORM_ENTRIES')
```

Writing preferences at close time En la función **`closing_cleanup__()`** agregue:

```
if self.w.PREFS_:
    # variable name (entry name, variable name, type, section name)
    self.w.PREFS_.putpref('Integer_value', self.integer_value, int, 'CUSTOM_FORM_ENTRIES')
    self.w.PREFS_.putpref('String_value', self.string_value, str, 'CUSTOM_FORM_ENTRIES')
```

12.11.2. Use `QSettings` To Read/Save Variables

Here is how to **load and save variables using PyQt's `QSettings`** functions:

Good practices

- *Use `Group` to keep names organized and unique*.
- *Account for none value* returned when reading a setting which has no entry.
- *Set defaults to cover the first time it is run* using the `or _<default_value>_` syntax.

---

nota
The file is actually saved in `~/.config/QtVcp`

Ejemplo In this example:

- We add `or 20` and `or 2.5` as defaults.

- The names `MyGroupName`, `int_value`, `float_value`, `myInteger`, and `myFloat` are user defined.
- Under the **`def initialized__(self):`** function add:

```
# set recorded columns sort settings
self.SETTINGS_.beginGroup("MyGroupName")
self.int_value = self.SETTINGS_.value('myInteger', type = int) or 20
self.float_value = self.SETTINGS_.value('myFloat', type = float) or 2.5
self.SETTINGS_.endGroup()
```

- Abajo de la función **`def closing_cleanup__(self):`** agregue:

```
# save values with QSettings
self.SETTINGS_.beginGroup("MyGroupName")
self.SETTINGS_.setValue('myInteger', self.int_value)
self.SETTINGS_.setValue('myFloat', self.float_value)
self.SETTINGS_.endGroup()
```

12.11.3. Add A Basic Style Editor

Being able to **edit a style on a running screen** is convenient.

Import StyleSheetEditor module in the IMPORT SECTION:

```
from qtvcp.widgets.stylesheeteditor import StyleSheetEditor as SSE
```

Instantiate StyleSheetEditor module in the INSTANTIATE SECTION:

```
STYLEEDITOR = SSE()
```

Create a keybinding in the INITIALIZE SECTION: Under the `+__init__.(self, halcomp, widgets, paths):+` function add:

```
KEYBIND.add_call('Key_F12','on_keycall_F12')
```

Create the key bound function in the KEYBINDING SECTION:

```
def on_keycall_F12(self,event,state,shift,cntrl):
    if state:
        STYLEEDITOR.load_dialog()
```

12.11.4. Request Dialog Entry

QtVCP uses `STATUS` messages to **pop up and return information from dialogs**.

Prebuilt dialogs keep track of their last position and include options for focus shading and sound.

To *get information back from the dialog* requires using a `STATUS` **`general`** message.

Import and Instantiate the Status module in the IMPORT SECTION

```
from qtvcp.core import Status
STATUS = Status()
```

This loads and initializes the `Status` library.

Register function for STATUS general messages in the INITIALIZE SECTION Under the `+__init__.(halcomp, widgets, paths)+` function:

```
STATUS.connect('general',self.return_value)
```

This registers STATUS to call the function `self.return_value` when a general message is sent.

Add entry dialog request function in the GENERAL FUNCTIONS section

```
def request_number(self):
    mess = {'NAME':'ENTRY','ID':'FORM__NUMBER', 'TITLE':'Set Tool Offset'}
    STATUS.emit('dialog-request', mess)
```

The function

- creates a Python `dict` with:
  - **`NAME`** - needs to be set to the *dialogs unique launch name*.
    NAME sets which dialog to request.
    ENTRY or CALCULATOR allows entering numbers.
  - **`ID`** - needs to be set to a *unique name that the function supplies*. ID should be a unique key.
  - **`TITLE`** sets the dialog title.
  - **Arbitrary data** can be added to the `dict`. The dialog will ignore them but send them back to the return code.
- Sends the `dict` as a **`dialog-request`** STATUS message

Add message data processing function in the CALLBACKS FROM STATUS section.

```
# Process the STATUS return message from set-tool-offset
def return_value(self, w, message):
    num = message.get('RETURN')
    id_code = bool(message.get('ID') == 'FORM__NUMBER')
    name = bool(message.get('NAME') == 'ENTRY')
    if id_code and name and num is not None:
        print('The {} number from {} was: {}'.format(name, id_code, num))
```

This catches all general messages so it must *check the dialog type and id code* to confirm it's our dialog. In this case we had requested an ENTRY dialog and our unique id was FORM_NUMBER, so now we know the message is for us. ENTRY or CALCULATOR dialogs return a float number.

12.11.5. Speak a Startup Greeting

This requires the espeak library installed on the system.

Import and instantiate the Status in the IMPORT section

```
from qtvcp.core import Status
STATUS = Status()
```

Emit spoken message in the INITIALIZE SECTION Under the `init.(self, halcomp, widgets, paths)` function:

```
STATUS.emit('play-alert','SPEAK Please remember to oil the ways.')
```

`SPEAK` is a keyword: *everything after it will be pronounced*.

12.11.6. ToolBar Functions

Toolbar buttons and submenus are added in Qt Designer but the code to make them do something is added in the handler file. To **add a submenus** in Qt Designer:

- Add a `Qaction` by typing in the toolbar column then clicking the + icon on the right.
- This will add a sub column that you need to type a name into.
- Now the original `Qaction` will be a `Qmenu` instead.
- Now erase the `Qaction` you added to that `Qmenu`, the menu will stay as a menu.

In this example we assume you added a toolbar with one submenu and three actions. These actions will be configured to create:

- a recent file selection menu,
- an about pop up dialog action,
- a quit program action, and
- a user defined function action.

The `objectName` of the toolbar button is used to identify the button when configuring it - *descriptive names help*.

Using the action editor menu, right click and select edit.
Edit the object name, text, and button type for an appropriate action.

In this example the:

- submenu name must be `menuRecent`,
- actions names must be `actionAbout`, `actionQuit`, `actionMyFunction`

Loads the toolbar_actions library in the IMPORT SECTION

```
from qtvcp.lib.toolbar_actions import ToolBarActions
```

Instantiate ToolBarActions module in the INSTANTIATE LIBRARY SECTION

```
TOOLBAR = ToolBarActions()
```

Configure submenus and actions in the SPECIAL FUNCTIONS SECTION Under the `def initialized_` function add:

```
TOOLBAR.configure_submenu(self.w.menuRecent, 'recent_submenu')
TOOLBAR.configure_action(self.w.actionAbout, 'about')
TOOLBAR.configure_action(self.w.actionQuit, 'Quit', lambda d:self.w.close())
TOOLBAR.configure_action(self.w.actionMyFunction, 'My Function', self.my_function)
```

Define the user function in the GENERAL FUNCTIONS SECTION

```
def my_function(self, widget, state):
    print('My function State = ()'.format(state))
```

The function to be called if the action "My Function" button is pressed.

12.11.7. Add HAL Pins That Call Functions

In this way you *don't need to poll the state of input pins*.

Loads the Qhal library in the IMPORT SECTION

```
from qtvcp.core import Qhal
```

This is to allow access to **QtVCP's HAL component**. For more info: Qhal
Qhal's newPin function returns a QPin object. For more info: QPin

Instantiate Qhal in the INSTANTIATE LIBRARY SECTION

```
QHAL = Qhal()
```

Add a function that gets called when the pin state changes Under the `initialized__` function, make sure there is an entry similar to this:

```
########################################
# Special Functions called from QtVCP
########################################

# at this point:
# the widgets are instantiated.
# the HAL pins are built but HAL is not set ready
def initialized__(self):
    self.pin_cycle_start_in = QHAL.newPin('cycle-start-in',QHAL.HAL_BIT, QHAL.HAL_IN)
    self.pin_cycle_start_in.pinValueChanged.connect(lambda o,s: self.cycleStart(s))
```

Define the function called by pin state change in the GENERAL FUNCTIONS SECTION

```
######################
# funciones generales #
######################

def cycleStart(self, state):
    if state:
        tab = self.w.mainTab.currentWidget()
        if  tab in( self.w.tab_auto,  self.w.tab_graphics):
            ACTION.RUN(line=0)
        elif tab == self.w.tab_files:
                self.w.filemanager.load()
        elif tab == self.w.tab_mdi:
            self.w.mditouchy.run_command()
```

This function assumes there is a Tab widget, named `mainTab`, that has tabs with the names `tab_auto`, `tab_graphics`, `tab_filemanager` and `tab_mdi`.

In this way the cycle start button works differently depending on what tab is shown.

This is simplified - *checking state and error trapping might be helpful*.

12.11.8. Read/Write System HAL Pins Directly

Sometimes you need to read a system pin and creating a HAL pin and connecting to it, is more work then required. You can read it directly without connecting to it.

Here is how to read a pin, parameter, or signal:

```
self.h.hal.get_value('spindle.0.at-speed')

# or use the Qhal library like this:
QHAL.getValue('spindle.0.at-speed')
```

Here is how to write to an unconnected pin or a non driven signal:

```
self.h.hal.set_p('componentName.pinName','10')
self.h.hal.set_s('componentName.signalName','10')

# or use the Qhal library like this:
QHAL.setPin('componentName.pinName', '10')
QHAL.setSignal('componentName.signalName', '10')
```

Using *self.h.hal* or *QHAL.hal* allows access to the HAL python module functions: Python HAL Interface

12.11.9. Add A Special Max Velocity Slider Based On Percent

Some times you want to **build a widget to do something not built in**. The built in Max velocity slider acts on units per minute, here we show how to do on percent.

The **STATUS** command makes sure the slider adjusts if LinuxCNC changes the current max velocity.

valueChanged.connect() *calls a function when the slider is moved*.

In Qt Designer add a **QSlider** widget called mvPercent, then add the following code to the handler file:

```
#############################
# SPECIAL FUNCTIONS SECTION #
#############################

def initialized__(self):
    self.w.mvPercent.setMaximum(100)
    STATUS.connect('max-velocity-override-changed', \
        lambda w, data: self.w.mvPercent.setValue( \
            (data / INFO.MAX_TRAJ_VELOCITY)*100 \
            )
        )
    self.w.mvPercent.valueChanged.connect(self.setMVPercentValue)

####################
# GENERAL FUNCTIONS #
####################

def setMVPercentValue(self, value):
    ACTION.SET_MAX_VELOCITY_RATE(INFO.MAX_TRAJ_VELOCITY * (value/100.0))
```

12.11.10. Toggle Continuous Jog On and Off

Generally selecting continuous jogging is a momentary button, that requires you to select the previous jog increment after.

We will build a button that toggles between continuous jog and whatever increment that was already selected.

En Qt Designer:

- Add an ActionButton with no action
- Call it btn_toggle_continuous.
- Set the AbstractButton property checkable to True.
- Set the ActionButton properties incr_imperial_number and incr_mm_number to 0.

- Use Qt Designer's slot editor to use the button signal `clicked(bool)` to call form's handler function `toggle_continuous_clicked()`.
  See Using Qt Designer To Add Slots section for more information.

Then add this code snippets to the handler file under the `initialized__` function:

```
# en este punto:
# se han instanciado los widgets.
# se han construido los pines HAL pero HAL no esta listo
def initialized__(self):
    STATUS.connect('jogincrement-changed', \
        lambda w, d, t: self.record_jog_incr(d,t) \
        )
    # asigna un incremento predeterminado para alternar a
    self.L_incr = 0.01
    self.L_text = "0.01in"
```

En la sección `GENERAL FUNCTIONS` agregue:

```
####################
# GENERAL FUNCTIONS #
####################

# if it isn't continuous, record the latest jog increment
# and untoggle the continuous button
def record_jog_incr(self,d, t):
    if d != 0:
        self.L_incr = d
        self.L_text = t
        self.w.btn_toggle_continuous.safecheck(False)
```

En la sección`CALLBACKS FROM FORM` agregue:

```
######################
# CALLBACKS FROM FORM #
######################

def toggle_continuous_clicked(self, state):
    if state:
        # set continuous (call the actionbutton's function)
        self.w.btn_toggle_continuous.incr_action()
    else:
        # reset previously recorded increment
        ACTION.SET_JOG_INCR(self.L_incr, self.L_text)
```

12.11.11. Class Patch The File Manager Widget

nota
Class patching (monkey patching) is a little like *black magic* - so use it *only when needed*. The Major problem is if the widget library functions are changed during development, the functions may break.
The File manager widget is designed to load a selected program in LinuxCNC.
But maybe you want to print the file name first.

We can "class patch" the library to *redirect the function call*.
You can do this class patch inside or outside the HandlerClass instance.
This will change what *self* represents in the function.

Outside the HanderClass, *self* will be the patched class instance.
Inside the HanderClass, *self* will be the HandlerClass instance.
This would change what functions/variables you can access in the function.
Here we show an inside the HandlerClass example:
In the `IMPORT SECTION` add:

```
from qtvcp.widgets.file_manager import FileManager as FM
```

Here we are going to:

1. *Keep a reference to the original function* (1) so we can still call it
2. *Redirect the class to call our custom function* (2) in the handler file instead.

```
########################################
# Special Functions called from QtVCP    #
########################################

# For changing functions in widgets we can 'class patch'.
# class patching must be done before the class is instantiated.
def class_patch__(self):
    self.old_load = FM.load # keep a reference of the old function (1)
    FM.load = self.our_load # redirect function to our handle file function (2)
```

3. *Write a custom function to replace the original*:
   This function must have the **same signature as the original function**.
   self is the HandlerClass instance *not* the patched class instance.
   In this example we are still going to call the original function by using the reference to it we recorded earlier.
   It *requires the first argument to be the widget instance*, which in this case is `self.w.filemanager` (the name given in the Qt Designer editor).

```
####################
# GENERAL FUNCTIONS #
####################

def our_load(self,fname):
    print(fname)
    self.old_load(self.w.filemanager,fname)
```

Now our custom function will print the file path to the terminal before loading the file. Obviously boring but shows the principle.

nota
There is another slightly different way to do this that can have advantages: you can *store the reference to the original function in the original class*.
The trick here is to make sure the function name you use to store it is not already used in the class. `super__` added to the function name would be a good choice.
We won't use that in built in QtVCP widgets.

```
########################################
# Special Functions called from QtVCP
########################################

# For changing functions in widgets we can 'class patch'.
# class patching must be done before the class is instantiated.
def class_patch__(self):
    FM.super__load = FM.load # keep a reference of the old function in the original class
    FM.load = self.our_load # redirect function to our handle file function

####################
# GENERAL FUNCTIONS #
####################

def our_load(self,fname):
    print(fname)
    self.w.filemanager.super__load(fname)
```

12.11.12. Adding Widgets Programmatically

In some situation it is only possible to **add widgets with Python code** rather then using the Qt Designer editor.

When adding QtVCP widgets programmatically, sometimes there are *extra steps* to be taken.

Here we are going to add a spindle speed indicator bar and up-to-speed LED to a tab widget corner. Qt Designer does not support adding corner widgets to tabs but PyQt does.

This is a cut down example from QtAxis screen's handler file.

Import required libraries First we must import the libraries we need, if they're not already imported in the handler file:

- `QtWidgets` gives us access to the `QProgressBar`,
- `QColor` is for the *LED color*,
- `StateLED` is the QtVCP library used to *create the spindle-at-speed LED*,
- `Status` is used to *catch LinuxCNC status information*,
- `Info` gives us *information about the machine configuration*.

