Manuel de l’intégrateur V2.7.9, 2017-06-02

Table des matières

1 Concepts importants pour l'intégrateur **1**

1.1 Système pas à pas . . . 1

1.1.1 Période de base . . . 1

1.1.2 Timing des pas . . . 2

1.2 Systèmes à servomoteurs . . . 2

1.2.1 Opération de base . . . 2

1.2.2 Terme proportionnel . . . 3

1.2.3 Terme intégral . . . 3

1.2.4 Terme dérivé . . . 4

1.2.5 Réglage de la boucle . . . 4

1.2.6 Réglage manuel . . . 4

1.3 RTAI . . . 4

1.3.1 ACPI . . . 4

I Configuration de LinuxCNC **5**

2 Test des capacités temps réel **6**

2.1 Test de latence . . . 6

2.2 Adresses des ports . . . 7

3 Configuration de LinuxCNC **9**

3.1 Script de lancement . . . 9

3.2 Fichiers utilisés pour la configuration . . . 10

3.3 Double passe (TWOPASS) . . . 11

3.4 Organisation du fichier ini . . . 12

3.4.1 Les commentaires . . . 12

3.4.2 Les sections . . . 12

3.4.3 Les variables . . . 13

3.4.4 Sections et variables utilisateur . . . 13

3.5 Détails des sections du fichier ini . . . 14

3.5.1 Section [EMC] . . . 14
3.5.2 Section [DISPLAY] . . . 14
3.5.3 Section [FILTER] . . . 15
3.5.4 Section [RS274NGC] . . . 16
3.5.5 Section [EMCMOT] . . . 17
3.5.6 Section [TASK] . . . 17
3.5.7 Section [HAL] . . . 17
3.5.8 Section [HALUI] . . . 18
3.5.9 Section [TRAJ] . . . 18
3.5.10 Sections [AXIS_n] . . . 19
3.5.11 Section [HOMING] . . . 20
3.5.12 Variables relatives aux servomoteurs . . . 21
3.5.13 Variables relatives aux moteurs pas à pas . . . 22
3.5.14 Section [EMCIO] . . . 22

4 Prise d'origine **24**
4.1 La prise d'origine . . . 24
4.2 Séquences de prise d'origine . . . 24
4.3 Configuration . . . 26
4.3.1 Vitesse de recherche (HOME_SEARCH_VEL) . . . 26
4.3.2 Vitesse de détection (HOME_LATCH_VEL) . . . 26
4.3.3 HOME_IGNORE_LIMITS . . . 27
4.3.4 HOME_USE_INDEX . . . 27
4.3.5 HOME_OFFSET . . . 27
4.3.6 Position de l'origine (HOME) . . . 27
4.3.7 HOME_IS_SHARED . . . 27
4.3.8 HOME_SEQUENCE . . . 27
4.3.9 VOLATILE_HOME . . . 27
4.3.10 LOCKING_INDEXER . . . 28

5 Tours **29**
5.1 Plan par défaut . . . 29
5.2 Réglages INI . . . 29

6 Fichiers TCL pour HAL **30**
6.1 Compatibilité . . . 30
6.2 Commandes haltcl . . . 30
6.3 Variables du fichier ini et haltcl . . . 31
6.4 Conversion des fichiers .hal en fichiers .tcl . . . 31
6.5 Notes à propos de haltcl . . . 31
6.6 Exemples pour haltcl . . . 32
6.7 Interactivité de haltcl . . . 32
6.8 Exemples pour haltcl fournis avec la distribution (sim) . . . 33

7 LinuxCNC et HAL **34**
7.1 motion (temps réel) . . . 34
7.1.1 Options . . . 35
7.1.2 Pins . . . 35
7.1.2.1 Utilisation des pins de HAL pour l'orientation broche avec M19 . . . 36
7.1.3 Paramètres . . . 36
7.1.4 Fonctions . . . 37
7.2 axis.N (temps réel) . . . 37
7.2.1 Pins . . . 37
7.2.2 Paramètres . . . 38
7.3 iocontrol (espace utilisateur) . . . 38
7.3.1 Pins . . . 38

8 Configuration d'un système pas/direction (dir/step) **39**
8.1 Introduction . . . 39
8.2 Fréquence de pas maximale . . . 39
8.3 Brochage . . . 39
8.4 Le fichier standard_pinout.hal . . . 40
8.5 Vue d'ensemble du fichier standard_pinout.hal . . . 41
8.6 Modifier le fichier standard_pinout.hal . . . 41
8.7 Modifier la polarité d'un signal . . . 42
8.8 Ajouter le contrôle de vitesse broche en PWM . . . 42
8.9 Ajouter un signal de validation **enable** . . . 42
8.10 Ajouter un bouton d'Arrêt d'Urgence externe . . . 42

9 PyVCP **44**
9.1 Introduction . . . 44
9.2 Construction d'un panneau pyVCP . . . 45
9.3 Sécurité avec pyVCP . . . 46
9.4 Utiliser pyVCP avec AXIS . . . 46
9.5 Panneaux PyVCP autonomes . . . 47
9.6 Documentation des widgets de pyVCP . . . 48
9.6.1 Syntaxe . . . 48
9.6.2 Notes générales . . . 48
9.6.3 Commentaires . . . 49
9.6.4 Editer un fichier XML . . . 49
9.6.5 Couleurs . . . 49
9.6.6 Pins de HAL . . . 49
9.6.7 Label . . . 49

9.6.8 Les leds . . . 50
9.6.9 La led ronde . . . 50
9.6.10 La led rectangulaire . . . 50
9.6.11 Le bouton (button) . . . 51
9.6.11.1 Bouton avec texte (Text Button) . . . 51
9.6.11.2 Case à cocher (checkbutton) . . . 51
9.6.11.3 Bouton radio (radiobutton) . . . 52
9.6.12 Affichage d'un nombre (number) . . . 52
9.6.12.1 Number . . . 52
9.6.12.2 Flottant . . . 53
9.6.12.3 Nombre s32 . . . 53
9.6.12.4 Nombre u32 . . . 53
9.6.13 Affichage d'images . . . 53
9.6.13.1 Image Bit . . . 54
9.6.13.2 Image u32 . . . 54
9.6.14 Barre de progression (bar) . . . 55
9.6.15 Galvanomètre (meter) . . . 56
9.6.16 Boîte d'incrément (spinbox) . . . 56
9.6.17 Curseur (scale) . . . 57
9.6.18 Bouton tournant (dial) . . . 58
9.6.19 Manivelle (jogwheel) . . . 59
9.7 Documentation des containers de pyVCP . . . 60
9.7.1 Bordures . . . 60
9.7.2 Hbox . . . 60
9.7.3 Vbox . . . 61
9.7.4 Labelframe . . . 62
9.7.5 Table . . . 62
9.7.6 Onglets (Tabs) . . . 63

10 Exemples d'utilisation de PyVCP **64**
10.1 Panneau PyVCP dans AXIS . . . 64
10.2 Panneaux flottants . . . 64
10.3 Boutons de Jog . . . 64
10.3.1 Créer les Widgets . . . 65
10.3.2 Effectuer les connections . . . 67
10.4 Testeur de port . . . 68
10.5 Compte tours pour GS2 . . . 71
10.5.1 Le panneau . . . 71
10.5.2 Les connections . . . 73

11 Création d'interfaces graphiques avec GladeVCP **74**
11.1 Qu'est-ce que GladeVCP? . . . 74
11.1.1 PyVCP par rapport à GladeVCP . . . 74
11.2 Description du fonctionnement, avec un exemple de panneau . . . 75
11.2.1 Description de l'exemple de panneau . . . 78
11.2.2 Description de l'éditeur de Glade . . . 78
11.2.3 Explorer la fonction de rappel de Python . . . 79
11.3 Créer et intégrer une interface utilisateur Glade . . . 79
11.3.1 Pré-requis: Installation de Glade . . . 79
11.3.2 Lancer Glade pour créer une nouvelle interface utilisateur . . . 79
11.3.3 Tester un panneau . . . 80
11.3.4 Préparer le fichier de commande HAL . . . 80
11.3.5 Intégration dans Axis, comme pour PyVCP . . . 81
11.3.6 Intégration dans un nouvel onglet d'Axis, à la suite des autres . . . 81
11.3.7 Intégration dans Touchy . . . 82
11.4 Options de GladeVCP en ligne de commande . . . 82
11.5 Références des Widgets HAL . . . 83
11.5.1 Nommage des Widgets HAL et de leurs pins . . . 83
11.5.2 Donner des valeurs aux Widgets HAL et à leurs pins . . . 83
11.5.3 Le signal *hal-pin-changed* . . . 84
11.5.4 Les boutons (HAL Button) . . . 84
11.5.5 Les échelles (Scales) . . . 85
11.5.6 La boîte d'incrément (SpinButton) . . . 85
11.5.7 Les labels . . . 85
11.5.8 Les conteneurs: HAL_HBox et HAL_Table . . . 86
11.5.9 Les Leds . . . 86
11.5.10 La barre de progression (ProgressBar) . . . 86
11.5.11 La boîte combinée (ComboBox) . . . 87
11.5.12 Les barres . . . 87
11.5.13 L'indicateur (HAL Meter) . . . 88
11.5.14 Gremlin, visualiseur de parcours d'outil pour fichiers .ngc . . . 89
11.5.15 Fonction de diagrammes animés: Widgets HAL dans un bitmap . . . 90
11.6 Références des Widgets LinuxCNC Action . . . 91
11.6.1 Les widgets LinuxCNC Action . . . 92
11.6.2 Les widgets LinuxCNC bascule action (ToggleAction) . . . 92
11.6.3 La bascule Action_MDI et les widgets Action_MDI . . . 92
11.6.4 Un exemple simple: Exécuter une commande MDI lors de l'appui sur un bouton. . . . 92
11.6.5 Paramètres passés avec les widgets Action_MDI et ToggleAction_MDI . . . 93
11.6.6 Un exemple plus avancé: Passer des paramètres à un sous-programme O-word . . . 93

11.6.7 Préparation d'une Action_MDI . . . 94
11.6.8 Utiliser l'objet LinuxCNC Stat pour traiter les changements de statut . . . 94
11.7 Programmation de GladeVCP . . . 95
11.7.1 Actions définies par l'utilisateur . . . 95
11.7.2 Un exemple: ajouter une fonction de rappel en Python . . . 95
11.7.3 L'événement valeur de HAL modifiée . . . 96
11.7.4 Modèle de programmation . . . 96
11.7.4.1 Modèle du gestionnaire simple . . . 96
11.7.4.2 Modèle de gestionnaire basé sur les classes . . . 97
11.7.4.3 Le protocole get_handlers . . . 97
11.7.5 Séquence d'initialisation . . . 97
11.7.6 Multiple fonctions de rappel avec le même nom . . . 98
11.7.7 Le drapeau GladeVCP **-U <useropts>** . . . 98
11.7.8 Variables persistantes dans GladeVCP . . . 98
11.7.8.1 Examen de la persistance, de la version et de la signature du programme . . . 98
11.7.9 Utilisation des variables persistantes . . . 98
11.7.10 Édition manuelle des fichiers .ini . . . 99
11.7.11 Ajouter des pins de HAL . . . 100
11.7.12 Ajout de timers . . . 100
11.7.13 Exemples, et lancez votre propre application GladeVCP . . . 100
11.8 Questions & réponses . . . 100
11.9 Troubleshooting . . . 101
11.10 Notes d'implémentation: la gestion des touches dans Axis . . . 101

12 Notions avancées **102**

13 Python Interface **103**
13.1 The linuxcnc Python module . . . 103
13.2 Usage Patterns for the LinuxCNC NML interface . . . 103
13.3 Reading LinuxCNC status . . . 103
13.3.1 linuxcnc.stat attributes . . . 104
13.3.2 The axis dictionary . . . 108
13.4 Preparing to send commands . . . 109
13.5 Sending commands through `linuxcnc.command` . . . 109
13.5.1 `linuxcnc.command` attributes . . . 110
13.5.2 `linuxcnc.command` methods: . . . 110
13.6 Reading the error channel . . . 112
13.7 Reading ini file values . . . 112
13.8 The `linuxcnc.positionlogger` type . . . 113
13.8.1 members . . . 113
13.8.2 methods . . . 113