```
############################
# **** IMPORT SECTION **** #
############################

from PyQt5 import QtWidgets
from PyQt5.QtGui import QColor
from qtvcp.widgets.state_led import StateLED as LED
from qtvcp.core import Status, Info
```

Instantiate Status and Info channels `STATUS` and `INFO` are initialized outside the handler class so as to be *global references* (no self. in front):

```
########################################
# **** instantiate libraries section **** #
########################################

STATUS = Status()
INFO = Info()
```

Register STATUS monitoring function For the spindle speed indicator we need to know the current spindle speed. For this we *register* with `STATUS` to:

- *Catch* the `actual-spindle-speed-changed` *signal*
- *Call* the `self.update_spindle()` *function*

```
########################
# **** INITIALIZE **** #
########################
# Widgets allow access to widgets from the QtVCP files.
# At this point the widgets and HAL pins are not instantiated.
def __init__(self,halcomp,widgets,paths):
    self.hal = halcomp
    self.w = widgets
    self.PATHS = paths

    STATUS.connect('actual-spindle-speed-changed', \
        lambda w,speed: self.update_spindle(speed))
```

Add the widgets to the tab We need to *make sure the Qt Designer widgets are already built* before we try to add to them. For this, we add a call to `self.make_corner_widgets()` function to build our extra widgets at the right time, i.e. under the `initialized__()` function:

```
########################################
# Special Functions called from QtScreen #
########################################

# en este punto:
# se han instanciado los widgets.
# se han construido los pines HAL pero HAL aún no esta listo
def initialized__(self):
    self.make_corner_widgets()
```

Create the widgets building functions Ok let's code the function to build the widgets and add them in the tab widget. We are assuming there is a tab widget built with Designer called *rightTab*.

We are assuming there is a tab widget built with Qt Designer called `rightTab`.

```
######################
# funciones generales #
######################

def make_corner_widgets(self):
    # make a spindle-at-speed green LED
    self.w.led = LED()                                          # ❶
    self.w.led.setProperty('is_spindle_at_speed_status',True) # ❷
    self.w.led.setProperty('color',QColor(0,255,0,255))        # ❸
    self.w.led.hal_init(HAL_NAME = 'spindle_is_at_speed')      # ❹
```

```
    # make a spindle speed bar
    self.w.rpm_bar = QtWidgets.QProgressBar()                    # (5)
    self.w.rpm_bar.setRange(0, INFO.MAX_SPINDLE_SPEED)           # (6)

    # container
    w = QtWidgets.QWidget()                                      # (7)
    w.setContentsMargins(0,0,0,6)
    w.setMinimumHeight(40)

    # layout
    hbox = QtWidgets.QHBoxLayout()                               # (8)
    hbox.addWidget(self.w.rpm_bar)                               # (9)
    hbox.addWidget(self.w.led)                                   # (10)
    w.setLayout(hbox)

    # add the container to the corner of the right tab widget
    self.w.rightTab.setCornerWidget(w)                           # (11)
```

(1), (1) This initializes the basic StateLed widget and uses `self.w.led` as the reference from then on.

(2), (2) Since the state LED can be used for many indications, we must set the property that designates it as a spindle-at-speed LED.

(3) This sets it as green when on.

(4) This is the extra function call required with some QtVCP widgets.
If `HAL_NAME` is omitted it will use the widget's `objectName` if there is one.
It gives the special widgets reference to:

`self.HAL_GCOMP`
: the *HAL component* instance

`self.HAL_NAME`
: This *widget's name* as a string

`self.QT_OBJECT_`
: This *widget's PyQt object instance*

`self.QTVCP_INSTANCE_`
: The *very top level parent* of the screen

`self.PATHS_`
: The *instance of QtVCP's path* library

`self.PREFS_`
: the *instance of an optional preference file*

`self.SETTINGS_`
: the `Qsettings` *object*

(5) Initializes a PyQt5 `QProgressBar`.

(6) Sets the max range of the progress bar to the max specified in the `INI`.

(7) We create a QWidget
Since you can only add one widget to the tab corner and we want two there, we must add both into a **container**.

(8) add a QHBoxLayout to the QWidget.

(9), (10) Then we add our QProgress bar and LED to the layout.

11 Finally we add the QWidget (with our QProgress bar and LED in it) to the tab widget's corner.

Create the STATUS monitoring function Now we build the function to actually update out the `QProgressBar` when `STATUS` updates the spindle speed:

```
########################
# callbacks from STATUS #
########################
def update_spindle(self, data):
    self.w.rpm_bar.setInvertedAppearance(bool(data<0))       # 1
    self.w.rpm_bar.setFormat('{0:d} RPM'.format(int(data)))  # 2
    self.w.rpm_bar.setValue(abs(data))                       # 3
```

1 In this case we chose to display left-to-right or right-to-left, depending if we are turning clockwise or anticlockwise.

2 This formats the writing in the bar.

3 This sets the length of the colored bar.

12.11.13. Update/Read Objects Periodically

Sometimes you need to **update a widget or read a value regularly** that isn't covered by normal libraries.

Here we update an LED based on a watched HAL pin every 100 ms.

We assume there is an LED named `led` in the Qt Designer UI file.

Load the Qhal library for access to QtVCP's HAL component En la sección `IMPORT` agregue:

```
from qtvcp.core import Qhal
```

Instantiate Qhal En la sección `INSTANTIATE LIBRARY` agregue:

```
QHAL = Qhal()
```

Now add/modify these sections to include code that is similar to this:

Register a function to be called at CYCLE_TIME period This is usually every 100 ms.

```
########################
# **** INITIALIZE **** #
########################
# widgets allows access to widgets from the QtVCP files
# at this point the widgets and hal pins are not instantiated
def __init__(self,halcomp,widgets,paths):
    self.hal = halcomp
    self.w = widgets
    self.PATHS = paths

    # register a function to be called at CYCLE_TIME period (usually every 100 ms)
    STATUS.connect('periodic', lambda w: self.update_periodic())
```

Crea la función personalizada para ser llamada periódicamente

```
####################
# general functions #
####################
def update_periodic(self):
    data = QHAL.getvalue('spindle.0.is-oriented')
    self.w.led.setState(data)
```

12.11.14. External Control With ZMQ

QtVCP can automatically set up **ZMQ messaging** *to send and/or receive remote messages from external programs.*

It uses ZMQ's **publish/subscribe messaging pattern**.

As always, consider **security** before letting programs interface though messaging.

12.11.14.1. ZMQ Messages Reading

Sometimes you want to **control the screen with a separate program**.

Enable reception of ZMQ messages In the `ScreenOptions` widget, you can select the property **`use_receive_zmq_option`**.
You can also set this property directly *in the handler file*, as in this sample.

We assume the `ScreenOptions` widget is called `screen_options` in Qt Designer:

```
########################
# **** INITIALIZE **** #
########################
# los widgets permiten acceso a los widgets desde los archivos QtVCP
# en este punto no se han instanciado los widgets y los pines HAL
def __init__(self,halcomp,widgets,paths):
    # selecciona directamente la recepción de mensaje ZMQ
    self.w.screen_options.setProperty('use_receive_zmq_option',True)
```

This **allows an external program to call functions in the handler file**.

Agregue una función para ser llamada a la recepción del mensaje ZMQ Let's add a specific function for testing. You will need to run LinuxCNC from a terminal to see the printed text.

```
####################
# general functions #
####################
def test_zmq_function(self, arg1, arg2):
    print('zmq_test_function called: ', arg1, arg2)
```

Create an external program sending ZMQ messages that will trigger function call Here is a sample external program to call a function. It alternates between two data sets every second. Run this in a separate terminal from LinuxCNC to see the sent messages.

```
#!/usr/bin/env python3
from time import sleep

import zmq
import json

context = zmq.Context()
socket = context.socket(zmq.PUB)
socket.bind("tcp://127.0.0.1:5690")
topic = b'QtVCP'

# prebuilt message 1
# makes a dict of function to call plus any arguments
x = {                                   # (1)
  "FUNCTION": "test_zmq_function",
  "ARGS": [True,200]
}
# convert to JSON object
m1 = json.dumps(x)
```

```
# prebuild message 2
x = {                                   # ❷
  "FUNCTION": "test_zmq_function",
  "ARGS": [False,0],
}
# convert to JSON object
m2 = json.dumps(x)

if __name__ == '__main__':
    while True:
        print('send message 1')
        socket.send_multipart([topic, bytes((m1).encode('utf-8'))])
        sleep(ms(1000))

        print('send message 2')
        socket.send_multipart([topic, bytes((m2).encode('utf-8'))])
        sleep(ms(1000))
```

❶, ❷ Set the **function to call** and the **arguments to send** to that function.

You will need to know the *signature* of the function you wish to call. Also note that the *message is converted to a JSON object*. This is because ZMQ sends byte messages not Python objects. `json` converts Python objects to bytes and will be converted back when received.

12.11.14.2. ZMQ Messages Writing

You may also want to **communicate with an external program from the screen**.

In the `ScreenOptions` widget, you can select the property **`use_send_zmq_message`**. You can also set this property directly *in the handler file*, as in this sample.

We assume the `ScreenOptions` widget is called `screen_options` in Qt Designer:

Enable sending of ZMQ messages

```
########################
# **** INITIALIZE **** #
########################
# 'widgets' permite acceso a los widgets desde los archivos QtVCP
# en este punto no se han instanciado los widgets y los pines HAL
def __init__(self, halcomp,widgets,paths):
    # selecciona directamente envío de mensajes ZMQ
    self.w.screen_options.setProperty('use_send_zmq_option',True)
```

This allows sending messages to a separate program.
The message sent will depend on what the external program is expecting.

Create a function to send ZMQ messages Let's add a specific function for testing.
You will need to run LinuxCNC from a terminal to see the printed text.
Also, something needs to be added to call this function, such as a button click.

```
####################
# general functions #
####################
def send_zmq_message(self):
    # This could be any Python object JSON can convert
    message = {"name": "John", "age": 30}
    self.w.screen_options.send_zmq_message(message)
```

Use or create a program that will receive ZMQ messages Here is a sample program that will receive the message and print it to the terminal:

```
import zmq
import json

# ZeroMQ Context
context = zmq.Context()

# Define the socket using the "Context"
sock = context.socket(zmq.SUB)

# Define subscription and messages with topic to accept.
topic = "" # all topics
sock.setsockopt_string(zmq.SUBSCRIBE, topic)
sock.connect("tcp://127.0.0.1:5690")

while True:
    topic, message = sock.recv_multipart()
    print('{} sent message:{}'.format(topic,json.loads(message)))
```

12.11.15. Sending Messages To Status Bar Or Desktop Notify Dialogs

Hay varias manera para **reportar información al usuario**.

Una **barra de estado** es usada para mostrar *información corta* al usuario.

---

nota
Not all screens have a status bar.

Status bar usage example

```
self.w.statusbar.showMessage(message, timeout * 1000)
```

`timeout` is in seconds and we assume `statusbar` is the Qt Designer set name of the widget.

You can also use the `Status` library to send a message to the `notify` library if it is enabled (usually set in `ScreenOptions` widget): This will send the message to the statusbar and the **desktop notify dialog**.

The messages are also recorded until the user erases them using controls. The users can recall any recorded messages.

There are several options:

STATUS.TEMPORARY_MESSAGE
Show the message for a short time only.

STATUS.OPERATOR_ERROR , STATUS.OPERATOR_TEXT , STATUS.NML_ERROR , STATUS.NML_TEXT

Example of sending an operator message:

```
STATUS.emit('error', STATUS.OPERATOR_ERROR, 'message')
```

You can send messages thru LinuxCNC's operator message functions. These are usually caught by the notify system, so are equal to above. They would be printed to the terminal as well.

```
ACTION.SET_DISPLAY_MESSAGE('MESSAGE')
ACTION.SET_ERROR_MESSAGE('MESSAGE')
```

12.11.16. Catch Focus Changes

Focus is used to **direct user action** such as keyboard entry to the proper widget.

Get currently focused widget

```
fwidget = QtWidgets.QApplication.focusWidget()
if fwidget is not None:
    print("focus widget class: {} name: {} ".format(fwidget, fwidget.objectName()))
```

Get focused widget when focus changes

```
# en este punto:
# los widgets se han instanciado.
# los pines HAL se han construido pero HAL no esta listo
def initialized__(self):
    QtWidgets.QApplication.instance().event_filter.focusIn.connect(self.focusInChanged)

######################
# funciones generales #
######################

def focusInChanged(self, widget):
    if isinstance(widget.parent(),type(self.w.gcode_editor.editor)):
        print('G-code Editor')
    elif isinstance(widget,type(self.w.gcodegraphics)):
        print('G-code Display')
    elif isinstance(widget.parent(),type(self.w.mdihistory) ):
        print('MDI History')
```

Notice we sometimes compare to `widget`, sometimes to `widget.parent()`.

This is because *some QtVCP widgets are built from multiple **sub-widgets*** and the latter actually get the focus; so we need to **check the parent** of those sub-widgets.

Other times the main widget is what gets the focus, e.g., the G-code display widget can be set to accept the focus. In that case there are no sub-widgets in it, so comparing to the `widget.parent()` would get you the container that holds the G-code widget.

12.11.17. Read Command Line Load Time Options

Some panels need information at load time for setup/options. QtVCP covers this requirement with *-o* options.
The *-o* argument is good for a few, relatively short options, that can be added to the loading command line.
For more involved information, reading an INI or preference file is probably a better idea.

Multiple *-o* options can be used on the command line so you must decode them.
*self.w.USEROPTIONS_* will hold any found *-o* options as a list of strings. You must parse and define what is accepted and what to do with it.

Example code to get -o options for camera number and window size

```
    def initialized__(self):

        # set a default camera number
        number = 0

        # check if there are any -o options at all
        if self.w.USEROPTIONS_ is not None:
```

```
            # if in debug mode print the options to the terminal
            LOG.debug('cam_align user options: {}'.format(self.w.USEROPTIONS_))

            # go through the found options one by one
            for num, i in enumerate(self.w.USEROPTIONS_):

                # if the -o option has 'size=' in it, assume it's width and height of  ↩
                   window
                # override the default width and height of the window
                if 'size=' in self.w.USEROPTIONS_[num]:
                    try:
                        strg = self.w.USEROPTIONS_[num].strip('size=')
                        arg = strg.split(',')
                        self.w.resize(int(arg[0]),int(arg[1]))
                    except Exception as e:
                        print('Error with cam_align size setting:',self.w.USEROPTIONS_[num ↩
                            ])

                #  # if the -o option has 'camnumber=' in it, assume it's the camera number ↩
                     to use
                elif 'camnumber=' in self.w.USEROPTIONS_[num]:
                    try:
                        number = int(self.w.USEROPTIONS_[num].strip('camnumber='))
                    except Exception as e:
                        print('Error with cam_align camera selection - not a number - using ↩
                             0')

        # set the camera number either as default or if -o option changed the 'number'  ↩
            variable, to that number.
        self.w.camview._camNum = number
```

12.11.18. G-code to read Qt preferences

Here is how to create an O-word program to read a QtDragon preference file entry and add it as a G-code parameter.
Calling this O-word will update the param *toolToLoad*
This uses *Python hot comment* to communicate with the embedded python instance.
See the Remap section of the Documents for a description.