14 La cinématique dans LinuxCNC **114**
14.1 Introduction . . . 114
14.1.1 Les articulations par rapport aux axes . . . 114
14.2 Cinématiques triviales . . . 114
14.3 Cinématiques non triviales . . . 115
14.3.1 Transformation avant . . . 116
14.3.2 Transformation inverse . . . 116
14.4 Détails d'implémentation . . . 116

15 Réglages des pas à pas **118**
15.1 Obtenir le meilleur pilotage logiciel possible . . . 118
15.1.1 Effectuer un test de latence . . . 118
15.1.2 Connaître ce dont vos cartes de pilotage ont besoin . . . 118
15.1.3 Choisir la valeur de BASE_PERIOD . . . 119
15.1.4 Utiliser steplen, stepspace, dirsetup, et/ou dirhold . . . 120
15.1.5 Pas de secret! . . . 120

16 Réglages d'une boucle PID **121**
16.1 Régulation à PID . . . 121
16.1.1 Les bases du contrôle en boucle . . . 121
16.1.2 Théorie . . . 122
16.1.2.1 Action Proportionnelle . . . 122
16.1.2.2 Action Intégrale . . . 122
16.1.2.3 Action Dérivée . . . 122
16.1.3 Réglage d'une boucle . . . 122
16.1.3.1 Méthode simple . . . 123
16.1.3.2 Méthode de Ziegler-Nichols . . . 123

II La logique Ladder **124**

17 La programmation en Ladder **125**
17.1 Introduction . . . 125
17.2 Exemple . . . 125

18 Classicladder Programming **127**
18.1 Ladder Concepts . . . 127
18.2 Languages . . . 127
18.3 Components . . . 127
18.3.1 Files . . . 127
18.3.2 Realtime Module . . . 128

18.3.3 Variables . . . 128
18.4 Loading the Classic Ladder user module . . . 129
18.5 Classic Ladder GUI . . . 129
18.5.1 Sections Manager . . . 129
18.5.2 Section Display . . . 130
18.5.3 The Variable Windows . . . 131
18.5.4 Symbol Window . . . 133
18.5.5 The Editor window . . . 134
18.5.6 Config Window . . . 135
18.6 Ladder objects . . . 137
18.6.1 CONTACTS . . . 137
18.6.2 IEC TIMERS . . . 137
18.6.3 TIMERS . . . 137
18.6.4 MONOSTABLES . . . 138
18.6.5 COUNTERS . . . 138
18.6.6 COMPARE . . . 138
18.6.7 VARIABLE ASSIGNMENT . . . 139
18.6.8 COILS . . . 141
18.6.8.1 JUMP COIL . . . 141
18.6.8.2 CALL COIL . . . 141
18.7 Classic Ladder Variables . . . 142
18.8 GRAFCET Programming . . . 142
18.9 Modbus . . . 143
18.9.1 MODBUS Settings . . . 146
18.9.2 MODBUS Info . . . 146
18.9.3 Communication Errors . . . 147
18.9.4 MODBUS Bugs . . . 147
18.10 Setting up Classic Ladder . . . 147
18.10.1 Add the Modules . . . 148
18.10.2 Adding Ladder Logic . . . 148
18.11 Ladder Examples . . . 153
18.11.1 Wrapping Counter . . . 153
18.11.2 Reject Extra Pulses . . . 154
18.11.3 External E-Stop . . . 155
18.11.4 Timer/Operate Example . . . 158
18.11.5 Tool Turret . . . 159
18.11.6 Sequential Example . . . 159

III Exemples d'utilisation **161**

19 Deuxième port parallèle sur port PCI **162**

20 Contrôle de la broche **163**

20.1 Vitesse broche en 0-10V . . . 163
20.2 Vitesse de broche en PWM . . . 163
20.3 Marche broche . . . 163
20.4 Sens de rotation de la broche . . . 164
20.5 Démarrage en rampe . . . 164
20.6 Vitesse de broche avec signal de retour . . . 165
20.7 Vitesse broche atteinte . . . 165

21 Utilisation d'une manivelle **167**

22 Broche avec variateur GS2 **169**

22.1 Exemple . . . 169

IV Diagnostics **170**

23 Moteurs pas à pas **171**

23.1 Problèmes communs . . . 171
23.1.1 Le moteur n'avance que d'un pas . . . 171
23.1.2 Le moteur ne bouge pas . . . 171
23.1.3 Distance incorrecte . . . 171
23.2 Messages d'erreur . . . 171
23.2.1 Erreur de suivi . . . 171
23.2.2 Erreur de RTAPI . . . 172
23.3 Tester . . . 172
23.3.1 Tester le timing des pas . . . 172

24 Glossary **174**

25 Legal Section **179**

25.1 Copyright Terms . . . 179
25.2 GNU Free Documentation License . . . 179

26 Index **183**

The LinuxCNC Team

Ce manuel est en évolution permanente. Si vous voulez nous aider à son écriture, sa rédaction, sa traduction ou la préparation des graphiques, merci de contactez n'importe quel membre de l'équipe de traduction ou envoyez un courrier électronique à emc-users@lists.sourceforge.net.

Copyright © 2000–2012 LinuxCNC.org

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".. If you do not find the license you may order a copy from Free Software Foundation, Inc. 59 Temple Place, Suite 330 Boston, MA 02111-1307

LINUX® is the registered trademark of Linus Torvalds in the U.S. and other countries. The registered trademark Linux® is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

Permission est donnée de copier, distribuer et/ou modifier ce document selon les termes de la « GNU Free Documentation License », Version 1.3 ou toute version ultérieure publiée par la « Free Software Foundation »; sans sections inaltérables, sans texte de couverture ni quatrième de couverture. Une copie de la licence est incluse dans la section intitulée « GNU Free Documentation License ». Si vous ne trouvez pas la licence vous pouvez en commander un exemplaire chez Free Software Foundation, Inc. 59 Temple Place, Suite 330 Boston, MA 02111-1307

(La version de langue anglaise fait foi)

AVIS
La version Française de la documentation de LinuxCNC est toujours en retard sur l'originale faute de disponibilité des traducteurs.
Il est recommandé d'utiliser la documentation en Anglais chaque fois que possible.
Si vous souhaitez être un traducteur bénévole pour la documentation française de LinuxCNC, merci de nous contactez.

NOTICE
The French version of the LinuxCNC documentation is always behind the original fault availability of translators.
It's recommended to use the English documentation whenever possible.
If you would like to be a volunteer editor for the French translation of LinuxCNC, please contact us.

Chapitre 1

Concepts importants pour l'intégrateur

1.1 Système pas à pas

1.1.1 Période de base

Période de base (BASE_PERIOD) est le *métronome* de l'ordinateur de LinuxCNC. [1] A chaque période le logiciel de génération de pas calcule si c'est le moment pour une autre impulsion de pas. Une période de base plus courte permet de produire plus d'impulsions de pas par seconde, mais si elle est trop courte l'ordinateur passera tout son temps à générer les impulsions et les autres services seront ralentis voir bloqués. La latence et les besoins des pilotes des moteurs pas à pas déterminent la durée minimum de la période que nous pouvons utiliser.

La latence la plus défavorable peut ne se présenter que quelquefois par minute ou même moins. La probabilité que cela se produise au même moment qu'un changement de direction du moteur et faible. Donc on peut obtenir des erreurs très rares et intermittentes qui ruinent une production de temps en temps et il est impossible d'intervenir pour régler le problème.

La façon la plus simple d'éviter ce problème est de choisir un BASE_PERIOD qui est la somme des plus longues exigences de temps de votre périphérique et le pire cas de latence de votre ordinateur. Ceci n'est pas toujours le meilleur choix.

Par exemple si le pilote moteur a besoin d'un maintien du signal de direction d'une durée de 20 μs est que la latence maximum est de 11 μs, la période de base sera de 11+20=31 μs ce qui donne une fréquence de génération de pas de 32258 pas par seconde dans un mode et de 16129 pas par seconde dans un autre mode.

Le problème est qu'avec les exigences de 20us pour le maintien du signal et les 11 μs de latence cela nous force à utiliser une période défavorable de 31 μs. Mais le générateur de pas du logiciel LinuxCNC a quelques paramètres qui nous laissent régler les divers temps d'une période de différentes manières.

Par exemple, si *steplen* [2] est changé de 1 à 2, alors il y aura deux périodes entre le commencement et la fin de l'impulsion de pas. De même, si *dirhold* [3] est changé de 1 à 3, il y aura au moins trois périodes entre l'impulsion de pas et un changement d'état de la commande de direction.

Si nous pouvons utilisez *dirhold* pour répondre aux exigences des 20μs de maintiens du signal de direction, le prochain délai à respecter est la durée de 4.5 μs du signal de pas haut, additionnons les 11 μs de latence au 4.5 μs de signal haut et nous obtenons une période minimum de 15.5 μs. Lorsque nous essayons la valeur de 15.5, nous trouvons que l'ordinateur est lent, donc nous réglons sur 16 μs. Si nous laissons *dirhold* à 1 (par défaut) alors temps minimum entre un pas et le changement de direction est de 16 μs moins les 11 μs de latence ce qui nous donne 5μs. Ce qui n'est pas suffisant il nous manque 15 μs. Puisque la période est 16 μs, nous avons besoin d'encore une période. Dans ce cas nous changeons *dirhold* de 1 à 2. Maintenant le temps minimal entre la fin de l'impulsion de pas et le changement du signal de direction et de 5+16=21 μs. Nous n'avons plus à nous inquiéter d'une erreur de direction à cause de la latence.

Pour plus d'informations sur stepgen voir la section stepgen du manuel de HAL.

1. Cette section fait référence à l'utilisation de *stepgen* le générateur de pas intégré à LinuxCNC. Certains dispositifs matériels ont leur propre générateur de pas et n'utilisent pas celui incorporé à LinuxCNC. Dans ce cas se référer au manuel du matériel concerné.
2. Steplen se réfère à un paramètre qui ajuste la performance du générateur de pas incorporé à LinuxCNC, *stepgen*, qui est un composant de HAL. Ce paramètre ajuste la longueur de l'impulsion de pas. Continuez à lire, on expliquera tous finalement.
3. dirhold se réfère à un paramètre qui adapte la longueur du maintien du signal de commande de direction.

1.1.2 Timing des pas

Sur certain pilote moteur le rapport entre la durée des espaces et la durée des impulsions n'est pas égal, dans ce cas le point (le moment) du pas est important. Si le pas se déclenche sur le front descendant alors, la broche de sortie doit être inversée.

1.2 Systèmes à servomoteurs

1.2.1 Opération de base

Les systèmes à servomoteurs sont capables de vitesses plus élevées pour une précision équivalente au moteur pas-à-pas, mais ils sont plus coûteux et complexes. Contrairement aux systèmes pas à pas, les servo-systèmes nécessitent un dispositif de rétro-action pour se positionner. Ils ne fonctionnent pas immédiatement sorti de la boite et ils doivent être calibrés pour fonctionner contrairement aux moteurs pas à pas.

Cette différence s'explique par le mode de régulation différent des deux systèmes. Les servosystèmes sont régulés en *boucle fermée* et le moteur pas à pas en *boucle ouverte*.

Que signifie *boucle fermée* ? Regardons un schéma simplifié et la façon dont un système de servomoteur fonctionne.

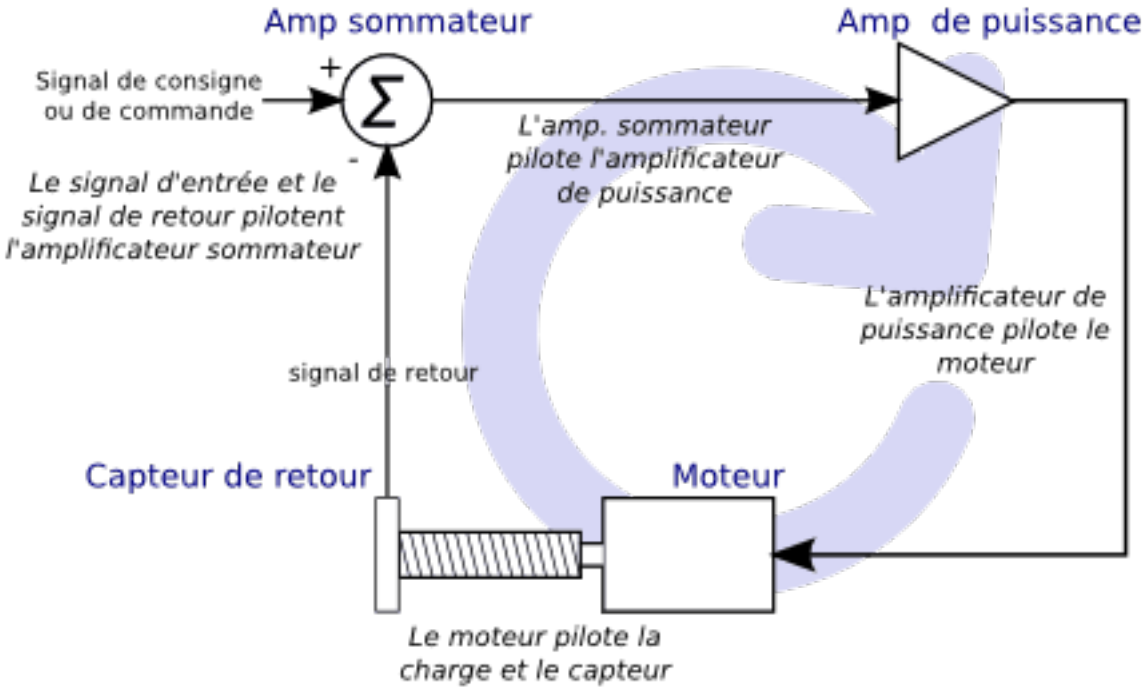


FIGURE 1.1 – Boucle fermée

Ce diagramme montre que le signal de consigne (de commande) et le signal de retour pilotent l'amplificateur sommateur, ensuite celui-ci pilote l'amplificateur de puissance, qui pilote le moteur, qui actionne la charge et le dispositif de retour d'information qui fournis le signal de retour. Cela se perçoit comme une boucle fermée où A contrôle B, B contrôle C, C contrôle D et D contrôle A.

Si vous n'avez pas travaillé avec des systèmes à servomoteurs auparavant, cela sera sans aucun doute étrange au premier abord, surtout par rapport aux plus normal des circuits électroniques, où le bon déroulement des entrées vers les sorties est de ne jamais revenir en arrière. [4]

4. Si cela peut aider, l'équivalent le plus proche dans le monde numérique ce sont les machines d'état, machines séquentielles où l'état des sorties à ce moment dépend de l'état que les entrées et sorties avaient avant. Si cela n'aide pas, alors passons.

Si tout contrôle tout le reste comment cela peut-il fonctionner, qui en a la charge ? La réponse est que LinuxCNC peut contrôler ce système, mais il doit le faire en choisissant une des différentes méthodes de contrôle.

La méthode de contrôle qu'utilise LinuxCNC, est l'une des plus simples et la meilleure appelée PID. PID est l'acronyme de **P**roportionnelle, **I**ntégrale et **D**érivée. La valeur proportionnelle détermine la réaction à l'erreur actuelle, la valeur intégrale détermine la réaction basée sur la somme d'erreurs récentes et la valeur dérivée détermine la réaction basée sur la vitesse de variation de l'erreur. Ce sont trois techniques communes de mathématique qui sont appliquées pour fournir un processus de suivi d'une consigne. Dans le cas de LinuxCNC le processus que nous voulons contrôler est l'actuelle position de l'axe et le point de consigne qui est la position commandée l'axe.

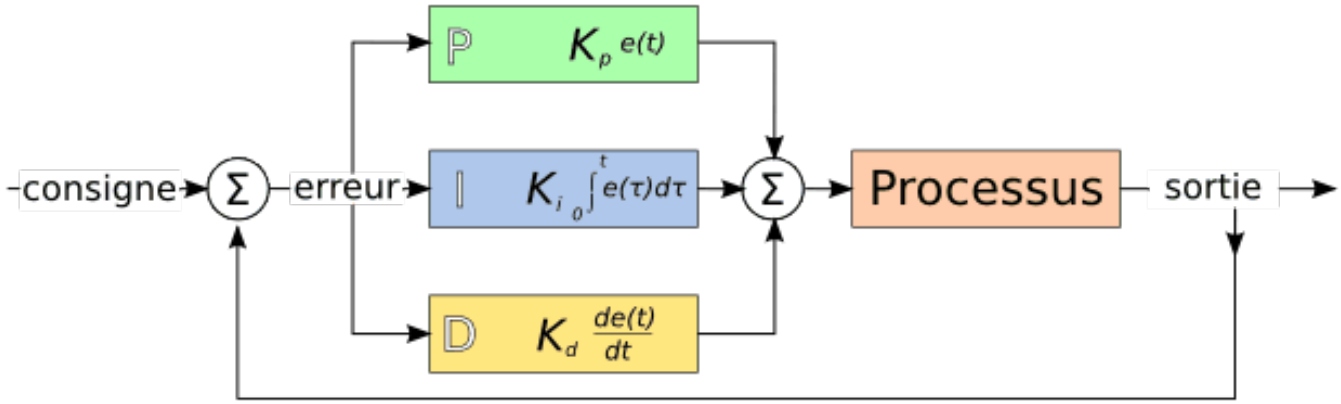


FIGURE 1.2 – Boucle PID

En ajustant trois composantes (proportionnelle, intégrale et dérivée) dans l'algorithme du contrôleur PID, nous pouvons concevoir une régulation qui s'adapte aux exigences de processus spécifiques. La réponse du contrôleur peut être décrite en trois termes de réactivité : une erreur, de tolérance, au dépassement du point de consigne et au taux d'oscillation du système.

1.2.2 Terme proportionnel

Le terme proportionnel appelé plus souvent gain proportionnel applique un changement à la sortie qui est proportionnelle a la valeur d'erreur courante. Un gain élevé provoque un grand changement à la sortie pour un petit changement de l'erreur. Si le gain est trop haut, le système peut devenir instable. Au contraire, un gain trop faible aboutit à une faible réponse de la sortie en réaction à une grande erreur d'entrée. Si le gain proportionnel est trop bas, il peut être trop faible pour répondre aux perturbations du système.

En l'absence de perturbation, un contrôle proportionnel pur ne se positionnera pas à sa valeur cible, mais conservera un état d'erreur statique qui est une fonction du gain proportionnel et du gain du processus. Malgré la compensation de l'état stationnaire, tant la théorie des systèmes asservis que la pratique industrielle indiquent que c'est le terme proportionnel qui devrait contribuer à la plus grande partie du changement de la sortie.

1.2.3 Terme intégral

La contribution du terme intégral est proportionnelle à l'amplitude de l'erreur et à sa durée. La somme des erreurs instantanées au fil du temps (intégration) donne la compensation accumulée qui devrait avoir été corrigée précédemment. L'intégration de l'erreur est alors multipliée par le gain d'intégral et ajoutée à la sortie du contrôleur.

Le terme intégral lorsqu'il est ajouté augmente le mouvement du processus vers la consigne, il élimine l'erreur de statisme qui se produit avec un régulateur proportionnel seul. Cependant, puisque le terme intégral doit répondre aux erreurs accumulées par le passé, il peut causer un dépassement de la valeur de consigne actuelle (dépasser le point de consigne et puis créer un écart dans l'autre sens).

1.2.4 Terme dérivé

Le taux de variation de l'erreur du processus est calculé en déterminant la pente de l'erreur au cours du temps (c'est-à-dire sa dérivée première en relation avec le temps) et en multipliant ce taux de changement par le gain de dérivé. Le terme dérivé ralentit le taux de variation de la sortie du régulateur, cet effet est plus visible à proximité du point de consigne du contrôleur. Par conséquent, le contrôle dérivé est utilisé pour réduire l'ampleur du dépassement que produit la composante intégrale et pour améliorer la stabilité de la combinaison contrôleur processus.

1.2.5 Réglage de la boucle

Si les paramètres du contrôleur PID (les gains des termes proportionnel, intégral et dérivé) sont mal choisis, l'entrée du processus contrôlé peut être instable, c'est-à-dire sa sortie diverge, avec ou sans oscillation et, est limitée seulement par la saturation ou la rupture mécanique. Le réglage fin d'une boucle de contrôle consiste en l'ajustement de ses paramètres de contrôle (gain proportionnel, gain intégral, gain dérivé) aux valeurs optimums pour la réponse désirée.

1.2.6 Réglage manuel

Une méthode de réglage simple consiste à régler les valeurs **I** et **D** à zéro. Augmentons la valeur de **P** jusqu'à ce que la sortie oscille, **P** devrait être paramétré approximativement à la moitié de cette valeur pour diminuer d'un quart l'amplitude de ce type de réponse. Augmentons sa valeur pour que n'importe quelle compensation soit correcte dans un temps raisonnable pour le processus. Cependant, une valeur trop élevée apporte de l'instabilité. Ensuite, augmentons la valeur de **D** pour que la réponse soit suffisamment rapide pour atteindre sa référence après une perturbation de charge. Cependant, une valeur trop grande de **D** provoquera une réponse excessive et un dépassement. Un réglage de boucle PID rapide a un dépassement léger pour atteindre le point de consigne plus rapidement, cependant, certains systèmes ne peuvent accepter de dépassement, dans ce cas, une boucle fermée sur-amortie est nécessaire, cela requière une valeur **P** significativement plus basse que celle provoquant l'oscillation.

1.3 RTAI

La *Real Time Application Interface* (RTAI) Interface d'application temps réel est utilisée pour fournir la meilleure performance temps réel. Le noyau patché RTAI permet d'écrire des applications avec des contraintes temporelles strictes. RTAI donne la possibilité d'avoir des logiciels comme ceux de génération de pas qui ont besoin d'un timing précis.

1.3.1 ACPI

L'Advanced Configuration and Power Interface (ACPI) a de nombreuses et différentes fonctions, dont la plupart interfèrent avec les performances du système temps réel. (Pour par exemple: la gestion de l'énergie, la réduction de puissance du processeur, la variation de fréquence du CPU, etc.) Le noyau LinuxCNC (et probablement tous les noyaux RTAI-patché) ont les fonctions ACPI désactivées. ACPI prend également soin de mettre hors tension le système après qu'un arrêt système a été commandé, et c'est pourquoi vous pourriez avoir besoin de presser sur le bouton d'alimentation pour éteindre complètement votre ordinateur. Le groupe RTAI a amélioré cela dans les versions récentes, de sorte que votre système LinuxCNC peut éteindre le système par lui-même.

Première partie

Configuration de LinuxCNC

Chapitre 2

Test des capacités temps réel

2.1 Test de latence

Ce test est le premier test qui doit être effectué sur un PC pour savoir si celui-ci est capable de piloter une machine CNC.

La latence correspond au temps pris par le PC pour stopper ce qui est en cours et répondre à une requête externe. Dans notre cas, la requête est l'horloge qui sert au cadencement des impulsions de pas. Plus basse est la latence, plus rapide pourra être l'horloge, plus rapides et plus douces seront les impulsions de pas.

La latence est de loin plus importante que la vitesse du CPU. Un vieux Pentium III qui répond aux interruptions avec 10 microsecondes entre chacune, peut donner de meilleurs résultats que le dernier modèle de μP ultra rapide.

Le CPU n'est pas le seul facteur déterminant le temps de latence. Les cartes mères, les cartes vidéo, les ports USB et de nombreuses autres choses peuvent détériorer le temps de latence. La meilleure façon de savoir si le matériel envisagé est apte, c'est d'exécuter *un test de latence*.