```
(filename myofile.ngc)
o<mi-archivo-o> sub

;py,from interpreter import *
;py,import os
;py,from qtvcp.lib.preferences import Access

; find and print the preference file path
;py,CONFPATH = os.environ.get('CONFIG_DIR', '/dev/null')
; adjust for your preference file name
;py,PREFFILE = os.path.join(CONFPATH,'qtdragon.pref')
;py,print(PREFFILE)

; get an preference instance
;py,Pref = Access(PREFFILE)

; load a preference and print it
;py,this.params['toolToLoad']=Pref.getpref('Tool to load', 0, int,'CUSTOM_FORM_ENTRIES')
;py,print('Tool to load->:',this.params['toolToLoad'])
```

```
; return the value
o<myofile> endsub [#<toolToLoad>]
M2
```

12.12. QtVCP Development

12.12.1. Descripción general

The intention of QtVCP is to **supply an infrastructure to support screen and VCP panel building for LinuxCNC**.

By providing a *diverse widget* set and supporting *custom coding*, QtVCP hopes that development energy will be expended in *one toolkit* rather than continuous re-invention.

By using the same toolkit across many screens/panels, users should have an easier time customizing/creating these, and developers should find it easier to help trouble shoot with less effort.

QtVCP uses a **Qt Designer built `.ui` file** and a **Python handler file**

- to **load and control a screen/panel that displays Qt widgets** and
- para **controlar el controlador de movimiento de LinuxCNC o pines HAL**.

There are *builtin screens and panels*, easily loaded by a user, or users can build/modify one of their own.

QtVCP uses **libraries and custom widgets** to hide some of the complexity of interfacing to LinuxCNC. By using QtVCP's library rather than LinuxCNC's, we can mitigate minor LinuxCNC code changes.

12.12.2. Builtin Locations

Builtin screens and panels are stored in separate folders:

- *Screens* in `share/qtvcp/screens`
- *Panels* in `share/qtvcp/panels`
- *Stock images* in `share/qtvcp/images`

Screens and panels are sorted by their folder name, which is also the name used to load them.

Inside the folder would be:

- the `.ui` *file*,
- el *archivo manejador*, y
- possibly the `.qss` *theme file*.

12.12.3. QtVCP Startup To Shutdown

QtVCP source is located in `+src/emc/usr_intf/qtvcp+` folder of LinuxCNC source tree.

12.12.3.1. QtVCP Startup

When QtVCP first starts:

1. It must decide if this object is a screen or a panel.
2. It searches for and collects information about paths of required files and useful folders.
3. It then:
    a. Builds the HAL component,
    b. Loads the window instance,
    c. Adds handler extensions,
    d. Installs an event filter.

Now the window/widgets are instantiated, the HAL pins are built. This also initiates the +_init_hal()+ function of the widgets. . Se llama a la función del manejador +initialized__()+ . The STATUS library is forced to update. . El componente HAL se establece como listo en este punto. . A variety of optional switch arguments are set, including calling a POSTGUI HAL file (if a screen). . Terminate signals are trapped and QtVCP now polls for events.

12.12.3.2. QtVCP Shutdown

Finally when QtVCP is asked to shutdown:

1. It calls shutdown functions in the handler file,
2. STATUS monitoring is shut down
3. HAL component gets killed

12.12.4. Path Information

When QtVCP loads it collects paths information.

This is available in the handler file's +__init__()+ function as **path**:

IMAGEDIR
: Path of builtin images

SCREENDIR
: Path of builtin motion controller screens

PANELDIR
: Path of builtin accessory panels

WORKINGDIR
: Path of where QtVCP was launched from

CONFIGPATH
: Path of the launched configuration

BASEDIR
: General path, used to derive all paths

BASENAME
: Generic name used to derive all paths

LIBDIR
Path of QtVCP's Python library

HANDLER
Path of handler file

XML
Path of .ui file

DOMAIN
Path of translation

IS_SCREEN
Screen/panel switch

12.12.5. Idiosyncrasies

These try to cover non-obvious situations.

12.12.5.1. Error Code Collecting

LinuxCNC's error code collecting can only be read from one place.

When read, it is ***consumed***, i.e. *no other object can read it*.

In QtVCP screens, it is recommended to *use the* `ScreenOptions` *widget to set up error reading*.

Errors are then ***sent to other objects*** *via* **`STATUS`** *signals*.

12.12.5.2. Velocidad de trote

LinuxCNC has no internal record of jog rate: *you must specify it at the time of jogging*.

QtVCP uses the `STATUS` library to *keep track of the latest linear and angular jog rates*.

It is **always specified in machine units per minute** and *must be converted when in non-machine units mode*.
So, if your machine is imperial based but you are in metric mode, changes to jog rate sent to `ACTION` functions must be converted to imperial.
In the same manner, if the machine is metric based and you are in imperial mode, changes to jog rate must be sent to `ACTION` functions in metric units.
For angular jog rates the units don't change in metric/imperial mode so you can send them to `ACTION` functions without conversion.

While you are free to ignore this jogging record while building screens, anyone modifying your screen and using the builtin jog rate widgets would not get the desired results as the `ACTION` library's **`DO_JOG`** function gets it's jog rate from the `STATUS` library.

12.12.5.3. Keybinding



aviso
Keybinding is always a *difficult-to-get-right-in-all-cases* affair.

Custom keybinding functions are to be *defined in the handler file*.

Most importantly widgets that require regular key input and not jogging, should be checked for in the `processed_key_event__` function.

12.12.5.4. Preference File

Some QtVCP widgets use the preference file to record important information.

This *requires the preference file to be set up early* in the widget initialization process.
The easiest way to do this is to **use the `ScreenOptions` widget**.

12.12.5.5. Widget Special Setup Functions

QtVCP looks for and calls the `+_hal_init()+` function *when the widget is first loaded*.

It is not called when using Qt Designer editor.

After this function is called the widget has access to some special variables:

`self.HAL_GCOMP`
: The *HAL component* instance

`self.HAL_NAME`
: This *widget's name* as a string

`self.QT_OBJECT_`
: This *widget's PyQt object instance*

`self.QTVCP_INSTANCE_`
: The *very top level parent* of the screen

`self.PATHS_`
: The *instance of QtVCP's path* library

`self.PREFS_`
: The *instance of an optional preference file*

`self.SETTINGS_`
: The `Qsettings` *object*

When making a custom widget, _import and sub class _the **`+_HalWidgetBase+`** class for this behavior.

12.12.5.6. Dialogs

Dialogs (AKA "pop up windows") are *best loaded with the* `ScreenOptions` *widget*, but they can be placed on the screen in Qt Designer.

It doesn't matter where on the layout but *to make them hidden*, cycle the `state` property to true then false.

By default, if there is a preference file, the dialogs will remember their last size/placement.
It is possible to override this so they open in the same location each time.

12.12.5.7. Styles (Themes)

While it is possible to set styles *in Qt Designer*, it is more convenient to change them later if they are all set in a ***separate `.qss` file***. The file should be put in the *same location as the handler file*.

Capítulo 13

Interfaces de usuario para Programación

13.1. Panelui

13.1.1. Introducción

Panelui is a non-realtime component to interface buttons to LinuxCNC or HAL:

- It decodes MESA 7I73 style key-scan codes and calls the appropriate routine.
- It gets input from a realtime component - sampler. Sampler gets its input from either the MESA 7I73 or sim_matrix_kb component.
- Panelui is configurable using an INI style text file to define button types, HAL pin types, and/or commands.
- It can be extended using a Python based *handler* file to add functions.

While actual input buttons are required to be momentary, Panelui will use this input to make toggle, radio, or momentary button output.

13.1.2. Loading Commands

The command used to load panelui (with optional -d debug switch):

```
loadusr -W panelui -d
```

This will initialize panelui, which will look for the INI file panelui.ini in the config folder or user folder.

Uno puede validar el archivo INI con este comando:

```
loadusr pyui
```

This will read, try to correct, then save the panelui.ini file. It will print errors to the terminal if found.

A typical HAL file will have these commands added:

```
# commands needed for panelui loading
#
# sampler is needed for panelui
# cfg= must always be u for panelui. depth sets the available buffer
loadrt sampler cfg=u depth=1025

#uncomment to validate the panelui INI file
#loadusr pyui

# -d = debug, -v = verbose debug
# -d will show you keypress identification and commands called
# -v is for develeper info
loadusr -W panelui -d

# using simulated buttons instead of the MESA 7I73 card
# so we load the sim_matrix_kb component to convert HAL pins to keyscan codes
loadrt sim_matrix_kb

# connect the components together.
# sampler talks to panelui internally
net key-scan    sim-matrix-kb.0.out
net key-scan    sampler.0.pin.0

# add panelui components to a thread

addf sim-matrix-kb.0    servo-thread
addf sampler.0          servo-thread
```

13.1.3. panelui.ini file reference

Key words

- KEY= This is used to designate the key that the button responds to. It can be NONE or ROW number and column number eg R1C2. A row and column can only be used once.
- OUTPUT= This sets the Button’s output type, eg S32, U32, FLOAT, BIT, NONE, COMMAND, ZMQ.
- DEFAULT= This sets the starting output of the group or button.
- GROUP= In radiobuttons, designates the group the button interacts with.
- GROUP_OUTPUT= sets the output the group pin will be, if this button is active.
- STATUS_PIN= If TRUE, a HAL pin will be added that reflects the current state of button.
- TRUE_STATE= sets the output the HAL pin will be, if button is TRUE.
- FALSE_STATE= sets the OUTPUT the HAL pin will be, if the button is FALSE.
- TRUE_COMMAND= sets the command and arguments to be called when the button is TRUE.
- FALSE_COMMAND= sets the command and arguments to be called when the button is FALSE.
- TRUE_FUNCTION= sets the ZMQ message function and arguments to be called when the button is TRUE.
- FALSE_FUNCTION= sets the ZMQ message function and arguments to be called when the button is FALSE.

HAL Prefix

```
[HAL_PREFIX]
    NAME= Yourname
```

This allows one to change the prefix of the HAL pins from *panelui* to an arbitrary name.

ZMQ Messaging Setup

```
[ZMQ_SETUP]
  TOPIC = 'QTVCP'
  SOCKET = 'tcp://127.0.0.1:5690'
  ENABLE = True
```

This sets up and enables ZMQ based messaging. TOPIC and SOCKET must match the listening program.

Radio Buttons Radiobutons allow only one button in the group to be active at a time. Each group has its own output pin, separate from each button in the group. Radio button definitions start with the text *RADIO_BUTTON* inside single brackets.

```
[RADIO_BUTTONS]
  # The double bracket section(s) define the group(s) of radio buttons.
  # The group name must be unique and is case sensitive.
  # Groups output is controlled by what button is active not directly by keycode.
  # DEFAULT references a button in the group by name and is case sensitive.
  [[group1_name]]
    KEY = NONE
    OUTPUT = FLOAT
    DEFAULT = small
    # The triple bracket sections define the buttons in this group.
    # button names must be unique and are case sensitive.
    # There must be at least two buttons in a group.
    #
    # This button, named 'small'is controller by the row 0 column 1 key.
    # It will cause the group output to be .0001 when it is pressed.
    # It has no output of its own, but has a status
    # pin which will follow its current state.
    # since this button is in a group, DEFAULT has no bearing.
    # since OUTPUT in not 'COMMAND' _COMMAND entries are ignored.
    [[[small]]]
      KEY = R0C1
      GROUP = group1_name
      GROUP_OUTPUT = .0001
      OUTPUT = NONE
      STATUS_PIN = True
      TRUE_STATE = TRUE
      FALSE_STATE = FALSE
      TRUE_COMMAND = NONE, NONE
      FALSE_COMMAND = NONE, NONE
      DEFAULT = false
    # This button, named 'large' is controller by the row 0 column 2 key.
    # It will cause the group output to be 1000 when it is pressed.
    # It has a S32 output of its own, will be 20 on true and 0 on false.
    # It also has a status pin which will follow its current state.
    # since this button is in a group, DEFAULT has no bearing.
    # since OUTPUT in not 'COMMAND' _COMMAND entries are ignored.
    [[[large]]]
      KEY = R0C2
      GROUP = group1_name
      GROUP_OUTPUT = 1000
      OUTPUT = S32
      STATUS_PIN = True
      TRUE_STATE = 20
```

```
      TRUE_COMMAND = NONE, NONE
      FALSE_COMMAND = NONE, NONE
      FALSE_STATE = 0
      DEFAULT = false
```

Toggle Buttons Togglebuttons only change state on each press of the button. Toggle button definitions start with the text *TOGGLE_BUTTON* inside single brackets.

```
[TOGGLE_BUTTONS]
  # Each button name inside double brackets, must be unique and is case sensitive.
  # This button, named 'tool_change'is controller by the row 2 column 5 key.
  # It has a BIT output, will output 1 on true state and 0 on false state.
  # It also has a status pin which will follow its current state.
  # DEFAULT sets this to true when first initialized.
  # The _COMMAND are not used since OUTPUT is not set to COMMAND but validation will
  # add the lines regardless
  [[tool_change]]
    KEY = R2C5
    OUTPUT = BIT
    TRUE_COMMAND = NONE, NONE
    FALSE_COMMAND = NONE, NONE
    STATUS_PIN = True
    DEFAULT = TRUE
    TRUE_STATE = 1
    FALSE_STATE = 0
```

Momentary Buttons Momentary buttons are true when pressed and false when released. Momentary button definitions start with the text *MOMENTARY_BUTTON* inside single brackets.

```
[MOMENTARY_BUTTONS]
  # Each button name inside double brackets, must be unique and is case sensitive.
  # This button, named 'spindle_rev'is controller by the row 2 column 3 key.
  # It has a COMMAND output, so will use TRUE_COMMAND and FALSE_COMMAND.
  # It also has a status pin which will follow its current state.
  # COMMANDs will have a command name and then any required arguments
  # This TRUE_COMMAND calls an internal command to start the spindle in reverse at 200 rpm
  # If the spindle is already started, it will increase the rpm.
  # DEFAULT is not used with Momentary buttons.
  # The _STATE are not used since OUTPUT is set to COMMAND but validation will
  # add the lines regardless
  [[spindle_rev]]
    KEY = R2C3
    OUTPUT = COMMAND
    TRUE_COMMAND = SPINDLE_REVERSE_INCREASE, 200
    FALSE_COMMAND = None, NONE
    STATUS_PIN = True
    DEFAULT = FALSE
    TRUE_STATE = 1
    FALSE_STATE = 0
```

13.1.4. Internal Command reference

There are a number of internal commands you may use.

home_selected

- required argument: axis number (int)

unhome_selected

- required argument: axis number (int)

spindle_forward_adjust

- optional argument: starting RPM (int) - default 100
- Description: If the spindle is stopped it will start in the forward direction. If it is already running it will increase or decrease the rpm depending on what direction the spindle is running in.

spindle_forward

- optional argument: starting RPM (int) - default 100

spindle_reverse

- optional argument: starting RPM (int) - default 100

spindle_reverse_adjust

- optional argument: starting RPM (int) - default 100
- Description: If the spindle is stopped it will start in the reverse direction. If it is already running it will increase or decrease the rpm depending on what direction the spindle is running in.

spindle_faster

- Description: increases spindle speed by 100 RPM

spindle_slower

- Description: decreases spindle speed by 100 RPM, until RPM is 100

set_linear_jog_velocity

- argumento requerido: velocidad en pulgadas por minuto (float)
- description: sets the jog velocity on axis 0,1,2,6,7,8 (X,Y,Z,U,V,W)

set_angular_jog_velocity

- required argument: velocity in degrees per minute (float)
- description: sets the jog velocity on axis 3,4,5 (A.B.C)

continuous_jog

- required arguments: axis number (int), direction (int)

incremental_jog

- required arguments: axis number (int), direction (int), distance (float)

quill_up

- optional arguments: machine Z axis absolute position (float)
- Description: Move Z axis to the given machine position

feed_hold

- required argument: state (bool 0 or 1)

feed_override

- required argument: rate (float)

rapid_override

- required argument: rate (float 0-1)

spindle_override

- required argument: rate (float)

max_velocity

- required argument: rate (float)

optional_stop

- required argument: state (bool 0 or 1)

block_delete

- required argument: state (bool 0 or 1)

single_block

- required argument: state (bool 0 or 1)

smart_cycle_start

- Description: If idle, starts G-code program, if paused runs one line.

re_start line

- required argument: line number (int)

mdi_and_return

- required argument: G-code command(s)
- Description: records the current mode, calls commands and then returns to mode.

mdi

- required argument: G-code command(s)
- Description: sets mode to MDI, calls commands.

13.1.5. ZMQ Messages

Panelui can send ZMQ based messages on button presses.
In this way panelui can interact will other programs such as QtVCP screens.

```
[TOGGLE_BUTTONS]
  [[zmq_test]]
    KEY = R2C3
    OUTPUT = ZMQ
    TRUE_FUNCTION = ZMQ_BUTTON, 200
    FALSE_FUNCTION = ZMQ_BUTTON, 0
    STATUS_PIN = False
    DEFAULT = FALSE
    TRUE_STATE = 1
    FALSE_STATE = 0
```

Here is a sample program that will receive the message and print it to the terminal.