La seule façon de découvrir ce qu'il en est sur un PC est d'exécuter le test de latence de HAL. Pour exécuter ce test, ouvrir une fenêtre de terminal à partir de *Applications* → *Accessoires* → *Terminal* et exécuter la commande suivante:

```
latency-test
```

Une fenêtre comme ci-dessous devrait s'ouvrir:

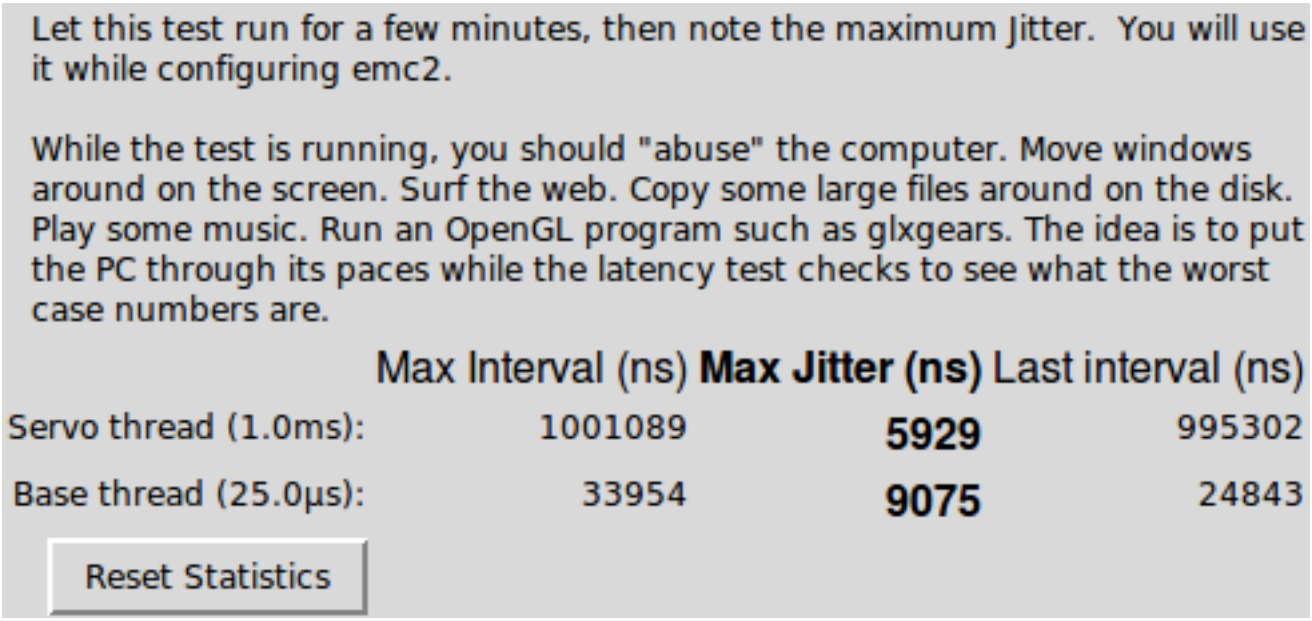


FIGURE 2.1 – Test de latence de HAL

Alors que le test est en cours d'exécution, il faut charger l'ordinateur au maximum. Déplacer les fenêtres sur l'écran. Surfer sur le Web. Écouter de la musique. Exécuter un programme OpenGL comme glxgears. L'idée est de charger le PC au maximum pour que le temps de latence soit mesuré dans le cas le plus défavorable et donc, connaitre la latence maximale.

Note
Ne pas exécuter LinuxCNC ou Stepconf pendant que latency-test est en cours d'exécution.

La colonne *max jitter* et la ligne *Base Thread* de l'exemple ci-dessus donne *9075*. Ce qui représente 9075 nanosecondes, soit 9.075 microsecondes. Noter ce nombre et l'entrer dans Stepconf quand il sera demandé.

Dans cet exemple de test de latence il n'a fallu que quelques secondes pour afficher cette valeur. Il est toutefois préférable de le laisser tourner pendant plusieurs minutes. Parfois même, dans le pire des cas, rien ne provoque de latence ou seulement des actions particulières. Par exemple, une carte mère Intel marchait très bien la plupart du temps, mais toutes les 64 secondes elle avait une très mauvaise latence de 300μs. Heureusement, il existe un correctif.

Alors, comment interpréter les résultats? Si le résultat de Max Jitter est en dessous d'environ 15-20 μs (15000-20000 nanosecondes), l'ordinateur pourra donner d'excellents résultats pour la génération logicielle des pas. Si le temps de latence est à plus de 30-50 microsecondes, de bons résultats seront obtenus, mais la vitesse maximum sera un peu faible, spécialement si des micropas sont utilisés ou si le pas de la vis est fin. Si les résultats sont de 100μS ou plus (100,000 nanosecondes), alors le PC n'est pas un bon candidat à la génération des pas. Les résultats supérieurs à 1 milliseconde (1,000,000 nanosecondes) éliminent, dans tous les cas, ce PC pour faire tourner LinuxCNC, en utilisant des micropas ou pas.

Note
Si une latence élevée est obtenue, il peut être possible de l'améliorer. Un PC avait une très mauvaise latence (plusieurs millisecondes) en utilisant la carte graphique interne. Un carte graphique d'occasion à $5US a résolu le problème. LinuxCNC n'exige pas de matériel de pointe.

2.2 Adresses des ports

Pour ceux qui construisent leur matériel, il est facile et économique d'augmenter le nombre d'entrées sorties d'un PC en lui ajoutant une carte PCI fournissant un ou deux ports parallèles supplémentaires. Faire suivre ces ports d'une couche d'opto-

isolation est utile pour éviter les courts circuits pouvant détruire la carte, voir même toute la carte mère. LinuxCNC supporte un maximum de 8 ports parallèles.

Certaines parmi les bonnes cartes parallèles sont à base de chipset Netmos. Elles fournissent un signal +5V bien propre, elles fournissent un ou deux ports parallèles.

Pour trouver les adresses d'entrées/sorties de ces cartes, ouvrir une console et utiliser la commande en ligne:

```
lspci -v
```

Rechercher ensuite dans la liste de matériel fourni, le nom du chipset de la nouvelle carte, dans cette exemple c'est l'entrée NetMos Technology pour une carte à deux ports:

```
0000:01:0a.0 Communication controller: \
      Netmos Technology PCI 9815 Multi-I/O Controller (rev 01)
      Subsystem: LSI Losgic / 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]
```

Après expérimentation, il se trouve que le premier port (incorporé à la carte) utilise la troisième adresse de la liste (c000) et le deuxième port (raccordé par une nappe) utilise la première adresse (b800).

Il est alors possible d'ouvrir dans l'éditeur le fichier .hal de la machine et d'insérer l'adresse trouvée à l'endroit approprié.

```
loadrt hal\_parport cfg="0x378 0xc000"
```

Noter la présence des guillemets "" encadrant les deux adresses, ils sont obligatoires dès qu'il y a plus d'une carte.

Il est nécessaire également d'ajouter les fonctions de lecture (read) et d'écriture (write) pour la nouvelle carte. Par exemple:

```
addf parport.1.read base-thread 1
addf parport.1.write base-thread -1
```

Noter que les valeurs peuvent être différentes de celles de cet exemple. Les cartes Netmos sont Plug-N-Play, elles peuvent donc changer leur adressage selon le connecteur PCI dans lequel elles sont placées. Si l'installation des cartes PCI de la machine est modifiée, ne pas oublier de vérifier leurs adresses avant de lancer LinuxCNC.

Chapitre 3

Configuration de LinuxCNC

3.1 Script de lancement

LinuxCNC est lancé par le fichier de script *linuxcnc*.

```
linuxcnc [options] [<ini-file>]
```

Avec les options suivantes: * *-v* = verbose - informations de fonctionnement * *-d* = commande d'écho à l'écran pour le débogage

Le fichier de script *linuxcnc* lit le fichier ini puis lance LinuxCNC. La section [HAL] du fichier ini, spécifie l'ordre de chargement des fichiers de HAL, si plusieurs sont utilisés. Après que les fichiers HAL soient chargés, l'interface graphique est chargée à son tour puis le fichier HAL POSTGUI. Si des objets pyvcp ont été créés avec des pins de HAL, le fichier *postgui.hal* doit effectuer les raccordements à ces pins, se reporter à la section HAL pour plus de détails.

Si aucun fichier ini n'est passé en argument au script *linuxcnc*, le sélecteur de configuration est lancé pour permettre à l'utilisateur de choisir parmi les exemples de configuration existants.

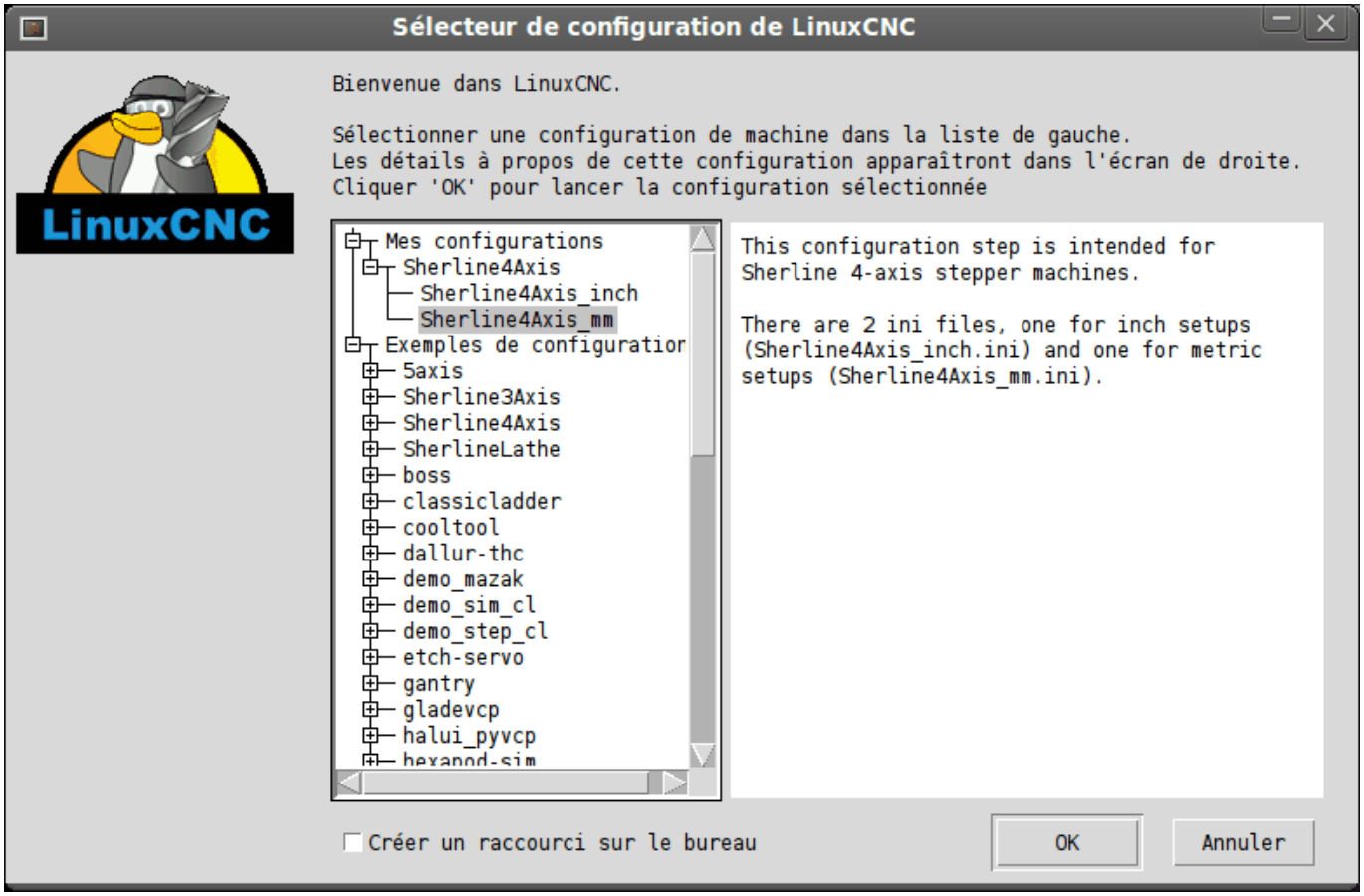


FIGURE 3.1 – Sélecteur de configuration

3.2 Fichiers utilisés pour la configuration

LinuxCNC est entièrement configuré avec des fichiers textes classiques. Tous ces fichiers peuvent être lus et modifiés dans n'importe quel éditeur de texte disponible dans toute distribution Linux. [1] Soyez prudent lorsque vous modifierez ces fichiers, certaines erreurs pourraient empêcher le démarrage de LinuxCNC. Ces fichiers sont lus à chaque fois que le logiciel démarre. Certains d'entre eux sont lus de nombreuses fois pendant l'exécution de LinuxCNC.