```
import zmq
import json

# ZeroMQ Context
context = zmq.Context()

# Define the socket using the "Context"
sock = context.socket(zmq.SUB)

# Define subscription and messages with topic to accept.
topic = "" # all topics
sock.setsockopt(zmq.SUBSCRIBE, topic)
sock.connect("tcp://127.0.0.1:5690")

while True:
    topic, message = sock.recv_multipart()
    print('{} sent message:{}'.format(topic,json.loads(message)))
```

13.1.6. Handler File Extension

A special file can be used to add custom python code that will be available as commands. panelui_handler.py must be written in python and be placed in the configuration folder. If panelui finds a file there it will add its function calls to the available commands. Here is an example of a handler file that adds two functions - hello_world and cycle_mode:

```
# standard handler call - This will always be required
def get_handlers(linuxcnc_stat, linucnc_cmd, commands, master):
     return [HandlerClass(linuxcnc_stat, linucnc_cmd, commands, master)]

# Also required - handler class class HandlerClass:

    # This will be pretty standard to gain access to everything
    # linuxcnc_stat: is the python status instance of LinuxCNC
    # linuxcnc_cmd: is the python command instance of LinuxCNC
    # commands: is the command instance so one can call the internal routines
    # master: give access to the master functions/data

    def __init__(self, linuxcnc_stat, linuxcnc_cmd, commands, master):
        self.parent = commands
```

```
        self.current_mode = 0

    # command functions are expected to have this layout:
    # def some_name(self, widget_instance, arguments from widget):
    # widget_instance gives access to the calling widget's function/data
    # arguments can be a list of arguments, a single argument, or None
    # depending on what was given in panelui's INI file.
    def hello_world(self, wname, m):
        # print to terminal so we know it worked
        print('\nHello world\n')
        print(m)     # print the argument(s)
        print(wname.metadata)    # Print the calling widgets internal metadata (from config ↩
            file)

        # Call a mdi command to print a msg in LinuxCNC.
        # This requires LinuxCNC to be homed, but does not check for that.
        # parent commands expect a widget_instance - None is substituted
        self.parent.mdi(None,'(MSG, Hello Linuxcnc World!)')

    # Each call to this function will cycle the mode of LinuxCNC.
    def cycle_mode(self, wname, m):
        if self.current_mode == 0:
            self.current_mode = 1
            self.parent.set_mdi_mode()
        elif self.current_mode == 1:
            self.current_mode = 2
            self.parent.set_auto_mode()
        else:
            self.current_mode = 0
            self.parent.set_manual_mode()
        print(self.current_mode)

    # Código de calentamiento, a menudo requerido
    def __getitem__(self, item):
        return getattr(self, item)
    def __setitem__(self, item, value):
        return setattr(self, item, value)
```

13.2. El módulo Python de LinuxCNC

Esta documentación describe el módulo python `linuxcnc`, que proporciona una API Python para hablar con LinuxCNC.

13.2.1. Introducción

Las interfaces de usuario controlan la actividad de LinuxCNC enviando mensajes NML al controlador de tareas LinuxCNC y supervisan los resultados observando la estructura de estado de LinuxCNC, así como el canal de informe de errores.

El acceso programático a NML es a través de una API de C ++; sin embargo, las partes más importantes de la interfaz NML para LinuxCNC también están disponibles para programas de Python a través del módulo `linuxcnc`.

Más allá de la interfaz NML para los canales de comando, estado y error, el módulo `linuxcnc` también contiene:

- soporte para leer valores de archivos INI

13.2.2. Patrones de uso de la interfaz NML LinuxCNC

El patrón general para el uso de `linuxcnc` es, en lineas generales, así:

- Importar el módulo `linuxcnc`.
- Establecer conexiones a los canales NML de comandos, estado y errores según sea necesario.
- Sondear el canal de estado, ya sea periódicamente o según sea necesario.
- Antes de enviar un comando, determina desde el estado si esta OK para hacerlo (por ejemplo, no tiene sentido enviar un comando *Run* si la tarea esta en el estado ESTOP o el intérprete no está inactivo).
- Envía el comando utilizando uno de los métodos del canal de comandos `linuxcnc`.

Para recuperar mensajes del canal de errores, sondear el canal de errores periódicamente y procesar los mensajes recuperados.

- Sondear el canal de estado, ya sea periódicamente o según sea necesario.
- Imprimir cualquier mensaje de error y explorar el código de excepción.

El módulo `linuxcnc` también define el tipo de excepción Python `error` para admitir informes de errores.

13.2.3. Leer el estado de LinuxCNC con el módulo Python linuxcnc

Aquí hay un fragmento de Python para explorar el contenido del objeto `linuxcnc.stat` que contiene más de 80 valores (correr mientras LinuxCNC se está ejecutando para valores típicos):

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import sys
import linuxcnc
try:
    s = linuxcnc.stat() # crea una conexión al canal de estado
    s.poll() # obtener valores actuales
except linuxcnc.error, detail:
    print ”error”, detail
    sys.exit(1)
for x in dir(s):
    if not x.startswith(”_”):
        print x, getattr(s,x)
```

El módulo linuxcnc usa la ruta de compilación predeterminada para el archivo de configuración NML a menos que se sobreescriba; ver Lectura de valores de archivo INI para un ejemplo.

13.2.3.1. Atributos en linuxcnc.stat

acceleration
(devuelve float) - aceleración predeterminada, refleja la entrada INI [TRAJ]DEFAULT_ACCELERATION.

active_queue
(devuelve entero) - número de mezclas en motion.

actual_position
(devuelve tupla de floats) - posición de trayectoria actual, (x y z a b c u v w), en unidades máquina.

adaptive_feed_enabled
(devuelve boolean) - estado de porcentaje de avance adaptativo (0/1).

ain
(devuelve tupla de floats) - valor actual de los pines de entrada analógica.

angular_units
(devuelve float) - unidades angulares de máquina para grados, refleja el valor INI [TRAJ]ANGULAR_UNI

aout
(devuelve tupla de floats) - valor actual de los pines de salida analógica.

axes (*Eliminado desde la versión 2.9*)
en su lugar, usar `axis_mask.bit_count()` para obtener el número de ejes configurados.

axis
(devuelve tupla de diccionarios) - reflejando los valores actuales del eje. Ver El diccionario de ejes.

axis_mask
(devuelve entero) - máscara de ejes disponibles según lo definido por `[TRAJ]COORDINATES` en el archivo INI. Devuelve la suma de los ejes X=1, Y=2, Z=4, A=8, B=16, C=32, U=64, V=128, W=256.

block_delete
(devuelve boolean) - el estado actual de eliminar bloque.

call_level
(returns integer)` - profundidad de subrutina actual. - 0 si no es una subrutina, depth si de lo contrario no se especifica.

command
(devuelve cadena) - comando actualmente en ejecución.

current_line
(devuelve entero) - línea actualmente en ejecución.

current_vel
(devuelve float) - velocidad actual en unidades de usuario por segundo.

cycle_time
(devuelve float) - período de hilo

debug
(devuelve entero) - bandera de depuración desde el archivo INI.

delay_left
(devuelve float) - tiempo restante en el comando dwell (G4), segundos.

din
(devuelve tupla de enteros) - valor actual de los pines de entrada digital.

distance_to_go
(devuelve float) - distancia restante del movimiento actual, según lo informado por el planificador de trayectoria.

dout
(devuelve tupla de enteros) - valor actual de los pines de salida digital.

dtg
(devuelve tupla de floats) - distancia restante del movimiento actual para cada eje, según lo informado por el planificador de trayectoria.

echo_serial_number
(devuelve entero) - El número de serie del último comando completado enviado por una UI a task. Todos los comandos llevan un número de serie. Una vez que el comando ha sido ejecutado, su número de serie se refleja en `echo_serial_number`.

enabled
(devuelve boolean) - bandera de habilitación de planificador de trayectoria.

estop
(devuelve entero) - Devuelve STATE_ESTOP o no.

exec_state
(devuelve entero) - estado de ejecución task. Uno de EXEC_ERROR, EXEC_DONE, EXEC_WAITING_FOR
EXEC_WAITING_FOR_MOTION_QUEUE, EXEC_WAITING_FOR_IO, EXEC_WAITING_FOR_MOTION_A
EXEC_WAITING_FOR_DELAY, EXEC_WAITING_FOR_SYSTEM_CMD, EXEC_WAITING_FOR_SPINDLE_

feed_hold_enabled
(devuelve boolean) - habilitar la bandera para retención de alimentación.

feed_override_enabled
(devuelve boolean) - habilitar bandera para ajuste de alimentación.

feedrate
(devuelve float) - porcentaje de velocidad de avance actual, 1.0 = 100 %.

file
(devuelve cadena) - nombre de archivo de código G cargado actualmente, con ruta.

flood
(devuelve entero) - Estado de inundación, FLOOD_OFF o FLOOD_ON.

g5x_index
(devuelve entero) - sistema de coordenadas actualmente activo, G54=1, G55=2, etc.

g5x_offset
(devuelve tupla de floats) - offset del sistema de coordenadas actualmente activo.

g92_offset
(devuelve tupla de floats) - pose del offset g92 actual.

gcodes
(devuelve dupla de enteros) - códigos G activos para cada grupo modal
Los valores enteros reflejan los números de código G nominales multiplicados por 10. (Ejemplos: 10 = 'G1, 430 = *G43*, 923 = *G92.3*)

heartbeat
(returns integer) - motion controller heartbeat counter. Increments every servo cycle. Rate is determined by `[EMCMOT]SERVO_PERIOD` in the INI file.

homed
(devuelve tupla de enteros) - articulaciones actualmente en casa, con 0 = no en casa, 1 = en casa.

id
(devuelve entero) - ID de movimiento actualmente en ejecución.

ini_filename
(devuelve string) - ruta al archivo INI pasado a linuxcnc.

inpos
(devuelve boolean) - bandera machine-in-position.

input_timeout
(devuelve boolean) - bandera de temporizador M66 en progreso.

interp_state
(devuelve entero) - estado actual del intérprete RS274NGC. Uno de INTERP_IDLE, INTERP_READING, INTERP_PAUSED, INTERP_WAITING.

interpreter_errcode
(devuelve entero) - código de retorno actual del intérprete RS274NGC. Uno de INTERP_OK, INTERP_EXIT, INTERP_EXECUTE_FINISH, INTERP_ENDFILE, INTERP_FILE_NOT_OPEN, INTERP_ERROR. ver src/emc/nml_intf/interp_return.hh

joint
(devuelve tupla de diccionarios) - refleja los valores de articulación actuales. Ver El diccionario de articulaciones.

joint_actual_position
(devuelve tupla de floats) - posiciones articulares reales.

joint_position
(devuelve tupla de floats) - Posiciones articulares deseadas.

joints
(devuelve entero) - número de articulaciones. Refleja [KINS]JOINTS del INI.

kinematics_type
(devuelve entero) - El tipo de cinemática. Uno de:

- KINEMATICS_IDENTITY
- KINEMATICS_FORWARD_ONLY
- KINEMATICS_INVERSE_ONLY
- KINEMATICS_BOTH

limit
(devuelve tupla de enteros) - máscaras de límites de eje. minHardLimit = 1, maxHardLimit = 2, minSoftLimit = 4, maxSoftLimit = 8.

linear_units
(devuelve float) - unidades lineales de máquina por mm, refleja valor INI [TRAJ]LINEAR_UNITS.

max_acceleration
(devuelve float) - aceleración máxima. Refleja [TRAJ]MAX_ACCELERATION.

max_velocity
(devuelve float) - velocidad máxima. Refleja la velocidad máxima actual. Si no es modificada por halui.max-velocity o similar debe reflejar [TRAJ]MAX_VALOCITY.

mcodes
(devuelve tupla de 10 enteros) - códigos M actualmente activos.

mist
(devuelve entero) - Estado de la niebla, ya sea MIST_OFF o MIST_ON

motion_line
(devuelve entero) - número de línea fuente del movimiento que se está ejecutando actualmente. Relación con `id` no aclarada.

motion_mode
(devuelve entero) - Este es el modo del controlador de movimiento. Uno de TRAJ_MODE_COORD, TRAJ_MODE_FREE, TRAJ_MODE_TELEOP.

motion_type
(devuelve entero) - El tipo de movimiento que se está ejecutando actualmente. Uno de:

- MOTION_TYPE_TRAVERSE
- MOTION_TYPE_FEED
- MOTION_TYPE_ARC
- MOTION_TYPE_TOOLCHANGE
- MOTION_TYPE_PROBING
- MOTION_TYPE_INDEXROTARY
- 0 si no hay movimiento en este momento.

optional_stop
(devuelve entero) - bandera de paro opcional.

paused
(devuelve boolean) - bandera de `movimiento en pausa`.

single_stepping
(devuelve boolean) - bandera de `movimiento de paso simple`.

pocket_prepped
(devuelve entero) - un comando Tx completado, y la ranura está preparada. -1 si la ranura no está preparada.

poll()
-*(función incorporada)* método para actualizar los atributos del estado actual.

position
(devuelve tupla de floats) - posición en trayectoria.

probe_tripped
(devuelve boolean) - bandera, True si la sonda se ha disparado (enclave).

probe_val
(devuelve entero) - refleja el valor del pin `motion.probe-input`.

probed_position
(devuelve tupla de floats) - posición donde se disparó la sonda.

probing
(devuelve booleano) - bandera, True si hay una operación de sonda en progreso.

program_units
(devuelve entero) - uno de CANON_UNITS_INCHES=1 , CANON_UNITS_MM=2, CANON_UNITS_CM=

queue
(devuelve entero) - tamaño actual de la cola del planificador de trayectoria.

queue_full
(devuelve boolean) - la cola del planificador de trayectoria está llena.

rapidrate
(devuelve float) - escala de ajustes de rápidos.

read_line
(devuelve entero) - línea que el intérprete RS274NGC está leyendo actualmente.

rotation_xy
(devuelve float) - ángulo de rotación XY actual alrededor del eje Z.

settings
(devuelve tupla de floats) - configuración actual del intérprete:
settings[0] = número de secuencia,
settings[1] = velocidad de alimentación,
settings[2] = velocidad,
settings[3] = `G64 P` tolerancia de mezcla,
settings[4] = `G64 Q` tolerancia CAM simple.

spindle
(devuelve tupla de diccionarios) - devuelve el estado actual del husillo ver <sec:the-spindle-dictionary, El diccionario spindle>>.

spindles
(devuelve entero) - Número de husillos. Refleja el valor INI `[TRAJ]SPINDLES`.

state
(devuelve entero) - estado actual de ejecución del comando. Uno de `RCS_DONE`, `RCS_EXEC`, `RCS_ERROR`.

task_mode
(devuelve entero) - modo de tarea actual. Uno de `MODE_MDI`, `MODE_AUTO`, `MODE_MANUAL`.

task_paused
(devuelve entero) - bandera de tarea pausada.

task_state
(devuelve entero) - estado actual de la tarea. Uno de `STATE_ESTOP`, `STATE_ESTOP_RESET`, `STATE_ON`, `STATE_OFF`.

tool_in_spindle
(devuelve entero) - número de herramienta actual.

taskbeat
(returns integer) - task main loop heartbeat counter. Increments every task cycle. Rate is determined by `[TASK]CYCLE_TIME` in the INI file.

tool_from_pocket
(devuelve entero) - número de cavidad para la herramienta cargada actualmente (0 si no hay herramienta cargada).

tool_offset
(devuelve tupla de floats) - valores de offsets de la herramienta actual.

tool_table
(devuelve tupla de tool_results) - lista de entradas de herramientas. Cada entrada es una secuencia de los siguientes campos: id, xoffset, yoffset, zoffset, aoffset, boffset, coffset, uoffset, voffset, woffset, diameter, frontangle, backangle, orientation. La identificación y orientación son enteros y el resto son floats.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
s = linuxcnc.stat()
s.poll()
# para encontrar la información de la herramienta cargada en el índice 0 de la tabla de ↩
    herramientas
if s.tool_table[0].id != 0: # se carga una herramienta
    print(s.tool_table[0].zoffset)
else:
    print(”No hay herramienta cargada”)
```

toolinfo(*toolno*)
(devuelve un diccionario de tooldata para toolno) - se requiere un stat.poll() inicial para inicializar. *toolno* debe ser mayor que cero y menor o igual al número de herramienta más grande en uso. Los elementos del diccionario incluyen todos los datos de herramienta: *toolno*, *pocketno*, *diameter*, *frontangle*, *backangle*, *orientation*, *xoffset*, *yoffset*, ... *woffset*, *comment*.
Como ejemplo, el script siguiente

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
s = linuxcnc.stat()
s.poll()
toolno = 1
print(s.toolinfo(toolno))
```

produce la salida:

```
': 0, 'xoffset': 0.0, 'yoffset': 0.0, 'zoffset': 0.18, 'aoffset': 0.0, 'boffset': 0.0, ' ↩
    coffset': 0.0, 'uoffset': 0.0, 'voffset': 0.0, 'woffset': 0.0, 'comment': 'Tool_18 28 ↩
    Jan23:18.53.25'}
```

velocity
(devuelve float) - Esta propiedad está definida, pero no tiene una interpretación útil.