Les fichiers de configuration inclus:
– *INI* Le fichier ini écrase les valeurs par défaut compilées dans le code de LinuxCNC. Il contient également des sections qui sont lues directement par HAL (Hardware Abstraction Layer, couche d'abstraction matérielle).
– *HAL* Les fichiers hal installent les modules de process, ils créent les liens entre les signaux de LinuxCNC et les broches spécifiques du matériel.
– *VAR* Ce fichier contient une suite de numéros de variables. Ces variables contiennent les paramètres qui seront utilisés par l'interpréteur. Ces valeurs sont enregistrées et réutilisées d'une exécution à l'autre.
– *TBL* Ce fichier contient les informations relatives aux outils. Voir la section *Fichier d'outils* du Manuel de l'utilisateur pour plus d'infos.
– *NML* Ce fichier configure les voies de communication utilisées par LinuxCNC. Il est normalement réglé pour lancer toutes les communications avec un seul ordinateur, peut être modifié pour communiquer entre plusieurs ordinateurs.
– *.linuxcncrc* Ce fichier enregistre des informations spécifiques à l'utilisateur, il a été créé pour enregistrer le nom du répertoire lorsque l'utilisateur choisit sa première configuration de LinuxCNC. [2]

1. Ne pas confondre un éditeur de texte et un traitement de texte. Un éditeur de texte comme gedit ou kwrite produisent des fichiers uniquement en texte. Les lignes de textes sont séparées les unes des autres. Un traitement de texte comme Open Office produit des fichiers avec des paragraphes, des mises en formes des mots. Ils ajoutent des codes de contrôles, des polices de formes et de tailles variées etc. Un éditeur de texte n'a rien de tout cela.
2. Habituellement, ce fichier est dans le répertoire home de l'utilisateur (ex: */home/robert/*)

Les éléments avec le repère *(hal)* sont utilisés seulement pour les fichiers de HAL en exemples. C'est une bonne convention. D'autres éléments sont utilisés directement par LinuxCNC et doivent toujours avoir la section et le nom donné à l'item.

3.3 Double passe (TWOPASS)

LinuxCNC 2.5 supporte le processus dit TWOPASS des fichiers de configuration hal, ce qui aide à la modularité des fichiers hal et améliore leur lisibilité. (les fichiers Hal sont spécifiés dans le fichier ini de LinuxCNC, dans l'instance HAL sous la forme *[HAL]HALFILE=nomdufichier*.

Normalement, un jeu de un ou plusieurs fichiers de configuration HAL doivent utiliser une seule et unique ligne loadrt pour charger le module du kernel qui pourra gérer de multiples instances d'un même composant. Par exemple: si vous utilisez une portes AND à deux entrées, composant (and2), à trois endroits différents de votre configuration, vous ne devez avoir que cette seule ligne quelque part pour le spécifier:

```
loadrt and2 count=3
```

Ce qui fournira finalement les composants and2.0, and2.1, and2.2.

Les configurations seront plus lisibles si vous spécifiez les composants sous la forme names=option quand c'est supporté, par exemple:

```
loadrt and2 names=aa,ab,ac
```

Ce qui nommera les composants aa, ab, ac.

Il pourrait apparaitre un problème de maintenance pour garder la trace des composants et de leur noms après avoir ajouté (ou enlevé) un composant, vous devrez trouver et mettre à jour, la ligne de directives de loadrt, applicable à ce composant.

Le processus TWOPASS est activé par inclusion d'un paramètre dans le fichier ini:

```
[HAL]TWOPASS=anything
```

Avec ce réglage, vous pouvez avoir de multiples spécifications comme:

```
loadrt and2 names=aa
...
loadrt and2 names=ab,ac
...
loadrt and2 names=ad
```

Ces commandes peuvent être placées dans différents fichiers HALFILES. Les HALFILES sont traités dans leur ordre d'apparition dans le fichier ini.

Avec le processus double passe, tous les [HAL]HALFILES sont lus une première fois et les multiples apparitions de la directive loadrt sont cumulées pour chaque module. Aucune commande hal n'est exécutée lors de cette passe initiale.

Après la passe initiale, les modules sont automatiquement chargés en nombre égal au nombre total lors de l'utilisation de count=option ou de tous les noms spécifiés individuellement lors de l'utilisation de names=option.

Une seconde passe est alors faite pour exécuter toutes les autres instructions de hal spécifiées dans les HALFILES. Les commandes addf qui associent les fonctions de composants avec l'exécution du thread sont exécutées selon leur ordre d'apparition avec les autres commandes dans cette seconde passe.

Bien que vous puissiez utiliser indifféremment les options avec count= ou names=, elles sont toutefois exclusives. Un seul type peut être utilisé pour un même module.

Le processus TWOPASS n'est pas effectif lors de l'usage de names=option. Cette option permet d'avoir un nom unique qui soit mnémonique ou plus pertinent avec la configuration. Par exemple: si vous utilisez un composant *dérivé* pour estimer la vitesse et l'accélération de chacun des coordonnées (x,y,z), utiliser la méthode count= donnera un composant au nom ésotérique comme ddt.0, ddt.1, ddt.2, etc.

Alternativement, l'utilisation de names=option comme:

```
loadrt ddt names=xvit,yvit,zvit
...
loadrt ddt names=xaccel,yaccel,zaccel
```

donnera des composants plus parlants, nommés xvit,yvit,zvit, xaccel,yaccel, zaccel.

Beaucoup de composants fournis avec la distribution ont été créés avec *comp utility* et supportent la méthode names=option. Il s'agit notamment de composants logiques qui sont les briques de beaucoup de configurations HAL.

Exemples d'inclusions:

```
and2,ddt,deadzone,flipflop,or2,or4,mux2,mux4,scale,sum2,timedelay,lowpass
```

et beaucoup d'autres.

Les composants utilisateur créés avec *comp utility* supportent également automatiquement la méthode names=option. En plus des composants générés avec *comp utility*, quelques autres composants comme *encoder* et *pid* supportent aussi names=option.

3.4 Organisation du fichier ini

Organisation du fichier ini

Un fichier ini typique suit une organisation simple;

- les commentaires.
- les sections.
- les variables.

Chacun de ces éléments est séparé, sur une seule ligne. Chaque fin de ligne ou retour chariot crée un nouvel élément.

3.4.1 Les commentaires

Une ligne de commentaires débute avec un **;** ou un **#**. Si le logiciel qui analyse le fichier ini rencontre l'un ou l'autre de ces caractères, le reste de la ligne est ignoré. Les commentaires peuvent être utilisés pour décrire ce que font les éléments du fichier ini.

```
; Ceci est le fichier de configuration de ma petite fraiseuse.
```

Des commentaires peuvent également être utilisés pour choisir entre plusieurs valeurs d'une seule variable.

```
DISPLAY = axis
# DISPLAY = touchy
```

Dans cette liste, la variable DISPLAY est positionnée sur axis puisque l'autre est commentée. Si quelqu'un édite une liste comme celle-ci et par erreur, dé-commente deux lignes, c'est la première rencontrée qui sera utilisée.

Noter que dans une ligne de variables, les caractères **#** et **;** n'indiquent pas un commentaire.

```
INCORRECT = valeur     # et un commentaire

# Commentaire correct
CORRECT = valeur
```

3.4.2 Les sections

Les différentes parties d'un fichier .ini sont regroupées en sections. Une section commence par son nom en majuscules entre crochets [UNE_SECTION]. L'ordre des sections est sans importance.

Les sections suivantes sont utilisées par LinuxCNC:

– *[EMC]* informations générales.
– *[DISPLAY]* sélection du type d'interface graphique.
– *[FILTER]* sélection d'un programme de filtrage.
– *[RS274NGC]* ajustements utilisés par l'interpréteur de g-code.
– *[EMCMOT]* réglages utilisés par le contrôleur de mouvements temps réel.
– *[TASK]* réglages utilisés par le contrôleur de tâche.
– *[HAL]* spécifications des fichiers .hal.
– *[HALUI]* commandes MDI utilisées par HALUI.
– *[TRAJ]* réglages additionnels utilisés par le contrôleur de mouvements temps réel.
– *[AXIS_n]* groupes de variables relatives à chaque axe.
– *[EMCIO]* réglages utilisés par le contrôleur d'entrées/sorties.

3.4.3 Les variables

Une ligne de variables est composée d'un nom de variable, du signe égal (=) et d'une valeur. Tout, du premier caractère non blanc qui suit le signe = jusque la fin de la ligne, est passé comme valeur à la variable. Vous pouvez donc intercaler des espaces entre les symboles si besoin. Un nom de variable est souvent appelé un mot clé.

Les paragraphes suivants détaillent chaque section du fichier de configuration, en utilisant des exemples de variables dans les lignes de configuration.

Certaines de ces variables sont utilisées par LinuxCNC. Elles doivent toujours utiliser le nom de section et le nom de variable dans leur appellation. D'autres variables ne sont utilisées que par HAL. Les noms des sections et les noms des variables indiquées, sont ceux qui sont utilisés dans les exemples de fichiers de configuration.

Les variables personnalisées peuvent être utilisées dans vos fichiers HAL avec la syntaxe suivante:

```
MACHINE = MaVariable
```

3.4.4 Sections et variables utilisateur

Certaines configurations utilisent des sections utilisateur et des variables personnalisées pour regrouper les paramètres en un seul emplacement pour améliorer la lisibilité du fichier ini.

Pour utiliser une section de variable utilisateur dans un fichier HAL, ajouter la section et la variable dans le fichier INI.

Exemple de section utilisateur

```
[OFFSETS]
OFFSET_1 = 0.1234
```

Pour ajouter une variable utilisateur à une section LinuxCNC, inclure simplement cette variable dans la section souhaitée.

Exemple de variable utilisateur

```
[AXIS_0]
TYPE = LINEAR
...
SCALE = 16000
```

Pour utiliser une variable utilisateur dans un fichier HAL, utiliser les noms de section et de variable en lieu et place de leurs valeurs.

Exemple d'utilisation dans un fichier HAL

```
setp offset.1.offset [OFFSETS]OFFSET_1
setp stepgen.0.position-scale [AXIS_0]SCALE
```

Note
La valeur stockée dans la variable doit correspondre au type spécifié pour la pin du composant.

3.5 Détails des sections du fichier ini

3.5.1 Section [EMC]

– *VERSION = $Revision: 1.5 $* - Le numéro de version du fichier INI. La valeur indiquée ici semble étrange, car elle est automatiquement mise à jour lors de l'utilisation du système de contrôle de révision. C'est une bonne idée de changer ce numéro à chaque fois que vous modifiez votre fichier. Si vous voulez le modifier manuellement, il suffit de changer le numéro sans toucher au reste.
– *MACHINE = ma machine* - C'est le nom du contrôleur, qui est imprimé dans le haut de la plupart des fenêtres. Vous pouvez insérer ce que vous voulez ici tant que ça reste sur une seule ligne.
– *DEBUG = 0* - Niveau de débogage 0 signifie qu'aucun message ne sera affiché dans le terminal pendant le fonctionnement de LinuxCNC. Les drapeaux de débogage ne sont généralement utiles que pour les développeurs.