13.2.3.2. El diccionario `axis`

La configuración de ejes y los valores de estado están disponibles a través de una lista de diccionarios por eje. Aquí hay un ejemplo de cómo acceder a un atributo de un eje particular: Tenga en cuenta que muchas propiedades que anteriormente estaban en el diccionario `axis` están ahora en el diccionario `joint`, porque en máquinas con cinemáticas no triviales estos elementos (como el backlash) no son propiedades de un eje.

max_position_limit
(devuelve float) - límite máximo (límite suave) para el movimiento del eje, en unidades máquina del parámetro units.configuration, refleja `[JOINT__n__]MAX_LIMIT`.

min_position_limit
(devuelve float) - límite mínimo (límite suave) para el movimiento del eje, en unidades máquina del parámetro units.configuration, refleja `[JOINT__n__]MIN_LIMIT`.

velocity
(devuelve float) - velocidad actual.

13.2.3.3. El diccionario `joint`

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
s = linuxcnc.stat()
s.poll()
print("Articulación 1 en casa: ", s.joint[1]["homed"])
```

Para cada articulación, están disponibles las siguientes claves de diccionario:

backlash
(devuelve float) - holgura mecánica en unidades máquina. parámetro de configuración, refleja `[JOINT___n__]BACKLASH`.

enabled
(devuelve entero) - no cero significa habilitada.

fault
(devuelve entero) - no cero significa fallo del amplificador del eje.

ferror_current
(devuelve float) - error de seguimiento actual.

ferror_highmark
(devuelve float) - magnitud del error de seguimiento máximo.

homed
(devuelve entero) - distinto de cero significa que está en casa.

homing
(devuelve entero) - distinto de cero significa homing en progreso.

inpos
(devuelve entero) - no cero significa en posición.

input
(devuelve float) - posición de entrada actual.

jointType
(devuelve entero) - parámetro de configuración de tipo de eje, refleja `[JOINT___n__]TYPE` con `LINEAR=1`, `ANGULAR=2`. Ver Configuración INI de articulación para más detalles.

max_ferror
(devuelve float) - máximo error de seguimiento. parámetro de configuración, refleja `[JOINT___n__]FERR`

max_hard_limit
(devuelve entero) - distinto de cero significa que se ha excedido el límite duro máximo.

max_position_limit
(devuelve float) - límite máximo (límite suave) para el movimiento articular, en unidades de máquina. parámetro de configuración, refleja `[JOINT___n__]MAX_LIMIT`.

max_soft_limit
distinto de cero significa que se ha excedido `max_position_limit`, int

min_ferror
(devuelve float) - parámetro de configuración, refleja `[JOINT___n__]MIN_FERROR`.

min_hard_limit
(devuelve entero) - distinto de cero significa que se excedió el límite duro mínimo.

min_position_limit
(devuelve float) - límite mínimo (límite suave) para el movimiento articular, en unidades de máquina. parámetro de configuración, refleja `[JOINT___n__]MIN_LIMIT`.

min_soft_limit
(devuelve entero) - distinto de cero significa que se ha excedido `min_position_limit`.

output
(devuelve float) - posición de salida ordenada.

override_limits
(devuelve entero) - no cero significa que los límites se han ajustado.

units
(devuelve float) - unidades de articulación por mm, o por grado para articulaciones angulares. (las unidades de articulación son las mismas que las unidades de máquina, a menos que se establezca lo contrario por el parámetro de configuración `[JOINT___n__]UNITS`)

velocity
(devuelve float) - velocidad actual.

13.2.3.4. El diccionario `spindle`

brake
(devuelve entero) - valor de la bandera del freno del husillo.

direction
(devuelve entero) - dirección rotacional del huso con adelante=1, atrás=-1.

enabled
(devuelve entero) - valor de la bandera de husillo habilitado.

homed
(no implementado actualmente)

increasing
(devuelve entero) - no es claro.

orient_fault
(devuelve entero)

orient_state
(devuelve entero)

override
(devuelve float) - escala de ajuste de velocidad del husillo.

override_enabled
(devuelve boolean) - valor de la bandera de ajuste del husillo habilitado.

speed
(devuelve float) - valor de velocidad del husillo, rpm, > 0: sentido horario, < 0: sentido antihorario.
Con G96 activo refleja la velocidad máxima establecida por la palabra *D* opcional de G96, o si se omite la palabra *D*, los valores predeterminados +/-1e30.

13.2.4. Preparación para enviar comandos

Algunos comandos pueden enviarse siempre, independientemente del modo y el estado; por lo tanto, siempre se puede llamar al método `linuxcnc.command.abort()`.

Otros comandos pueden enviarse solo en el estado apropiado, y probarlos puede ser un poco complicado. Por ejemplo, un comando MDI solo se puede enviar si:

- ESTOP no se ha activado, y
- la máquina está encendida y
- los ejes están en casa y
- el intérprete no está funcionando y

- el modo está configurado en `modo MDI`

Una prueba apropiada antes de enviar un comando MDI a través de `linuxcnc.command.mdi()` podría ser:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
s = linuxcnc.stat()
c = linuxcnc.command()

def ok_for_mdi():
    s.poll()
    return not s.estop and s.enabled and (s.homed.count(1) == s.joints) and (s.interp_state ↩
        == linuxcnc.INTERP_IDLE)

if ok_for_mdi():
    c.mode(linuxcnc.MODE_MDI)
    c.wait_complete() # espera hasta que se ejecute el cambio de modo (¡o se agote el  ↩
       tiempo de espera predeterminado de 5s!)
    c.mdi("G0 X10 Y20 Z30")
```



aviso
Leer información importante sobre *wait_complete()* en la sección 'métodos de linuxcnc.command' abajo.

13.2.5. Enviar comandos a través de `linuxcnc.command`

Antes de enviar un comando, inicializar un canal de comando así:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
c = linuxcnc.command()

# Ejemplos de uso para algunos de los comandos enlistados a continuación:
c.abort()

c.auto(linuxcnc.AUTO_RUN, program_start_line)
c.auto(linuxcnc.AUTO_STEP)
c.auto(linuxcnc.AUTO_PAUSE)
c.auto(linuxcnc.AUTO_RESUME)

c.brake(linuxcnc.BRAKE_ENGAGE)
c.brake(linuxcnc.BRAKE_RELEASE)

c.flood(linuxcnc.FLOOD_ON)
c.flood(linuxcnc.FLOOD_OFF)

c.home(2)

c.jog(linuxcnc.JOG_STOP,        jjogmode, joint_num_or_axis_index)
c.jog(linuxcnc.JOG_CONTINUOUS,  jjogmode, joint_num_or_axis_index, velocity)
c.jog(linuxcnc.JOG_INCREMENT,   jjogmode, joint_num_or_axis_index, velocity, increment)

c.load_tool_table()
```

```
c.maxvel(200.0)

c.mdi("G0 X10 Y20 Z30")

c.mist(linuxcnc.MIST_ON)
c.mist(linuxcnc.MIST_OFF)

c.mode(linuxcnc.MODE_MDI)
c.mode(linuxcnc.MODE_AUTO)
c.mode(linuxcnc.MODE_MANUAL)

c.override_limits()

c.program_open("foo.ngc")
c.reset_interpreter()

c.tool_offset(toolno, z_offset,  x_offset, diameter, frontangle, backangle, orientation)
```

13.2.5.1. Atributos de `linuxcnc.command`

`serial`
el número de serie del comando actual

13.2.5.2. Métodos de `linuxcnc.command`:

`abort()`
enviar mensaje EMC_TASK_ABORT.

`auto(int [, int])`
ejecutar, dar un paso, pausar o reanudar un programa.

`brake(int)`
poner o soltar el freno del husillo.

`debug(int)`
establecer el nivel de depuración a través del mensaje EMC_SET_DEBUG.

`display_msg(string)`
envía un mensaje de despliegue al operador a la pantalla. (254 caracteres máx.)

`error_msg(string)`
envía un mensaje de error de operador en la pantalla. (254 caracteres máx.)

`feedrate(float)`
asigna el ajuste de alimentación, 1.0 = 100 %.

`flood(int)`
encender/apagar inundación.

Sintaxis
flood(command)
flood(linuxcnc.FLOOD_ON)
flood(linuxcnc.FLOOD_OFF)

Constantes
FLOOD_ON
FLOOD_OFF

home(int)
lleva a casa una articulación determinada.

jog(command-constant, bool, int[, float[, float]])

Sintaxis
jog(command, jjogmode, joint_num_or_axis_index, velocity[, distance])
jog(linuxcnc.JOG_STOP, jjogmode, joint_num_or_axis_index)
jog(linuxcnc.JOG_CONTINUOUS, jjogmode, joint_num_or_axis_index, velocity)
jog(linuxcnc.JOG_INCREMENT, jjogmode, joint_num_or_axis_index, velocity, distance)

Constantes de comando
linuxcnc.JOG_STOP
linuxcnc.JOG_CONTINUOUS
linuxcnc.JOG_INCREMENT

jjogmode

True
solicita trote de articulación individual (requiere teleop_enable(0))

False
solicita trote de eje de coordenadas cartesianas (requiere teleop_enable(1))

joint_num_or_axis_index

Para trote articular (jjogmode=1)
joint_number

Para trote de eje de coordenadas cartesianas (jjogmode=0)
índice de base cero de la coordenada del eje con respecto a las letras de coordenadas conocidas XYZABCUVW (x=>0,y=>1,z=>2,a=>3,b=>4,c=>5,u=>6,v=>7,w=>8)

load_tool_table()
vuelve a cargar la tabla de herramientas.

maxvel(float)
establece la velocidad máxima

mdi(string)
envía un comando MDI. Máximo 254 caracteres.

mist(int)
activa/desactiva la niebla.

Sintaxis
mist(command)
mist(linuxcnc.MIST_ON)
mist(linuxcnc.MIST_OFF)

Constantes
MIST_ON
MIST_OFF

mode(int)
establece modo (MODE_MDI, MODE_MANUAL, MODE_AUTO).

override_limits()
establece la bandera de ajuste de límites de eje.

program_open(string)
abre un archivo NGC.

rapidrate()
establece factor de ajuste de rápidos

reset_interpreter()
restablece el intérprete RS274NGC

set_adaptive_feed(int)
establece la bandera de alimentación adaptativa

set_analog_output(int, float)
establece el pin de salida analógica al valor

set_block_delete(int)
establece la bandera de eliminado de bloque

set_digital_output(int, int)
establece el pin de salida digital al valor

set_feed_hold(int)
activa/desactiva la retención de alimentación

set_feed_override(int)
activa/desactiva el ajuste de alimentación

set_max_limit(int, float)
establece el límite de posición máxima para un eje dado

set_min_limit()
establece el límite de posición mínima para un eje dado

set_optional_stop(int)
activa/desactiva parada opcional

set_spindle_override(int [, int])
establece ajuste del husillo habilitado. Predeterminado al husillo 0.

spindle(dirección: int, velocidad: float=0, husillo: int=0, esperar_velocidad: int=0)

- dirección: [SPINDLE_FORWARD, SPINDLE_REVERSE, SPINDLE_OFF, SPINDLE_INCREASE, SPINDLE_DECREA o SPINDLE_CONSTANT]
- velocidad: en RPM, predeterminada a 0.
- husillo: Número de husillo a ordenar, predeterminado a 0.
- esperar_velocidad: si es 1 motion esperará la velocidad antes de continuar, predeterminado a no esperar.



aviso
Los comandos MDI ignorarán esto. ”S1000” apagará el husillo después de esto.

text_msg(string)
envía un mensaje de texto al operador en la pantalla (máximo 254 caracteres).

```
#!/usr/bin/env python3
import linuxcnc
c = linuxcnc.command()

# Aumentar la velocidad del husillo 0 en 100 rpm. El husillo debe estar encendido primero
c.spindle(linuxcnc.INCREASE)
```

```
# Aumentar la velocidad del husillo 2 en 100 rpm. El husillo debe estar encendido primero
c.spindle(linuxcnc.SPINDLE_INCREASE, 2)

# Establece la velocidad del husillo de 0 a 1024 rpm
c.spindle.(linuxcnc.SPINDLE_FORWARD, 1024)

# Establece la velocidad del husillo 1 a -666 rpm
c.spindle.(linuxcnc.SPINDLE_REVERSE, 666, 1)

# Detiene el husillo 0.
c.spindle.(linuxcnc.SPINDLE_OFF)

# Detiene el husillo 0 explícitamente.
c.spindle.(linuxcnc.SPINDLE_OFF, 0)
```

spindleoverride(float [, int])
Establece el factor de ajuste del husillo. Predeterminado al husillo 0.

state(int)
Establece el estado de la máquina. El estado de la máquina debe ser STATE_ESTOP, STATE_ESTOP_RESET, STATE_ON o STATE_OFF.

task_plan_sync()
Al finalizar esta llamada, el archivo VAR en el disco se actualiza con valores vivos del intérprete.

teleop_enable(int)
Habilita/deshabilita el modo teleop (deshabilitado para trote articular).

tool_offset(int, float, float, float, float, float, int)
Establece el offset de herramienta. Ver ejemplo de uso arriba.

traj_mode(int)
Establece el modo trayectoria. El modo es uno de MODE_FREE, MODE_COORD o MODE_TELEOP.

unhome(int)
Deja sin casa una articulación determinada.

wait_complete([float])
Espera a que se complete el último comando. Recibe un tiempo de espera opcional en segundos.
Si se omite, el tiempo de espera predeterminado es de 5 segundos.
Devuelve -1 si se agota el tiempo de espera.
Devuelve RCS_DONE o RCS_ERROR según el estado de ejecución del comando.
Nótese que la ejecución de Python se bloqueará hasta que esta función regrese.

13.2.6. Lectura del canal de error

Para manejar mensajes de error, conectarse al canal de error y ejecutar periódicamente poll().

Ten en cuenta que el canal NML para mensajes de error tiene una cola (distinta a los canales de comando y estado), lo que significa que el primer consumidor de un mensaje de error elimina ese mensaje de la cola; el que tu otro consumidor de mensajes de error (p. ej. AXIS) ”vea” el mensaje depende del momento. Se recomienda tener solo una tarea lectora del canal de error en una configuración.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
e = linuxcnc.error_channel()

error = e.poll()
```

```
if error:
    kind, text = error
    if kind in (linuxcnc.NML_ERROR, linuxcnc.OPERATOR_ERROR):
        typus = "error"
    else:
        typus = "info"
    print typus, text
```

13.2.7. Lectura de valores de archivo INI

Aquí hay un ejemplo para leer valores de un archivo INI a través del objeto `linuxcnc.ini`:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# run as:
# python ini-example.py ~/emc2-dev/configs/sim/axis/axis_mm.ini

import sys
import linuxcnc

inifile = linuxcnc.ini(sys.argv[1])

# inifile.find () devuelve None si no se encontró la clave
# la siguiente expresión es útil para establecer un valor predeterminado:

machine_name = inifile.getstring("EMC", "MACHINE", fallback="unknown")
print("machine name: ", machine_name)

# inifile.findall () devuelve una lista de coincidencias o una lista vacía
# si no se encontró la clave:

extensions = inifile.findall("FILTER", "PROGRAM_EXTENSION")
print ("extensiones: ", extensions)

# override default NML file by INI parameter if given
nmlfile = inifile.getstring("EMC", "NML_FILE", fallback="")
if nmlfile:
    linuxcnc.nmlfile = os.path.join(os.path.dirname(sys.argv[1]), nmlfile)

# Other examples:
realval = inifile.getreal("AXIS_X", "MAX_VELOCITY", fallback=5.0)
boolval = inifile.getbool("JOINT_0", "HOME_USE_INDEX", fallback=False)

# None is returned without fallback= if the variable was not found
intval  = inifile.getint( "KINS", "JOINTS")
if None == intval:
    print("Error: [KINS]JOINTS not defined or an invalid integer"
```

O para el mismo archivo INI de LinuxCNC:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# ejecutar así:
# python3 ini-example2.py

import linuxcnc

stat = linuxcnc.stat()
stat.poll()
```

```
inifile = linuxcnc.ini(stat.ini_filename)

# Ver el ejemplo de arriba para el uso del objeto 'inifile'
```

13.2.8. El tipo `linuxcnc.positionlogger`

Algunas sugerencias de uso se pueden obtener de `src/emc/usr_intf/gremlin/gremlin.py`.