3.5.2 Section [DISPLAY]

Les différentes interfaces graphiques utilisent différentes options qui ne sont pas supportées par toutes les interfaces utilisateur. Les deux principales interfaces pour LinuxCNC sont *AXIS* et *Touchy*. Axis est une interface pour une utilisation avec un ordinateur classique et son moniteur, Touchy est à utiliser avec les ordinateurs à écran tactile. Pour plus d'informations, voire la section Interfaces du Manuel de l'utilisateur.
– *DISPLAY = axis* - Le nom de l'interface graphique à utiliser. Les options disponibles sont les suivantes: *axis*, *touchy*, *keystick*, *mini*, *tklinuxcnc*, *xlinuxcnc*,
– *POSITION_OFFSET = RELATIVE* - Le système de coordonnées (RELATIVE ou MACHINE) à utiliser au démarrage de l'interface utilisateur. Le système de coordonnées RELATIVE reflète le G92 et le décalage d'origine G5x actuellement actifs.
– *POSITION_FEEDBACK = ACTUAL* - Valeur de la position (COMMANDED ou ACTUAL) à afficher au démarrage de l'interface utilisateur. La position COMMANDED est la position exacte requise par LinuxCNC. La position ACTUAL est la position retournée par l'électronique des moteurs.
– *MAX_FEED_OVERRIDE = 1.2* - La correction de vitesse maximum que l'opérateur peut utiliser. 1.2 signifie 120% de la vitesse programmée.
– *MIN_SPINDLE_OVERRIDE = 0.5* - Correction de vitesse minimum de broche que l'opérateur pourra utiliser. 0.5 signifie 50% de la vitesse de broche programmée. (utile si il est dangereux de démarrer un programme avec une vitesse de broche trop basse).
– *MAX_SPINDLE_OVERRIDE = 1.0* - Correction de vitesse maximum de broche que l'opérateur pourra utiliser. 1.0 signifie 100% de la vitesse de broche programmée.
– *DEFAULT_SPINDLE_SPEED = 100* - Vitesse de broche par défaut quand celle-ci démarre en mode manuel. Dans AXIS, si cette variable est absente, la vitesse de démarrage est alors fixée à 1 tr/mn. Ce n'est pas la vitesse minimum.
– *PROGRAM_PREFIX = ~/linuxcnc/nc_files* - Répertoire par défaut des fichiers de g-codes et emplacement des M-codes définis par l'utilisateur. Les recherches de fichiers s'effectueront d'abords dans cet emplacement, avant les chemins des sous-programmes et des fichiers M utilisateur, si il est spécifié dans la section [RS274NGC].
– *INTRO_GRAPHIC = linuxcnc.gif* - L'image affichée sur l'écran d'accueil.
– *INTRO_TIME = 5* - Durée d'affichage de l'écran d'accueil.
– *CYCLE_TIME = 0.05* - Cycle time in seconds that display will sleep between polls.
Les éléments suivants sont utilisés uniquement si AXIS est sélectionné comme programme d'interface utilisateur.
– *DEFAULT_LINEAR_VELOCITY = .25* - Vitesse minimum par défaut pour les jogs linéaires, en unités machine par seconde. Seulement utilisé dans l'interface AXIS.
– *MIN_VELOCITY = .01* - Valeur approximative minimale du curseur de vitesse de jog.
– *MAX_LINEAR_VELOCITY = 1.0* - Vitesse maximum par défaut pour les jogs linéaires, en unités machine par seconde. Seulement utilisé dans l'interface AXIS.
– *MIN_LINEAR_VELOCITY = .01* - Approximativement la valeur minimale du curseur de vitesse de jog.
– *DEFAULT_ANGULAR_VELOCITY = .25* - Vitesse minimum par défaut pour les jogs angulaires, en unités machine par seconde. Seulement utilisé dans l'interface AXIS.
– *MIN_ANGULAR_VELOCITY = .01* - Valeur approximative minimale du curseur de vitesse angulaire de jog.
– *MAX_ANGULAR_VELOCITY = 1.0* - Vitesse maximum par défaut pour les jogs angulaires, en unités machine par seconde. Seulement utilisé dans l'interface AXIS.
– *INCREMENTS = 1 mm, .5 mm, ...* - Définit les incréments disponibles pour le jog incrémental. Les incréments peuvent être utilisés pour remplacer la valeur par défaut. Ces valeurs doivent contenir des nombres décimaux (ex. 0.1000) ou des nombres

fractionnaires (ex. 1/16), éventuellement suivis par une unité parmi *cm*, *mm*, *um*, *inch*, *in* ou *mil*. Si aucune unité n'est spécifiée, les unités natives de la machine seront utilisées.
- Distances métriques et impériales peuvent être mélangées
  INCREMENTS = 1 inch, 1 mil, 1 cm, 1 mm, 1 um sont des entrées valides.
- *OPEN_FILE = /chemin/complet/du/fichier.ngc* Le fichier ngc à utiliser au démarrage d'AXIS. Utilisez une chaîne vide "" et aucun fichier ne sera chargé au démarrage.
- *EDITOR = gedit* - L'éditeur à utiliser lors du choix *Éditer fichier* du menu d'AXIS, pour éditer le G-code. Ceci doit être configuré pour que cet item de menu s'active. Une autre possibilité valide est: *gnome-terminal -e nano*.
- *TOOL_EDITOR = tooledit* - L'éditeur de texte à utiliser pour éditer les tables d'outils. (par exemple en sélectionnant "Fichiers > Éditer la table. d'outils" dans le menu d'Axis). D'autres entrées comme *gedit*, *gnome-terminal -e vim*, *gvim* ou *nano* sont valides.
- *PYVCP = /filename.xml* - Le fichier de description du panneau PyVCP. Voir la section PyVCP.
- *LATHE = 1* - Passe l'affichage en mode tour, avec vue de dessus et la visu soit en rayon, soit en diamètre.
- *GEOMETRY = XYZABCUVW* - Contrôle de prévisualisation du parcours d'outil d'un mouvement rotatif. Cet item consiste en une suite de lettre d'axe, optionnellement précédé d'un signe **-**. Seuls, les axes définis par **[TRAJ]AXES** peuvent être utilisés. Cette séquence spécifie l'ordre dans lequel l'effet de chaque axe est appliqué. Un signe **-** inverse le sens de la rotation. La chaine GEOMETRY correcte dépend de la configuration de la machine et de la cinématique utilisée pour la contrôler. La chaine exemple GEOMETRY=XYZBCUVW est pour une machine à 5 axes pour laquelle la cinématique déplace UVW en coordonnées système de l'outil et XYZ déplace la pièce en coordonnées système. L'ordre des lettres est important, parce qu'il donne expressément l'ordre dans lequel les différentes transformations seront appliquées. Par exemple: tourner autour de C puis de B est différent de tourner autour de B puis de C. La géométrie n'a pas d'effet sans rotation d'axes.
- *ARCDIVISION = 64* - Ajuste la valeur de prévisualisation des arcs. Les arcs sont visualisés en les divisant par un nombre de lignes droites; un semi-cercle est divisé en *ARCDIVISION* de tronçons. Les valeurs élevées donnent une meilleure précision à la pré-visualisation, mais sont plus lentes et donne un écran plus saccadé. Les petites valeurs sont moins précises mais plus rapides, l'affichage résultant est plus rapide. La valeur par défaut de 64 signifie qu'un cercle de 3 pouces maximum sera affiché dans moins de 3 centièmes de mm, (.03%). [3]
- *MDI_HISTORY_FILE = -* Le nom du fichier d'historique des commandes MDI. Si rien n'est spécifié, Axis enregistrera cet historique dans *.axis_mdi_history* dans le répertoire home de l'utilisateur. C'est très pratique dans le cas de multiples configurations sur la même machine.
- *HELP_FILE = tklinucnc.txt* - Chemin du fichier d'aide (non utilisé avec AXIS).

3.5.3 Section [FILTER]

AXIS a la possibilité d'envoyer les fichiers chargés au travers d'un programme de filtrage. Ce filtrage peut réaliser toutes sortes de tâches. Parfois aussi simple que s'assurer que le programme se termine bien par M2, ou parfois aussi compliqué que détecter si le fichier d'entrée est une image et en générer le G-code pour graver la forme qu'il à ainsi défini. La section *[FILTER]* du fichier ini, contrôle comment les filtres fonctionnent. Premièrement, pour chaque type de fichier, écrire une ligne *PROGRAM_EXTENSION*. Puis, spécifier le programme à exécuter pour chaque type de filtre. Ce programme reçoit le nom du fichier d'entrée dans son premier argument, il doit écrire le code RS274/NGC sur la sortie standard. C'est cette sortie qui sera affichée dans la zone de texte, pré-visualisée dans la zone du parcours d'outil et enfin, exécutée par LinuxCNC quand il sera mis en marche.

```
PROGRAM_EXTENSION = .extension Description
```

Si votre fichier de sortie est tout en majuscules, vous devez ajouter la ligne suivante:

```
PROGRAM_EXTENSION = .NGC XYZ Post Processor
```

Les lignes suivantes ajoutent le support pour le convertisseur *image-to-gcode* fourni avec LinuxCNC:

```
PROGRAM_EXTENSION = .png,.gif,.jpg Greyscale Depth Image
    png = image-to-gcode
    gif = image-to-gcode
    jpg = image-to-gcode
```

Il est également possible de spécifier un interpréteur:

3. Dans LinuxCNC 2.4 et précédents, la valeur par défaut était de 128.

```
PROGRAM_EXTENSION = .py Python Script
    py = python
```

De cette façon, n'importe quel script Python pourra être ouvert et ses sorties seront traitées comme du g-code. Un exemple de script de ce genre est disponible: nc_files/holecircle.py. Ce script crée le G-code pour percer une série de trous séquents à la périphérie d'un cercle. De nombreux générateurs de G-code sont par ailleurs disponibles sur le wiki: à la page des générateurs de G-code.

Si la variable d'environnement AXIS_PROGRESS_BAR est activée, alors les lignes écrites sur stderr de la forme

```
FILTER_PROGRESS=%d
```

activeront la barre de progression d'AXIS qui donnera le pourcentage. Cette fonctionnalité devrait être utilisée par tous les filtres susceptibles de fonctionner pendant un long moment.

Les filtres Python doivent utiliser la fonction *print* pour sortir le résultat dans Axis.

Cet exemple de programme filtre un fichier et ajoute un axe W correspondant à l'axe Z. Il marchera selon la présence d'un espace entre chaque mot d'axe.

```
#! /usr/bin/env python

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:])
```

3.5.4 Section [RS274NGC]

– *PARAMETER_FILE = monfichier.var* - Le fichier situé dans le même répertoire que le fichier ini qui contiendra les paramètres utilisés par l'interpréteur (enregistré entre chaque lancement).
– *ORIENT_OFFSET = 0* - Une valeur flottante ajoutée au paramètre R d'une opération d'orientation de la broche par M19. Utilisée pour définir une position zéro quelconque quelle que soit l'orientation de montage du codeur de broche.

– *RS274NGC_STARTUP_CODE = G17 G20 G40 G49 G64 P0.001 G80 G90 G92 G94 G97 G98* - Une chaine de codes NGC qui sera utilisée pour initialiser l'interpréteur. Elle ne se substitue pas à la spécification des G-codes modaux du début de chaque fichier ngc. Les codes modaux des machines diffèrent, ils pourraient être modifiés par les G-codes interprétés plutôt dans la session.
– *SUBROUTINE_PATH = ncsubroutines:/tmp/testsubs:lathesubs:millsubs* - Spécifie une liste, séparée par (:) d'au maximum 10 répertoires dans lesquels seront cherchés les fichier de sous-programme spécifiés dans le g-code. Ces répertoires sont inspectés après que ne le soit [DISPLAY]PROGRAM_PREFIX (si il est spécifié) et avant que ne le soit [WIZARD]WIZARD_ROOT (si il est spécifié). les recherches s'effectuent dans l'ordre dans lequel les chemins sont listés. La première occurrence avec le sous-programme recherché est utilisée. Les répertoires sont spécifiés relativement au répertoire courant du fichier ini ou par des chemins absolus. La liste ne doit contenir aucun espace blanc.
– *USER_M_PATH = myfuncs:/tmp/mcodes:experimentalmcodes* - Spécifie une liste de répertoires, séparés par (:) (sans aucun espace blanc) pour les fonctions définies par l'utilisateur. Les répertoires sont spécifiés par rapport au répertoire courant pour les fichiers ini ou en chemins absolus. La liste ne doit contenir aucun espace blanc.
– *USER_DEFINED_FUNCTION_MAX_DIRS=5* - Défini le nombre maximum de répertoires au moment de la compilation. Une recherche est faite pour chaque fonction utilisateur définie possible, typiquement *M100* à *M199*.
L'ordre de recherche est le suivant:

1. [DISPLAY]PROGRAM_PREFIX (si il est spécifié)
2. Si [DISPLAY]PROGRAM_PREFIX n'est pas spécifié, cherche dans le répertoire par défaut: nc_files
3. Recherche ensuite dans chaque répertoire de la liste [RS274NGC]USER_M_PATH Le premier M1xx trouvé au cours de la recherche est utilisé pour chaque M1xx.

Note
[WIZARD]WIZARD_ROOT est un chemin de recherche valide mais l'assistant n'est pas encore complétement implémenté et les résultats, découlant de son utilisation, sont imprévisibles.

3.5.5 Section [EMCMOT]

D'autres entrées peuvent être rencontrées dans cette section, elles ne doivent pas être modifiées.
– *EMCMOT = motmod* - Utilise typiquement le nom du contrôleur de mouvement.
– *BASE_PERIOD = 50000* - (HAL) Période de base des tâches, exprimée en ns.
– *SERVO_PERIOD = 1000000* - (hal) Période de la tâche *Servo*, exprimée également en nanosecondes.
– *TRAJ_PERIOD = 1000000* - (hal) Période du *planificateur de trajectoire*, exprimée en nanosecondes.

3.5.6 Section [TASK]

– *TASK = milltask* - Indique le nom de la *tâche* exécutable. La tâche réalise différentes actions, telles que communiquer avec les interfaces utilisateur au dessus de NML, communiquer avec le planificateur de mouvements temps réel dans la mémoire partagée non-HAL, et interpréter le g-code. Actuellement il n'y a qu'une seule tâche exécutable qui fait sens pour 99,9% des utilisateurs, milltask.
– *CYCLE_TIME = 0.010* - Période exprimée en secondes, à laquelle TASK va tourner. Ce paramètre affecte l'intervalle de polling lors de l'attente de la fin d'un mouvement, lors de l'exécution d'une pause d'instruction et quand une commande provenant d'une interface utilisateur est acceptée. Il n'est généralement pas nécessaire de modifier cette valeur.

3.5.7 Section [HAL]

– *TWOPASS=ON* - Utilise le processus *twopass* (double passe) pour charger les composants HAL. Avec le processus TWOPASS, tous les fichiers [HAL]HALFILES sont premièrement lus et les occurrences multiples des directives à loadrt pour chaque module sont cumulées. Aucune commande HAL n'est exécutée à la première passe.

– *HALFILE = example.hal* - Exécute le fichier *example.hal* au démarrage. Si *HALFILE* est spécifié plusieurs fois, les fichiers sont exécutés dans l'ordre de leur apparition dans le fichier ini. Presque toutes les configurations auront au moins un *HALFILE* . Les systèmes à moteurs pas à pas ont généralement deux de ces fichiers, un qui spécifie la configuration générale des moteurs *core_stepper.hal* et un qui spécifie le brochage des sorties *xxx_pinout.hal*.
– *HAL = command* - Exécute *command* comme étant une simple commande hal. Si *HAL* est spécifié plusieurs fois, les commandes sont exécutées dans l'ordre où elles apparaissent dans le fichier ini. Les lignes *HAL* sont exécutées après toutes les lignes *HALFILE*.
– *SHUTDOWN = shutdown.hal* - Exécute le fichier *shutdown.hal* quand LinuxCNC s'arrête. Selon les pilotes de matériel utilisés, il est ainsi possible de positionner les sorties sur des valeurs définies quand LinuxCNC s'arrête normalement. Cependant, parce qu'il n'y a aucune garantie que ce fichier sera exécuté (par exemple, dans le cas d'une panne de l'ordinateur), il ne remplace pas une véritable chaîne physique d'arrêt d'urgence ou d'autres dispositifs logiciels de protection des défauts de fonctionnement comme la pompe de charge ou le watchdog.
– *POSTGUI_HALFILE = example2.hal* - (Seulement avec les interfaces TOUCHY et AXIS) Exécute *example2.hal* après que l'interface graphique ait créé ses HAL pins.

3.5.8 Section [HALUI]

– *MDI_COMMAND = G53 G0 X0 Y0 Z0* - Une commande MDI peut être exécuté en utilisant *halui.mdi-command-00*. Incrémente le nombre pour chaque commande énumérée dans la section [HALUI].

3.5.9 Section [TRAJ]

La section [TRAJ] contient les paramètres généraux du module planificateur de trajectoires de EMCMOT. Vous n'aurez pas à modifier ces valeurs si vous utilisez LinuxCNC avec une machine à trois axes en provenance des USA. Si vous êtes dans une zone métrique, utilisant des éléments matériels métriques, vous pourrez utiliser le fichier *stepper_mm.ini* dans lequel les valeurs sont déjà configurées dans cette unité.
– *COORDINATES = X Y Z* - Les noms des axes à contrôler. X, Y, Z, A, B, C, U, V et W sont valides. Seuls les axes nommés dans *COORDINATES* seront acceptés dans le G-code. Cela n'a aucun effet sur l'ordonnancement des noms d'axes depuis le G-code (X- Y- Z-) jusqu'aux numéros d'articulations. Pour une *cinématique triviale*, X est toujours l'articulation 0, A est toujours l'articulation 3, U est toujours l'articulation 6 et ainsi de suite. Il est permis d'écrire les noms d'axe par paire (ex: X Y Y Z pour une machine à portique) mais cela n'a aucun effet.
– *AXES = 3* - Une unité de plus que le plus grand numéro d'articulation du système. Pour une machine XYZ, les articulations sont numérotées 0, 1 et 2. Dans ce cas, les AXES sont 3. Pour un système XYUV utilisant une *cinématique triviale*, l'articulation V est numérotée 7 et donc les AXES devraient être 8. Pour une machine à cinématique non triviale (ex: scarakins) ce sera généralement le nombre d'articulations contrôlées.
– *JOINTS = 3* - (Cette variable de configuration est utilisée seulement par Axis et non par le planificateur de trajectoire du contrôleur de mouvement.) Elle spécifie le nombre d'articulations (moteurs) que comporte le système. Par exemple, une machine XYZ avec un seul moteur pour chacun des 3 axes, comporte 3 articulations (joints). Une machine à portique avec un seul moteur sur deux de ses axes et deux moteurs sur le troisième axe, comporte 4 articulations (joints).
– *HOME = 0 0 0* - Coordonnées de l'origine machine de chaque axe. De nouveau, pour une machine 4 axes, vous devrez avoir 0 0 0 0. Cette valeur est utilisée uniquement pour les machines à cinématique non triviale. Sur les machines avec cinématique triviale, cette valeur est ignorée.
– *LINEAR_UNITS=<units>* - Le nom des unités utilisées dans le fichier INI. Les choix possibles sont *in*, *inch*, *imperial*, *metric*, *mm*. Cela n'affecte pas les unités linéaires du code NC (pour cela il y a les mots G20 et G21).
– *ANGULAR_UNITS=<units>* - Le nom des unités utilisées dans le fichier INI. Les choix possibles sont *deg*, *degree* (360 pour un cercle), *rad*, *radian* (2pi pour un cercle), *grad*, ou *gon* (400 pour un cercle). Cela n'affecte pas les unités angulaires du code NC. Dans le code RS274NGC, les mots A-, B- et C- sont toujours exprimés en degrés.
– *DEFAULT_VELOCITY = 0.0167* - La vitesse initiale de jog des axes linéaires, en unités par seconde. La valeur indiquée ici correspond à une unité par minute.
– *DEFAULT_ACCELERATION = 2.0* - Dans les machines à cinématique non triviale, l'accélération utilisée pour *teleop* jog (espace cartésien), en unités machine par seconde par seconde.
– *MAX_VELOCITY = 5.0* - Vitesse maximale de déplacement pour les axes, exprimée en unités machine par seconde. La valeur indiquée est égale à 300 unités par minute.
– *MAX_ACCELERATION = 20.0* - Accélération maximale pour les axes, exprimée en unités machine par seconde par seconde.

– *POSITION_FILE = position.txt* - Si réglée à une valeur non vide, les positions des axes (joins) sont enregistrées dans ce fichier. Cela permet donc de redémarrer avec les mêmes coordonnées que lors de l'arrêt, ce qui suppose, que hors puissance, la machine ne fera aucun mouvement pendant tout son arrêt. C'est utile pour les petites machines sans contact d'origine machine. Si vide, les positions ne seront pas enregistrées et commenceront à 0 à chaque fois que LinuxCNC démarrera.
– *NO_FORCE_HOMING = 1* - LinuxCNC oblige implicitement l'utilisateur à référencer la machine par une prise d'origine machine avant de pouvoir lancer un programme ou exécuter une commande dans le MDI, seuls les mouvements de Jog sont autorisés avant les prises d'origines. Mettre NO_FORCE_HOMING = 1 permet à l'opérateur averti de s'affranchir de cette restriction de sécurité lors de la phase de mise au point de la machine.

AVERTISSEMENT
NO_FORCE_HOMING mise à 1 permettra à la machine de franchir les limites logicielles pendant les mouvements ce qui n'est pas souhaitable pour un fonctionnement normal!

3.5.10 Sections [AXIS_n]

Les sections [AXIS_0], [AXIS_1], etc. contiennent les paramètres généraux des composants individuels du module de contrôle. La numérotation des sections axis commence à 0 et augmente jusqu'au nombre d'axes spécifiés dans la variable [TRAJ] AXES, moins 1.

Généralement (mais pas toujours):
– AXIS_0 = X
– AXIS_1 = Y
– AXIS_2 = Z
– AXIS_3 = A
– AXIS_4 = B
– AXIS_5 = C
– AXIS_6 = U
– AXIS_7 = V
– AXIS_8 = W
  – *TYPE = LINEAR* - Type des axes, soit LINEAR, soit ANGULAR.
  – *WRAPPED_ROTARY = 1* - Lorsque ce paramètre est réglé à 1 pour un axe angulaire l'axe se déplace de 0 à 359.999 degrés. Les nombres positifs déplacent l'axe dans le sens positif et les nombres négatifs dans le sens négatif.
  – *LOCKING_INDEXER = 1* - Quand ce paramètre est mis à 1, un mouvement en G0 sur cet axe va produire un signal de déblocage sur la pin *axis.N.unlock*, puis attendre le signal *axis.N.is-unlocked* de cet axe pour déplacer l'axe à la vitesse rapide prévue pour cet axe. Après ce mouvement, le signal *axis.N.unlock* retombera à false et les mouvements attendront que *axis.N.is-unlocked* redevienne false. Le mouvement des autres axes n'est pas autorisé lors du mouvement d'un axe rotatif à verrou.
  – *UNITS = inch* - Ce réglage écrase celui des variables [TRAJ] UNITS si il est spécifié. (ex: [TRAJ]LINEAR_UNITS si le TYPE de cet axe est LINEAR, [TRAJ]ANGULAR_UNITS si le TYPE de cet axe est ANGULAR)
  – *MAX_VELOCITY = 1.2* - Vitesse maximum pour cet axe en unités machine par seconde.
  – *MAX_ACCELERATION = 20.0* - Accélération maximum pour cet axe en unités machine par seconde au carré.
  – *BACKLASH = 0.000* - Valeur de compensation du jeu en unités machine. Peut être utilisée pour atténuer de petites déficiences du matériel utilisé pour piloter cet axe. Si un backlash est ajouté à un axe et que des moteurs pas à pas sont utilisés, la valeur de STEPGEN_MAXACCEL doit être 1.5 à 2 fois plus grande que celle de MAX_ACCELERATION pour cet axe.
  – *COMP_FILE = file.extension* - Fichier dans lequel est enregistrée une structure de compensation spécifique à cet axe. Le fichier peut être nommé *xscrew.comp*, par exemple, pour l'axe X. Les noms de fichiers sont sensibles à la casse et peuvent contenir des lettres et/ou des chiffres. Les valeurs sont des triplets par ligne séparés par un espace. La première valeur est nominale (où elle devrait l'être). Les deuxième et troisième valeurs dépendront du réglage de COMP_FILE_TYPE. Actuellement la limite de LinuxCNC est de 256 triplets par axe. Si COMP_FILE est spécifié, BACKLASH est ignoré. Les valeurs sont en unités machine.
  – *COMP_FILE_TYPE = 0 ou 1* -
    – *Si 0:* Les deuxième et troisième valeurs spécifient la position en avant (de combien l'axe est en avance) et la position en arrière (de combien l'axe est en retard), positions qui correspondent à la position nominale.

  - *Si 1:* Les deuxième et troisième valeurs spécifient l'ajustement avant (à quelle distance de la valeur nominale lors d'un déplacement vers l'avant) et l'ajustement arrière (à quelle distance de la valeur nominale lors d'un déplacement vers l'arrière), positions qui correspondent à la position nominale.

Exemple de triplet avec COMP_FILE_TYPE = 0: 1.00 1.01 0.99
Exemple de triplet avec COMP_FILE_TYPE = 1: 1.00 0.01 -0.01

- *MIN_LIMIT = -1000* - Limite minimale des mouvements de cet axe (limite logicielle), en unités machine. Quand cette limite tend à être dépassée, le contrôleur arrête le mouvement.
- *MAX_LIMIT = 1000* - Limite maximale des mouvements de cet axe (limite logicielle), en unités machine. Quand cette limite tend à être dépassée, le contrôleur arrête le mouvement.
- *MIN_FERROR = 0.010* - Valeur indiquant, en unités machine, de combien le mobile peut dévier à très petite vitesse de la position commandée. Si MIN_FERROR est plus petit que FERROR, les deux produisent une rampe de points de dérive. Vous pouvez imaginer un graphe sur lequel une dimension représente la vitesse et l'autre, l'erreur tolérée. Quand la vitesse augmente, la quantité d'erreurs de suivi augmente également et tend vers la valeur FERROR.
- *FERROR = 1.0* - FERROR est le maximum d'erreur de suivi tolérable, en unités machine. Si la différence entre la position commandée et la position retournée excède cette valeur, le contrôleur désactive les calculs des servomoteurs, positionne toutes les sorties à 0.0 et coupe les amplis des moteurs. Si MIN_FERROR est présent dans le fichier .ini, une vitesse proportionnelle aux erreurs de suivi est utilisée. Ici, le maximum d'erreur de suivi est proportionnel à la vitesse, quand FERROR est appliqué à la vitesse rapide définie dans [TRAJ]MAX_VELOCITY et proportionnel aux erreurs de suivi pour les petites vitesses. L'erreur maximale admissible sera toujours supérieure à MIN_FERROR. Cela permet d'éviter que de petites erreurs de suivi sur les axes stationnaires arrêtent les mouvements de manière impromptue. Des petites erreurs de suivi seront toujours présentes à cause des vibrations, etc. La polarité des valeurs de suivi détermine comment les entrées sont interprétées et comment les résultats sont appliqués aux sorties. Elles peuvent généralement être réglées par tâtonnement car il n'y a que deux possibilités. L'utilitaire de calibration peut être utilisé pour les ajuster interactivement et vérifier les résultats, de sorte que les valeurs puissent être mises dans le fichier INI avec un minimum de difficultés. Cet utilitaire est accessible dans Axis depuis le menu *Machine* puis *Calibration* et dans TkLinuxCNC depuis le menu *Réglages* puis *Calibration*.