13.2.8.1. miembros

`npts`
: número de puntos.

13.2.8.2. métodos

`start(float)`
: iniciar el registrador de posición y ejecutar cada ARG segundos

`clear()`
: borrar el registrador de posición

`stop()`
: detener el registrador de posición

`call()`
: dibuja el trazo posterior ahora.

`last([int])`
: devuelve el punto más reciente de la trazo o None

13.3. El módulo Python HAL

Esta documentación describe el módulo python `hal`, que proporciona una API Python para crear y acceder pines y señales HAL.

13.3.1. Uso básico

```
#!/usr/bin/env python3
import hal, time
h = hal.component("passthrough")
h.newpin("in", hal.HAL_FLOAT, hal.HAL_IN)
h.newpin("out", hal.HAL_FLOAT, hal.HAL_OUT)
h.ready()
```

13.3.2. Funciones

component
+
El componente en sí es creado por una llamada al constructor `hal.component`. Los argumentos son el nombre del componente HAL y (opcionalmente) el prefijo utilizado para los nombres de los pines y los parámetros. Si el prefijo no se especifica, se usa el nombre del componente.
.Ejemplo

```
h = hal.component("passthrough")
```

newpin
+
Crea pin nuevo.
Argumentos: sufijo del nombre del pin, tipo de pin y dirección del pin. Para parámetros, los argumentos son: sufijo del nombre del parámetro, tipo de parámetro, y dirección del parámetro.
.Ejemplo:

```
h.newpin("in", hal.HAL_FLOAT, hal.HAL_IN)
```

ready
Le indica al sistema HAL que el componente esta inicializado. Bloquea la adición de pines.

unready
Permite al componente agregar pines después de llamar a *ready()*. Uno debería llamar a *ready()* del componente después.

component_exists
Existencia del componente especificado en este momento.

Ejemplo

```
hal.component_exists("testpanel")
```

component_is_ready
Componente especificado listo en este momento.

Ejemplo

```
hal.component_is_ready("testpanel")
```

get_msg_level
Obtiene el nivel de mensaje de tiempo real actual.

set_msg_level
Establece el nivel de mensaje de tiempo real. usado para información de depuración.

connect
Conecta un pin a una señal.

Ejemplo

```
hal.connect("pinname","nombre_de_señal")
```

disconnect
Desconecta un pin de una señal.

Ejemplo

```
hal.disconnect("nombredelpin")
```

get_value
Lee un pin, parámetro o señal directamente.

Ejemplo

```
value = hal.get_value("iocontrol.0.emc-enable-in")
```

get_info_pins()
Devuelve una lista de diccionarios de todos los pines del sistema.

```
listOfDicts = hal.get_info_pins()
pinName1 = listOfDicts[0].get('NAME')
pinValue1 = listOfDicts[0].get('VALUE')
pinType1 = listOfDicts[0].get('TYPE')
pinDirection1 = listOfDicts[0].get('DIRECTION')
```

get_info_signals()
Devuelve una lista de diccionarios de todas las señales del sistema.

```
listOfDicts = hal.get_info_signals()
signalName1 = listOfDicts[0].get('NAME')
signalValue1 = listOfDicts[0].get('VALUE')
driverPin1 = listOfDicts[0].get('DRIVER')
```

get_info_params()
Devuelve una lista de diccionarios de todos los parámetros del sistema.

```
listOfDicts = hal.get_info_pins()
pinName1 = listOfDicts[0].get('NAME')
pinValue1 = listOfDicts[0].get('VALUE')
pinDirection1 = listOfDicts[0].get('DIRECTION')
```

new_sig
Crea una nueva señal del tipo especificado.

Ejemplo

```
hal.new_sig("signalname",hal.HAL_BIT)
```

pin_has_writer
El pin especificado tiene un pin de manejo conectado.
Devuelve verdadero o falso.

```
h.in.pin_has_writer()
```

get_name
Obtener nombre de objeto HAL.
Devuelve una cadena.

```
h.in.get_name()
```

get_type
Obtiene el tipo de objeto HAL.
Devuelve un entero.

```
h.in.get_type()
```

get_dir
Obtiene el tipo de dirección de objeto HAL.
Devuelve un entero.

```
h.in.get_dir()
```

get
Obtener el valor del objeto HAL.

```
h.in.get()
```

set
Asigna el valor del objeto HAL.

```
h.out.set(10)
```

is_pin
El objeto ¿es pin o parámetro?
Devuelve verdadero o falso.

```
h.in.is_pin()
```

sampler_base
TODO

stream_base
TODO

stream
TODO

set_p
Establece un valor en cualquier pin en el sistema HAL.

Ejemplo

```
hal.set_p(”nombredepin”,”10”)
```

set_s
Establece un valor de cualquier señal sin conexión en el sistema HAL.

Ejemplo

```
hal.set_s(”nombredeseñal”,”10”)
```

13.4. GStat Python Module

13.4.1. Intro

GStat is a Python class used to send messages from LinuxCNC to other Python programs. It uses GObject to deliver messages, making it easy to listen for specific information. This is referred to as event-driven programming, which is more efficient then every program polling LinuxCNC at the same time. GladeVCP, Gscreen, Gmoccapy and QtVCP use GStat extensively. GStat is in the `hal_glib` module.

Descripción general

- First, a program imports the `hal_glib` module and instantiates GStat.
- Then it *connects* to the messages it wishes to monitor.
- GStat checks LinuxCNC’s status every 100 ms and if there are differences from the last check, it will send a callback message to all the connected programs with the current status.
- When GStat calls the registered function, it sends the GStat object plus any return codes from the message.

Typical code signatures:

```
GSTAT.connect('MESSGAE-TO-LISTEN-FOR', FUNCTION_TO_CALL)

def FUNCTION_TO_CALL(gstat_object, return_codes):
```

Often LAMBDA is used to strip the GSTAT object and manipulate the return codes:

```
GSTAT.connect('MESSGAE-TO-LISTEN-FOR', lambda o, return: FUNCTION_TO_CALL(not return))

def FUNCTION_TO_CALL(return_codes):
```

13.4.2. Sample GStat Code

There are some basic patterns for using GStat, depending on what library you are using them in. If using GStat with GladeVCP, Gscreen, or QtVCP, the GObject library is not needed as those toolkits already set up GObject.

13.4.2.1. Sample HAL component code pattern

This program creates two HAL pins that output the status of G20/G21.

```
#!/usr/bin/env python3

import gi
gi.require_version('Gtk', '3.0')
from gi.repository import GObject
from gi.repository import GLib
import hal
from hal_glib import GStat
GSTAT = GStat()

# callback to change HAL pin state
def mode_changed(obj, data):
        h['g20'] = not data
        h['g21'] = data

# Hace un componente y pines
h = hal.component("metric_status")
h.newpin("g20", hal.HAL_BIT, hal.HAL_OUT)
h.newpin("g21", hal.HAL_BIT, hal.HAL_OUT)
h.ready()

# connect a GSTAT message to a callback function
GSTAT.connect("metric-mode-changed",mode_changed)

# force GSTAT to initialize states
GSTAT.forced_update()

# loop till exit
try:
    GLib.MainLoop().run()
except KeyboardInterrupt:
    raise SystemExit
```

This would be loaded with `loadusr python PATH-TO-FILE/FILENAME.py` or if you need to wait for the pins to be made before continuing:
`loadusr python -Wn metric_status PATH-TO-FILE/FILENAME.py`
The pins would be: `metric_status.g20` and `metric_status.g21`.

13.4.2.2. GladeVCP Python extension code pattern

This file assumes there are three GTK labels named:

- *state_label*
- *e_state_label*
- *interp_state_label*

```
#!/usr/bin/env python3

from hal_glib import GStat
GSTAT = GStat()

class HandlerClass:

    def __init__(self, halcomp, builder, useropts):
```

```
        self.builder = builder

        GSTAT.connect("state-estop",lambda w: self.update_estate_label('ESTOP'))
        GSTAT.connect("state-estop-reset",lambda w: self.update_estate_label('RESET'))

        GSTAT.connect("state-on",lambda w: self.update_state_label('MACHINE ON'))
        GSTAT.connect("state-off",lambda w: self.update_state_label('MACHINE OFF'))

        GSTAT.connect("interp-paused",lambda w: self.update_interp_label('Paused'))
        GSTAT.connect("interp-run",lambda w: self.update_interp_label('Run'))
        GSTAT.connect("interp-idle",lambda w: self.update_interp_label('Idle'))

    def update_state_label(self,text):
        self.builder.get_object('state_label').set_label("State: %s" % (text))

    def update_estate_label(self,text):
        self.builder.get_object('e_state_label').set_label("E State: %s" % (text))

    def update_interp_label(self,text):
        self.builder.get_object('interp_state_label').set_label("Interpreter State: %s" % ( ↩
            text))

def get_handlers(halcomp,builder,useropts):
    return [HandlerClass(halcomp,builder,useropts)]
```

13.4.2.3. QtVCP Python extension code pattern

QtVCP extends GStat, so must be loaded differently but all the messages are available in QtVCP. This handler file assumes there are three QLabels named:

- *state_label*
- *e_state_label*
- *interp_state_label*

```
#!/usr/bin/env python3

from qtvcp.core import Status
GSTAT = Status()

class HandlerClass:

    def __init__(self, halcomp,widgets,paths):
        self.w = widgets

        GSTAT.connect("state-estop",lambda w: self.update_estate_label('ESTOP'))
        GSTAT.connect("state-estop-reset",lambda w: self.update_estate_label('RESET'))

        GSTAT.connect("state-on",lambda w: self.update_state_label('MACHINE ON'))
        GSTAT.connect("state-off",lambda w: self.update_state_label('MACHINE OFF'))

        GSTAT.connect("interp-paused",lambda w: self.update_interp_label('Paused'))
        GSTAT.connect("interp-run",lambda w: self.update_interp_label('Run'))
        GSTAT.connect("interp-idle",lambda w: self.update_interp_label('Idle'))

    def update_state_label(self,text):
        self.w.state_label.setText("State: %s" % (text))
```

```
    def update_estate_label(self,text):
        self.w.e_state_label.setText("E State: %s" % (text))

    def update_interp_label(self,text):
        self.winterp_state_label.setText("Interpreter State: %s" % (text))

def get_handlers(halcomp,builder,useropts):
    return [HandlerClass(halcomp,widgets,paths)]
```

13.4.3. Messages

periodic
(devuelve nada) - enviado cada 100 ms.

state-estop
(returns nothing) - Sent when LinuxCNC is goes into estop.

state-estop-reset
(returns nothing) - Sent when LinuxCNC comes out of estop.

state-on
(returns nothing) - Sent when LinuxCNC is in machine on state.

state-off
(returns nothing) - Sent when LinuxCNC is in machine off state.

homed
(returns string) - Sent as each joint is homed.

all-homed
(returns nothing) - Sent when all defined joints are homed.

not-all-homed
(returns string) - Sends a list of joints not currently homed.

override_limits_changed
(returns string) - Sent if LinuxCNC has been directed to override its limits.

hard-limits-tripped
(returns bool, Python List) - Sent when any hard limit is tripped. bool indicates if any limit is tripped, the list shows all available joint's current limit values.

mode-manual
(returns nothing) - Sent when LinuxCNC switches to manual mode.

mode-mdi
(returns nothing) - Sent when LinuxCNC switches to MDI mode.

mode-auto
(returns nothing) - Sent when LinuxCNC switches to auto mode.

command-running
(returns nothing) - Sent when running a program or MDI

command-stopped
(returns nothing) - Sent when a program or MDI stopped

command-error
(devuelve nada) - Enviado cuando hay un error de comando

interp-run
(returns nothing) - Sent when LinuxCNC's interpreter is running an MDI or program.

interp-idle
(returns nothing) - Sent when LinuxCNC's interpreter is idle.

interp-paused
(returns nothing) - Sent when LinuxCNC's interpreter is paused.

interp-reading
(returns nothing) - Sent when LinuxCNC's interpreter is reading.

interp-waiting
(returns nothing) - Sent when LinuxCNC's interpreter is waiting.

jograte-changed
(returns float) - Sent when jog rate has changed.
LinuxCNC does not have an internal jog rate.
This is GStat's internal jog rate.
It is expected to be in the machine's native units regardless of the current unit mode .

jograte-angular-changed
(returns float) - Sent when the angular jog rate has changed.
LinuxCNC does not have an internal angular jog rate.
This is GStat's internal jog rate.
It is expected to be in the machine's native units regardless of the current unit mode .

jogincrement-changed
(returns float, text) - Sent when jog increment has changed.
LinuxCNC does not have an internal jog increment.
This is GStat's internal jog increment.
It is expected to be in the machine's native units regardless of the current unit mode .

jogincrement-angular-changed
(returns float, text) - Sent when angular jog increment has changed.
LinuxCNC does not have an internal angular jog increment.
This is GStat's internal angular jog increment.
It is expected to be in the machine's native units regardless of the current unit mode .

program-pause-changed
(returns bool) - Sent when program is paused/unpaused.

optional-stop-changed
(returns bool) - Sent when optional stop is set/unset

block-delete-changed
(returns bool) - sent when block delete is set/unset.

file-loaded
(devuelve string) - Enviado cuando LinuxCNC ha cargado el archivo

reload-display
(returns nothing) - Sent when there is a request to reload the display

línea modificada
(returns integer) - Sent when LinuxCNC has read a new line.
LinuxCNC does not update this for every type of line.

tool-in-spindle-changed
(returns integer) - Sent when the tool has changed.

tool-info-changed
(returns Python object) - Sent when current tool info changes.

current-tool-offset
(returns Python object) - Sent when the current tool offsets change.

motion-mode-changed
(devuelve entero) - Enviado cuando modo de movimiento ha cambiado

spindle-control-changed
(returns integer, bool, integer, bool) - (spindle num, spindle on state, requested spindle direction & rate, at-speed state)
Sent when spindle direction or running status changes or at-speed changes.

current-feed-rate
(returns float) - Sent when the current feed rate changes.

current-x-rel-position
(devuelve float) - Enviado cada 100 ms.

current-position
(returns pyobject, pyobject, pyobject, pyobject) - Sent every 100 ms.
Returns tuples of position, relative position, distance-to-go and the joint actual position. Before homing, on multi-joint axes, only joint position is valid.

current-z-rotation
(devuelve float) - El ángulo de rotación XY actual alrededor del eje Z

requested-spindle-speed-changed
(devuelve float) - Enviado cuando las RPM requeridas actuales cambian

actual-spindle-speed-changed
(returns float) - Sent when the actual RPM changes based on the HAL pin `spindle.0.speed-in`.

spindle-override-changed
(devuelve float) - Enviado cuando cambia el valor de anulación del husillo
en porcentaje

feed-override-changed
(devuelve float) - Enviado cuando el valor de anulación de alimentación cambia
en porcentaje

rapid-override-changed
(devuelve float) - Enviado cuando el valor de anulación rapidez cambia
en porcentaje (0-100)

max-velocity-override-changed
(devuelve float) - Enviado cuando la velocidad máxima anula cambios de valor
en unidades por minuto

feed-hold-enabled-changed
(devuelve bool) - Enviado cuando cambia el estado de retención de alimentación

itime-mode
(devuelve bool) - Enviado en el cambio de estado G93
(modo de tiempo inverso)

fpm-mode
(devuelve bool) - Enviado en cambio de estado G94
(modo alimentación por minuto)

fpr-mode
(devuelve bool) - Enviado en cambio de estado G95
(modo alimentación por revolución)

css-mode
(devuelve bool) - Enviado con cambio de estado G96
(modo de alimentación de superficie constante)

rpm-mode
(devuelve bool) - Enviado con cambio de estado G97
(modo RPM constantes)

radius-mode
(devuelve bool) - Enviando con cambios de estado G8
muestra X en modo radio

diameter-mode
(devuelve bool) - Enviado con cambios de estado G7
muestra X en modo diámetro

flood-changed
(returns bool) - Sent when flood coolant state changes.

mist-changed
(returns bool) - Sent when mist coolant state changes.

m-code-changed
(devuelve cadena) - Enviado cuando cambia el código M activo

g-code-changed
(devuelve cadena) - Enviado cuando cambia el código G activo

metric-mode-changed
(devuelve boolean) - Enviado cuando el estado G21 cambia

user-system-changed
(devuelve cadena) - Enviado cuando cambia el sistema de coordenadas de referencia (G5x)

mdi-line-selected
(returns string, string) - intended to be sent when an MDI line is selected by user.
This depends on the widget/libraries used.

gcode-line-selected
(returns integer) - intended to be sent when a G-code line is selected by user.
This depends on the widget/libraries used.

graphics-line-selected
(returns integer) - intended to be sent when graphics line is selected by user.
This depends on the widget/libraries used.

graphics-loading-progress
(returns integer) - intended to return percentage done of loading a program or running a program.
This depends on the widget/libraries used.

graphics-gcode-error
(returns string) - intended to be sent when a G-code error is found when loading.
This depends on the widget/libraries used.

graphics-gcode-properties
(returns Python dict) - Enviado cuando el código G se ha cargado.
El diccionario contiene las claves siguientes:

- `name` (*string*): Name of the loaded file
- `size` (*string*): Size in bytes and lines
- `g0` (*string*): Total rapid distance

- `g1` (*string*): Total feed distance
- `run` (*string*): Estimated program run time
- `toollist` (*list*): List of used tools
- `x` (*string*): X extents (bounds) [1]
- `x_zero_rxy` (*string*): X extents without rotation around z (bounds) [1]
- `y` (*string*): Y extents (bounds) [1]
- `y_zero_rxy` (*string*): Y extents without rotation around z (bounds) [1]
- `z` (*string*): Z extents (bounds) [1]
- `z_zero_rxy` (*string*): Z extents without rotation around z (bounds) [1]
- `machine_unit_sys` (*string*): Machine units (*Metric* or *Imperial*)
- `gcode_units` (*string*): Units in G-code file (*mm* or *in*)

---

nota

1. See the images extends non-rotated and extends rotated 30 degrees for a better understanding.

---

graphics-view-changed
(returns string, Python dict or None) - intended to be sent when graphics view is changed.
This depends on the widget/libraries used.

mdi-history-changed
(returns None) - intended to be sent when an MDI history needs to be reloaded.
This depends on the widget/libraries used.

machine-log-changed
(returns None) - intended to be sent when machine log has changed.
This depends on the widget/libraries used.

update-machine-log
(returns string, string) - intended to be sent when updating the machine.
This depends on the widget/libraries used.

move-text-lineup
(returns None) - intended to be sent when moving the cursor one line up in G-code display.
This depends on the widget/libraries used.

move-text-linedown
(returns None) - intended to be sent when moving the cursor one line down in G-code display.
This depends on the widget/libraries used.

dialog-request
(returns Python dict) - intended to be sent when requesting a GUI dialog.
It uses a Python dict for communication. The dict must include the following keyname pair:

- NAME: *requested dialog name*
  The dict usually has several keyname pairs - it depends on the dialog.
  dialogs return information using a general message
  This depends on the widget/libraries used.

focus-overlay-changed
(returns bool, string, Python object) - intended to be sent when requesting an overlay to be put over the display.
This depends on the widget/libraries used.

play-sound
(returns string) - intended to be sent when requesting a specific sound file to be played.
This depends on the widget/libraries used.

virtual-keyboard
(returns string) - intended to be sent when requesting a on screen keyboard.
This depends on the widget/libraries used.

dro-reference-change-request
(returns integer) - intended to be sent when requesting a DRO widget to change its reference.
0 = machine, 1 = relative, 3 = distance-to-go
This depends on the widget/libraries used.

show-preferences
(returns None) - intended to be sent when requesting the screen preferences to be displayed.
This depends on the widget/libraries used.

shutdown
(returns None) - intended to be sent when requesting LinuxCNC to shutdown.
This depends on the widget/libraries used.

status-message
returns python dict (message), python dict (options) Intended for a screen/panel to get status/log messages from widgets, but can be used generally.
The listening object is expected to look for and handle at least these entries:

The message dict would include:

- TITLE: (string)
- SHORTTEXT: (string)
- DETAILS: (string)

The options dict would include:

- LEVEL: (integer)
- LOG: (bool)

The listening object could use this to display information on a text line or message dialog.
The LEVEL would indicate urgency 0 = DEFAULT 1 = WARNING 2 = CRITICAL
LOG indicates whether the message should be logged to a file/page if available.
LOG messages would be assumed to use the DETAILS entry.

An example of how to send a message:

```
mess = {'SHORTTEXT':'File Copy Failed',
        'TITLE':'FileManager',
        'DETAILS':'There is not enough room on disk to copy the file'}
opt = {'LOG':True,'LEVEL':2}
STATUS.emit('status-message',mess,opt)
```

Un ejemplo del objeto oyente:

```
# tell STATUS we want to respond to any sent 'status-message' messages.
STATUS.connect('status-message', lambda w, d, o: self.add_external_status(m,o))

    def add_external_status(self, message, option):

        # extract and trap errors for expected entries
        level = option.get('LEVEL', STATUS.DEFAULT)
        log = option.get("LOG", True)
        title = message.get('TITLE', '')
        mess = message.get('SHORTTEXT', '')
        logtext = message.get('DETAILS', '')

        # call a function to print the message on a statusbar:
        self.add_status(mess, level, noLog=True)

        # request a log file update
        if log:
            STATUS.emit('update-machine-log', "{}\n{}".format(title, logtext), 'TIME')
```

error
(returns integer, string) - intended to be sent when an error has been reported .
integer represents the kind of error. ERROR, TEXT or DISPLAY
string is the actual error message.
This depends on the widget/libraries used.

general
(returns Python dict) - intended to be sent when message must be sent that is not covered by a more specific message.
General message should be used a sparsely as reasonable because all object connected to it will have to parse it.
It uses a Python dict for communication.
The dict should include and be checked for a unique id keyname pair:

- ID: *UNIQUE_ID_CODE*
  The dict usually has more keyname pair - it depends on implementation.

forced-update
(returns None) - intended to be sent when one wishes to initialize or arbitrarily update an object.
This depends on the widget/libraries used.

progress
(returns integer, Python object) - intended to be sent to indicate the progress of a filter program.
This depends on the widget/libraries used.

following-error
(devuelve lista Pyhton) - devuelve una lista de todas las articulaciones en error de seguimiento actual.

13.4.4. Funciones

These are convenience functions that are commonly used in programming.

set_jograte
(float) - LinuxCNC has no internal concept of jog rate -each GUI has its own. This is not always convenient.

This function allows one to set a jog rate for all objects connected to the signal `jograte-changed`.
It defaults to 15.
GSTAT.set_jog_rate(10) would set the jog rate to 10 machine-units-per-minute and emit the `jograte-cha`
signal.

get_jograte()
(Nothing) - x = GSTAT.get_jograte() would return GSTAT's current internal jograte (float).

set_jograte_angular
(float) -

get_jograte_angular
(None) -

set_jog_increment_angular
(float, string) -

get_jog_increment_angular
(None) -

set_jog_increments
(float, string) -

get_jog_increments
(None) -

is_all_homed
(nothing) - This will return the current state of all_homed (BOOL).

machine_is_on
(nothing) - This will return the current state of machine (BOOL).

estop_is_clear
(nothing) - This will return the state of Estop (BOOL)

set_tool_touchoff
(tool,axis,value) - This command will

1. record the current mode,
2. switch to MDI mode,
3. invoke the MDI command: G10 L10 P[TOOL] [AXIS] [VALUE],
4. espera a que termine,
5. invoke G43,
6. espera a que termine,
7. switch back to the original mode.

set_axis_origin
(axis,value) - This command will

1. record the current mode,
2. switch to MDI mode,
3. invoke the MDI command: G10 L20 P0 [AXIS] [VALUE],
4. espera a que termine,
5. regresa al modo original,
6. emit a *reload-display* signal.

do_jog
(axis_number,direction, distance) - This will jog an axis continuously or at a set distance.
You must be in the proper mode to jog.

check_for_modes
(mode) - This function checks for required LinuxCNC mode.
It returns a Python tuple (state, mode)
mode will be set the mode the system is in
state will set to:

- false if mode is 0
- false if machine is busy
- true if LinuxCNC is in the requested mode
- None if possible to change, but not in requested mode

get_current_mode
(nada) - devuelve integer: el modo actual de LinuxCNC.

set_selected_joint
(integer) - records the selected joint number internally.
requests the joint to be selected by emitting the
joint-selection-changed message.

get_selected_joint
(None) - returns integer representing the internal selected joint number.

set_selected_axis
(string) - records the selected axis letter internally.
Requests the axis to be selected by emitting the `axis-selection-changed` message.

get_selected_axis
(None) - returns string representing the internal selected axis letter.

is_man_mode
(None) -

is_mdi_mode
(None) -

is_auto_mode
(None) -

is_on_and_idle
(None) -

is_auto_running
(None) -

is_auto_paused
(None) -

is_file_loaded
(None) -

is_metric_mode
(None) -

is_spindle_on
(None) -

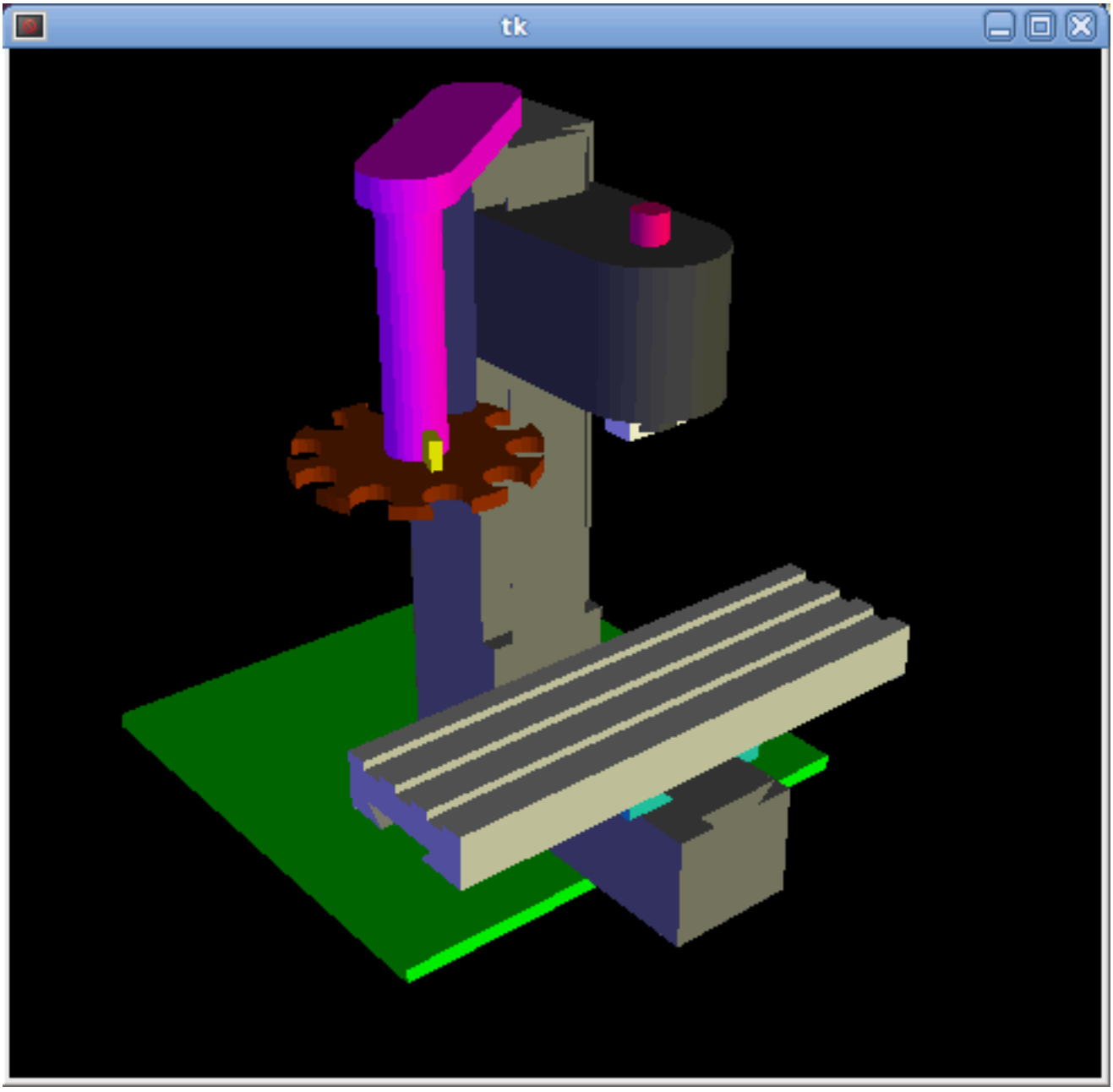
shutdown
(None) -

13.4.5. Known Issues

Some status points are reported wrongly during a running program because the interpreter runs ahead of the current position of a running program. This will hopefully be resolved with the merge of state-tags branch.

13.5. Vismach

Vismach is a set of Python functions that can be used to create and animate models of machines. Vismach displays the model in a 3D viewport and the model parts are animated as the values of associated HAL pins change.



The Vismach viewport view can be manipulated as follows:

- **zoom** by scroll wheel or right button drag,
- **pan** by left button drag,
- **rotate** by middle-button drag or shift-drag.

A Vismach model takes the form of a Python script and can use standard Python syntax. This means that there is more than one way to lay out the script, but in the examples given in this document I will use the simplest and most basic of them.

The basic sequence in creating the Vismach model is

- Create the HAL pins that control the motion.
- Crear las partes.
- Define how they move.
- Ensambla en grupos de movimiento.

13.5.1. Iniciar el script

It is useful for testing to include the *#!/usr/bin/env python3* to allow the file to be run as a script. The first thing to do is to import the required libraries.

```
#!/usr/bin/env python3

from vismach import *
import hal
import math
import sys
```

13.5.2. Crea los pines HAL.

HAL pins are created with the normal Python ”hal” library, and are not specific to Vismach. Further details can be found in the Creating Non-realtime Components in Python section. A component should be created with a name that matches the script file name and then the HAL pins are added to that component. They will be referenced by their component handle and short name when used to animate the Vismach model.

```
c = hal.component(”samplegui”)
c.newpin(”joint0”, hal.HAL_FLOAT, hal.HAL_IN)
c.newpin(”joint1”, hal.HAL_FLOAT, hal.HAL_IN)
c.ready()
```

Will create HAL pins *samplegui.joint0* and *samplegui.joint1*. When loading the Vismach model with *loadusr -W samplegui* the *c.ready()* function tells loadusr it’s ready.

13.5.3. Creating Parts

It is probably easiest to create geometry in a CAD package and **import** into the model script with the `AsciiSTL()` or `AsciiOBJ()` functions. Both functions can take one of two named arguments, either a filename or raw data:

- `part = AsciiSTL(filename="path/to/file.stl") + part = AsciiSTL(data="solid part1 facet normal ....") + part = AsciiOBJ(filename="path/to/file.obj") + part = AsciiOBJ(data="v 0.123 0.234 0.345 1.0 ...")`
  The parts will be created in the Vismach space in the same locations as they occupy in the STL or OBJ space. This means that it may be possible to assemble the model in the CAD package.

Alternatively parts can be created inside the model script from a range of **shape primitives**. Many shapes are created at the origin and need to be moved to the required location after creation:

- `cylinder = CylinderX(x1, r1, x2, r2) + cylinder = CylinderY(y1, r1, y2, r2) + cylinder = CylinderZ(z1, r1, z2, r2)`
  Creates a (optionally tapered) cylinder on the given axis with the given radii at the given points on the axis.
- `sphere = Sphere(x, y, z, r)`
  Creates a sphere of radius r at (x,y,z)
- `triangle = TriangleXY(x1, y1, x2, y2, x3, y3, z1, z2) + triangle = TriangleXZ(x1, z1, x2, z2, x3, z3, y1, y2) + triangle = TriangleYZ(y1, z1, y2, z2, y3, z3, x1, x2)`
  Creates a triangular plate between planes defined by the last two values parallel to the specified plane, with vertices given by the three coordinate pairs.
- `arc = ArcX(x1, x2, r1, r2, a1, a2)`
  Create an arc shape.
- `box = Box(x1, y1, z1, x2, y2, z2)`
  Creates a rectangular prism with opposite corners at the specified positions and edges parallel to the XYZ axes.
- `box = BoxCentered(xw, yw, zw)`
  Creates an xw by yw by zw box centred on the origin.
- `box = BoxCenteredXY(xw, yw, z)`
  Creates a box of width xw / yw and height z.

Composite parts may be created by **assembling** these primitives either at creation time or subsequently using `Collection()`:

```
part1 = Collection([Sphere(100,100,100,50), CylinderX(100,40,150,30)])
part2 = Box(50,40,75,100,75,100)
part3 = Collection([part2, TriangleXY(10,10,20,10,15,20,100,101)])
part4 = Collection([part1, part2])
```

13.5.4. Moving Parts

Parts may need to be moved in the Vismach space to assemble the model. They may also need to be moved to create the animation as the animation rotation axis is created at the origin (but moves with the Part):

- `part1 = Translate([part1], x, y, z)`
  Move part1 the specified distances in x, y and z.
- `part1 = Rotate([part1], theta, x, y, z)`
  Rotate the part by angle theta about an axis between the origin and x, y, z.

13.5.5. Animating Parts

To animate the model (controlled by the values of HAL pins) there are two functions *HalTranslate* and *HalRotate*. For parts to move inside an assembly they need to have their HAL motions defined before being assembled with the ”Collection” command. The rotation axis and translation vector move with the part as it is moved by the vismach script during model assembly, or as it moves in response to the HAL pins as the model is animated:

- `part = HalTranslate([part], comp, ”hal_pin”, xs, ys, zs)`
  The function arguments are:
  - first a *collection/part* which can be pre-created earlier in the script, or could be created at this point if preferred eg `part1 = HalTranslate([Box(....)], ...)`.
  - The *HAL component* is the next argument, i.e. the object returned by the `comp = hal.component(...)` command. After that is the name of the HAL in that will animate the motion, this needs to match an existing HAL pin that is part of the HAL component created earlier in the script.
  - Then follow the *X, Y, Z scales*.
    For a Cartesian machine created at 1:1 scale this would typically be 1,0,0 for a motion in the positive X direction.
    However if the STL file happened to be in cm and the machine was in inches, this could be fixed at this point by using 0.3937 (1cm /2.54in) as the scale.
- `part = HalRotate([part], comp, ”hal_pin”, angle_scale, x, y, z)`
  This command is similar in its operation to HalTranslate except that it is typically necessary to move the part to the origin first to define the axis.
  - The *axis of rotation* is from the origin point to the point defined by (x,y,z).
    When the part is moved back away from the origin to its correct location the axis of rotation can be considered to remain ”embedded” in the part.
  - *Rotation angles* are in degrees, so for a rotary joint with a 0-1 scaling you would need to use an angle scale of 360.

13.5.6. Assembling the model.

In order for parts to move together they need to be assembled with the Collection() command. It is important to assemble the parts and define their motions in the correct sequence. For example to create a moving head milling machine with a rotating spindle and an animated draw bar you would:

- Create the head main body.
- Create the spindle at the origin.
- Define the rotation.
- Move the head to the spindle or spindle to the head.
- Crear la barra de dibujo.
- Define the motion of the draw bar.
- Assemble the three parts into a head assembly.
- Define the motion of the head assembly.

In this example the spindle rotation is indicated by rotation of a set of drive dogs:

```
#Drive dogs
dogs = Box(-6,-3,94,6,3,100)
dogs = Color([1,1,1,1],[dogs])
dogs = HalRotate([dogs],c,"spindle",360,0,0,1)
dogs = Translate([dogs],-1,49,0)

#Drawbar
draw = CylinderZ(120,3,125,3)
draw = Color([1,0,.5,1],[draw])
draw = Translate([draw],-1,49,0)
draw = HalTranslate([draw],c,"drawbar",0,0,1)

# head/spindle
head = AsciiSTL(filename="./head.stl")
head = Color([0.3,0.3,0.3,1],[head])
head = Translate([head],0,0,4)
head = Collection([head, tool, dogs, draw])
head = HalTranslate([head],c,"Z",0,0,0.1)

# base
base = AsciiSTL(filename="./base.stl")
base = Color([0.5,0.5,0.5,1],[base])
# mount head on it
base = Collection([head, base])
```

Finally a single collection of all the machine parts, floor and work (if any) needs to be created:

- For a *serial machine* each new part will be added to the collection of the previous part.
- For a *parallel machine* there may be several "base" parts.

Thus, for example, in scaragui.py link3 is added to link2, link2 to link1 and link1 to link0, so the final model is created by:

```
model = Collection([link0, floor, table])
```

Whereas a VMC model with separate parts moving on the base might have:

```
model = Collection([base, saddle, head, carousel])
```

13.5.7. Other functions

- `part = Color([colorspec], [part])`
  Sets the display color of the part. Note that unlike the other functions the part definition comes second in this case.
  The colorspec consists of the three RGB values and an opacity. For example [1,0,0,0.5] for a 50 % opacity red.
- `myhud = Hud()`
  Creates a heads-up display in the Vismach GUI to display such items as axis positions.
- `tooltip = Capture()`
  Think of this as an invisible part that needs to be attached to the tooltip to track the position and orientation of the tool coordinate system. It is actually a transformation matrix that is constantly updated as the model moves.
- `work = Capture()`
  Same as above but attached to the work table to track the work coordinate system.

- `main(model, tooltip, work, size=10, hud=0, rotation_vectors=None, lat=0, lon=0)`
  This is the command that makes it all happen, creates the display etc.
  - *model* should be a collection that contains all the machine parts.
  - *tooltip* and *work* need to be created by `Capture()`. Vismach needs this information to draw the backplot which is basically the tooltip position drawn in the work coordinate system.
    See `scaragui.py` for an example of how to connect the tool tip to a tool and the tool to the model.
  - Either *rotation_vectors* or *latitude/longitude* can be used to set the original viewpoint and it is advisable to do as the default initial viewpoint is rather unhelpfully from immediately overhead.
  - *size* sets the extent of the volume visualized in the initial view.
  - *hud* refers to a head-up display of axis positions.

13.5.8. Basic structure of a Vismach script.

```
#imports
from vismach import *
import hal
#create the HAL component and pins
comp = hal.component("compname")
comp.newpin("pin_name", hal.HAL_FLOAT, hal.HAL_IN)
...
#create the floor, tool and work
floor = Box(-50, -50, -3, 50, 50, 0)
work = Capture()
tooltip = Capture()
...
#Build and assemble the model
part1 = Collection([Box(-6,-3,94,6,3,100)])
part1 = Color([1,1,1,1],[part1])
part1 = HalRotate([part1],comp,"pin_name",360,0,0,1)
part1 = Translate([dogs],-1,49,0)
...
#create a top-level model
model = Collection([base, saddle, head, carousel])
#Start the visualization
main(model, tooltip, work, 100, lat=-75, lon=215)
```

Parte III

Glosario, Derechos de Copia e Historia

Capítulo 14

Overleaf

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.

Derechos de autor © 2000-2025 LinuxCNC.org

Se otorga permiso para copiar, distribuir y/o modificar este documento bajo los términos de la Licencia de Documentación Libre GNU, versión 1.1 o cualquier versión posterior publicada por la Fundación de Software Libre; sin secciones invariantes, sin textos en la portada frontal y sin textos en la contra-portada. Se incluye una copia de la licencia en la sección titulada ”Licencia de Documentación Libre GNU”.

Si no encuentra la licencia, puede solicitar una copia a:

```
Free Software Foundation, Inc.
51 Franklin Street
Fifth Floor
Boston, MA 02110-1301 USA.
```

(La versión en idioma inglés es autoritaria)

LINUX® es la marca registrada de Linus Torvalds en los EE. UU. y otros países. La marca registrada Linux® se utiliza de conformidad con una sublicencia de LMI, el licenciatario exclusivo de Linus Torvalds, propietario de la marca en un base mundial.

El proyecto LinuxCNC no está afiliado a Debian®. *Debian* es una marca registrada propiedad de Software in the Public Interest, Inc.

El proyecto LinuxCNC no está afiliado a UBUNTU®. *UBUNTU* es una marca registrada propiedad de Canonical Limited.

Capítulo 15

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 16

Términos de Derechos de Copia

16.1. Sección legal

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

16.1.1. Términos de derechos de autor

Copyright (c) 2000-2022 LinuxCNC.org

Se otorga permiso para copiar, distribuir y/o modificar este documento bajo los términos de la Licencia de Documentación Libre GNU, versión 1.1 o cualquier versión posterior publicada por la Fundación de Software Libre; sin secciones invariantes, sin textos en la portada frontal y sin textos en la contraportada. Se incluye una copia de la licencia en la sección titulada ”Licencia de Documentación Libre GNU”.

16.1.2. Licencia GNU de Documentation Libre

Licencia de Documentación Libre GNU Versión 1.1, Marzo 2000

Derechos de autor © 2000 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA. Todos están permitidos para copiar y distribuir copias físicas de este documento licencia, pero no esta permitido modificarlo.

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.

Capítulo 17

Historia de LinuxCNC

17.1. Origen

EMC (Controlador de Máquina Mejorado) fue creado por el NIST, el Instituto Nacional de Normas y Tecnología, que es una agencia del Departamento de Comercio del Gobierno de los Estados Unidos.

El NIST primero se interesó en escribir un paquete de control de movimiento como una plataforma de prueba de conceptos y estándares. El patrocinio temprano de General Motors resultó en una adaptación de la versión incipiente de EMC usando tarjetas de control inteligentes PMAC, ejecutándose en una versión ”en tiempo real” de Windows NT y controlando una gran fresadora.

Como le es obligatorio a todos los *productos de trabajo* de los empleados del gobierno federal de EE.UU., el software resultante y el informe al respecto deben estar en el dominio público y, por ello, fue debidamente publicado un informe al respecto, incluyendo en Internet. Fue allí donde Matt Shaver descubrió EMC. Contactó con el NIST y entró en conversaciones con Fred Proctor sobre la adaptación del código para utilizarlo para controlar hardware menos costoso que se utilizaría para actualización y remplazo de controles CNC que estaban obsoletos o simplemente muertos. NIST quedó intrigado porque ellos también querían algo menos costoso. Para lanzar este esfuerzo cooperativo, se creó un acuerdo formal que garantizó que el código resultante y el diseño permanecerían en el dominio público.

Las primeras consideraciones se centraron en reemplazar el costoso y temperamental Sistema de Windows NT ”en tiempo real”. Se propuso que se intentaría con una relativamente nueva (en aquel momento) extensión en tiempo real del sistema operativo Linux. Esta idea fue llevada a cabo con éxito. Lo siguiente fue el tema de las costosas tarjetas inteligentes de control de movimiento. En esas fechas, el poder de procesamiento de una PC era considerado lo suficientemente grande como para tomar el control directo de las rutinas de movimiento. Una búsqueda rápida de hardware disponible resultó en la selección de las tarjetas de interfaz ”Servo-To-Go” como la primer plataforma para permitir que una PC controle directamente los motores. Se agregó a la interfaz de usuario existente software para planificación de trayectoria, control de bucle PID y un intérprete RS274. Matt utilizó con éxito esta versión para actualizar un par de máquinas con controles muertos y esta se convirtió en el sistema EMC que primero llamó la atención del mundo exterior. La mención de EMC en el grupo de noticias USENET rec.crafts.metalworking resultó en los primeros adoptantes como Jon Elson construyendo sistemas para aprovechar EMC.

NIST creó una lista de correo para las personas interesadas en EMC. Con el paso del tiempo, otros, ademas del NIST, se interesaron en mejorar EMC. Mucha gente solicitó pequeñas mejoras en el código. Ray Henry quería refinar el interfaz de usuario. Como Ray era reacio a intentar alterar el código C en el que se escribió la interfaz, se buscó un método más simple. Fred Proctor, del NIST, sugirió un lenguaje de secuencias de comandos y escribió código para interactuar con el lenguaje de script Tcl/Tk para las comunicaciones internas NML de EMC. Con esta herramienta Ray pasó a escribir un programa Tcl/Tk que se convirtió en la interfaz de usuario predominante para EMC en el momento.

Desde la perspectiva del NIST, vea esto paper escrito por William Shackleford y Frederick Proctor, describiendo la historia de EMC y su transición a código abierto.

Para ese tiempo, el interés en EMC comienza a subir sustancialmente. Cuantas más y más personas intentaron instalar EMC, la dificultad de parchar un kernel de Linux con las extensiones en tiempo real y compilar el código EMC se volvió notoriamente obvia. Hubo muchos intentos de documentar el proceso y se intentaron escribir scripts, algunos con éxito moderado. El problema de hacer coincidir la versión correcta de los parches y compiladores con la versión seleccionada de Linux seguía apareciendo. Paul Corner vino al rescate con la BDI (Brain Dead Install) que era un CD con el cual podían instalarse un sistema de trabajo completo (Linux, parches y EMC). El enfoque BDI abrió el mundo de EMC a una comunidad de usuarios mucho más grande. A medida que esta comunidad continuó creciendo, la lista de correo de EMC y los archivos de código se movieron a SourceForge y se estableció el sitio web LinuxCNC.

Con una comunidad más grande de usuarios participando, EMC se convirtió en un foco principal de interés en las exposiciones CNC en curso en NAMES y NAMES se convirtió en el evento anual de reunión para EMC. Durante los primeros años, las reuniones solo sucedieron porque las partes interesadas estaban en NAMES. En 2003, la comunidad de usuarios de EMC tuvo su primera reunión pública anunciada. Se celebró el lunes después de NAMES en el vestíbulo donde se celebró la exhibición NAMES. La organización fue floja, pero nació la idea de una capa de abstracción de hardware (HAL) y fue propuesto el movimiento para reestructurar el código para facilitar el desarrollo (EMC2).

17.1.1. Cambio de nombre

En la primavera de 2011, la Junta Directiva de LinuxCNC fue contactada por una firma de abogados representando a EMC Corporation (www.emc.com) sobre el uso de ”EMC” y ”EMC2” para identificar el software ofrecido en linuxcnc.org. EMC Corporation ha registrado varias marcas comerciales relacionadas con EMC y EMC² (EMC con numeral superíndice dos).

Después de varias conversaciones con el representante de EMC Corporation, el resultado final fue que, comenzando con el próximo gran lanzamiento del software, linuxcnc.org dejará de identificarlo usando ”emc” o ”EMC”, o aquellos términos seguidos de dígitos. En la medida que la Junta Directiva de LinuxCNC controla los nombres utilizados para identificar el software ofrecido en linuxcnc.org, la junta aceptado esto.

Como resultado, fue necesario elegir un nuevo nombre para el software. De las opciones que la junta consideró, hubo consenso de que ”LinuxCNC” era la mejor opción, ya que ese ha sido el nombre de nuestro sitio web durante años.

En preparación para el nuevo nombre, hemos recibido una sublicencia de Marca registrada LINUX® de la Fundación Linux (www.linuxfoundation.org), protegiendo nuestro uso del nombre LinuxCNC. (LINUX® es la marca registrada de Linus Torvalds en los EE. UU. y otros países.)

El esfuerzo de cambio de marca incluyó al sitio web linuxcnc.org, canales IRC y versiones del software y la documentación, comenzando con 2.5.0.

17.1.2. Información adicional

NIST publicó un artículo que describe el lenguaje RS274NGC y el centro de mecanizado abstracto que controla, así como una implementación temprana de EMC. El documento también está disponible en https://linuxcnc.org/files/RS274NGCv3.pdf .

NIST también publicó un artículo sobre la historia de EMC y sus transición a open source. El documento también está disponible en https://linuxcnc.org/files/Use-of-Open-Source-Distribution-for-a-Machine-Tool-Controller.pdf