3.5.11 Section [HOMING]

Les paramètres suivants sont relatifs aux prises d'origine, pour plus d'informations, lire le chapitre sur la POM.

- *HOME = 0.0* - La position à laquelle le mobile ira à la fin de la séquence de prise d'origine.
- *HOME_OFFSET = 0.0* - Position du contact d'origine machine de l'axe ou de l'impulsion d'index, en unités machine. Lorsque le point d'origine est détecté pendant le processus de prise d'origine, c'est cette position qui est assignée à ce point. Dans le cas du partage de capteur entre l'origine et les limites d'axe et de l'utilisation d'une séquence de prise d'origine qui laisse le capteur dans l'état activé, la valeur de HOME_OFFSET peut être utilisée pour définir une position du capteur différente du 0 utilisé alors pour l'origine.
- *HOME_SEARCH_VEL = 0.0* - Vitesse du mouvement initial de prise d'origine, en unités machine par seconde. Une valeur de zéro suppose que la position courante est l'origine machine. Si la machine n'a pas de contact d'origine, laisser cette valeur à zéro.
- *HOME_LATCH_VEL = 0.0* - Vitesse du mouvement de dégagement du contact d'origine, en unités machine par seconde.
- *HOME_FINAL_VEL = 0.0* - Vitesse du mouvement final entre le contact d'origine et la position d'origine, en unités machine par seconde. Si cette variable est laissée à 0 ou absente, la vitesse de déplacement rapide est utilisée. Doit avoir une valeur positive.
- *HOME_USE_INDEX = NO* - Si l'encodeur utilisé pour cet axe fournit une impulsion d'index et qu'elle est gérée par la carte contrôleur, il est possible de mettre sur Yes. Quand il est sur yes, il aura une incidence sur le type de séquence de prise d'origine utilisée.
- *HOME_IGNORE_LIMITS = NO* - Si la machine utilise un seul et même contact comme limite d'axe et origine machine de l'axe. Cette variable devra alors être positionnée sur yes. Dans ce cas le contact de limite de cet axe est ignoré pendant la séquence de prise d'origines. Il est nécessaire de configurer la séquence pour qu'à la fin du mouvement le capteur ne reste pas dans l'état activé qui aboutirait finalement à un message d'erreur du capteur de limite.
- *HOME_IS_SHARED = <n>* - Si l'entrée du contact d'origine est partagée par plusieurs axes, mettre <n> à 0 pour permettre la POM même si un des contacts partagés est déjà attaqué. Le mettre à 1 pour interdire la prise d'origine dans ce cas.
- *HOME_SEQUENCE = <n>* - Utilisé pour définir l'ordre dans lequel les axes se succéderont lors d'une séquence de *POM générale*. **<n>** commence à 0, aucun numéro ne peut être sauté. Si cette variable est absente ou à -1, la POM de l'axe ne pourra pas être exécutée par la commande *POM générale*. La POM de plusieurs axes peut se dérouler simultanément.
- *VOLATILE_HOME = 0* - Lorsqu'il est activé (mis à 1), l'origine machine de cette articulation sera effacée si la machine est en marche et que l'arrêt d'urgence est activé. Ceci est utile si la machine possède des contacts d'origine mais n'a pas de retour de position comme une machine à moteur pas à pas de type pas/direction.

3.5.12 Variables relatives aux servomoteurs

Les éléments suivants sont pour les systèmes à servomoteurs et à pseudos servomoteurs. Cette description suppose que les unités en sortie du composant PID sont des Volts.

– *DEADBAND = 0.000015* - (dans HAL) Quelle distance est assez proche de la consigne pour considérer le moteur en position, en unités machine. Cette variable est fréquemment réglée pour une distance équivalente à 1, 1.5, 2, ou 3 impulsions de comptage du codeur, mais cela n'a rien d'une règle stricte. Un réglage lâche (large) permet de moins solliciter le servo au détriment de la précision. Un réglage serré (petit) permettra d'atteindre une grande précision mais le servo sera plus sollicité. Est-ce vraiment plus précis si c'est plus incertain ? En règle générale, il est préférable d'éviter le plus possible de solliciter le servo, si c'est possible.

Ayez la prudence de ne pas chercher à aller en dessous d'une impulsion de codeur, sinon vous enverrez votre servo quelque part où il ne sera pas heureux ! Cela peut arriver entre réglage lent et réglage nerveux et même un réglage impropre peut provoquer des couinements, des grincements dus aux oscillations provoquées par ce mauvais réglage. Il est préférable de perdre une ou deux impulsions au début des réglages, au moins jusqu'à avoir bien dégrossi les réglages.

Exemple de calcul en unités machine par top de codeur à utiliser pour décider de la valeur de DEADBAND (bande morte):

X pouces / top de codeur =1 tour / 1000 top de codeur * 1 top de codeur / 4 top en quadrat
*** 0.2 pouce / tour =0.200 pouce / 4000 top de codeur = 0.000050 pouce / top de codeur.**

– *BIAS = 0.000* - (dans HAL) (Parfois appelé *offset*) il est utilisé par hm2-servo et quelques autres. Le Bias est une valeur constante qui est ajoutée sur la sortie. Dans la plupart des cas, elle peut rester à zéro. Toutefois, il peut être intéressant pour compenser un décalage de l'ampli du servo, ou équilibrer le poids d'un objet se déplaçant verticalement. Le bias est mis à zéro quand la boucle PID est désactivée, comme tous les autres composants de la sortie.
– *P = 50* - (hal) La composante Proportionnelle du gain de l'ampli moteur de cet axe. Cette valeur multiplie l'erreur entre la position commandée et la position actuelle en unités machine, elle entre dans le calcul de la tension appliquée à l'ampli moteur. Les unités du gain **P** sont des Volts sur des unités machine, par exemple: **Volt/mm** si l'unité machine est le millimètre.
– *I = 0* - (hal) La composante Intégrale du gain de l'ampli moteur de cet axe. Cette valeur multiplie l'erreur cumulative entre la position commandée et la position actuelle en unités machine, elle entre dans le calcul de la tension appliquée à l'ampli moteur. Les unités du gain **I** sont des Volts sur des unités machine par seconde, exemple: **Volt/mm*s** si l'unité machine est le millimètre.
– *D = 0* - (hal) La composante Dérivée du gain de l'ampli moteur de cet axe. Cette valeur multiplie la différence entre l'erreur courante et les précédentes, elle entre dans le calcul de la tension appliquée à l'ampli moteur. Les unités du gain **D** sont des Volts sur des unités machine sur des secondes, exemple: **Volt/(mm/s)** si l'unité machine est le millimètre.
– *FF0 = 0* - (hal) Gain à priori (retour vitesse) d'ordre 0. Cette valeur est multipliée par la position commandée, elle entre dans le calcul de la tension appliquée à l'ampli moteur. Les unités du gain FF0 sont des Volts sur des unités machine, exemple: **Volt/mm** si l'unité machine est le millimètre.
– *FF1 = 0* - (hal) Gain à priori (retour vitesse) de premier ordre. Cette valeur est multipliée par l'écart de la position commandée par seconde, elle entre dans le calcul de la tension appliquée à l'ampli moteur. Les unités du gain FF1 sont des Volts sur des unités machine par seconde, exemple: **Volt/(mm/s)** si l'unité machine est le millimètre.
– *FF2 = 0* - (hal) Gain à priori (retour vitesse) de second ordre. Cette valeur est multipliée par l'écart de la position commandée par seconde au carré, elle entre dans le calcul de la tension appliquée à l'ampli moteur. Les unités du gain FF2 sont des Volts sur des unités machine par des secondes au carré, exemple: **Volt/mm/s^2** si l'unité machine est le millimètre.
– *OUTPUT_SCALE = 1.000* -
– *OUTPUT_OFFSET = 0.000* - (hal) Ces deux valeurs sont les facteurs d'échelle et offset pour la sortie de l'axe à l'amplificateurs moteur. La seconde valeur (offset) est soustraite de la valeur de sortie calculée (en Volts) puis divisée par la première valeur (facteur d'échelle), avant d'être écrite dans le convertisseur D/A. Les unités du facteur d'échelle sont des Volts réels par Volts en sortie de DAC. Les unités de la valeur d'offset sont en Volts. Ces valeurs peuvent être utilisées pour linéariser un DAC. Plus précisément, quand les sorties sont écrites, LinuxCNC converti d'abord les unités quasi-SI des sorties concernées en valeurs brutes, exemple: Volts pour un amplificateur DAC. Cette mise à l'échelle ressemble à cela:
  raw = output-offset/scale la valeur d'échelle peut être obtenue par analyse des unités, exemple: les unités sont [unités SI en sortie]/[unités de l'actuateur]. Par exemple, sur une machine sur laquelle une tension de consigne de l'ampli de 1 Volt donne une vitesse de 250 mm/s :
  amplifier [volts] = (output[mm/s] - offset[mm/s]) / 250mm/(s/Volt)
  Notez que les unités d'offset sont en unités machine, exemple: mm/s et qu'elles sont déjà soustraites depuis la sonde de lecture. La valeur de cet offset est obtenue en prenant la valeur de votre sortie qui donne 0,0 sur la sortie de l'actuateur. Si le DAC est linéarisé, cet offset est normalement de 0.0.

L'échelle et l'offset peuvent être utilisés pour linéariser les DAC, d'où des valeurs qui reflètent les effets combinés du gain de l'ampli, de la non linéarité du DAC, des unités du DAC, etc. Pour ce faire, suivez cette procédure:

– Construire un tableau de calibrage pour la sortie, piloter le DAC avec la tension souhaitée et mesurer le résultat. Voir le tableau ci-dessous pour un exemple de mesures de tension.
– Par la méthode des moindres carrés, obtenir les coefficients **a**,**b** tels que: **`mesure = a*raw+b`**
– Noter que nous voulons des sorties brutes de sorte que nos résultats mesurés soient identiques à la sortie commandée. Ce qui signifie:
– **`cmd = a*raw+b`**
– **`raw = (cmd-b)/a`**
– En conséquence, les coefficients **a** et **b** d'ajustement linéaire peuvent être directement utilisés comme valeurs d'échelle et d'offset pour le contrôleur.

Brutes (Raw)	Mesurées
-10	-9.93
-9	-8.83
0	-0.03
1	0.96
9	9.87
10	10.87

– *MAX_OUTPUT = 10* - (hal) La valeur maximale pour la sortie de la compensation PID pouvant être envoyée sur l'ampli moteur, en Volts. La valeur calculée de la sortie sera fixée à cette valeur limite. La limite est appliquée avant la mise à l'échelle de la sortie en unités brutes. La valeur est appliquée de manière symétrique aux deux côtés, le positif et le négatif.
– *INPUT_SCALE = 20000* - (hal) Spécifie le nombre d'impulsions qui correspond à un mouvement de une unité machine telle que fixée dans la section TRAJ. Pour un axe linéaire, une unité machine sera égale à la valeur de LINEAR_UNITS. Pour un axe angulaire, une unité machine sera égale à la valeur de ANGULAR_UNITS. Un second chiffre, si spécifié, sera ignoré. Par exemple, sur un codeur de 2000 impulsions par tour, un réducteur de 10 tours/pouce et des unités demandées en pouces, nous avons:
`INPUT_SCALE = 2000 top/tour * 10 tour/pouce = 20000 top/pouce`

3.5.13 Variables relatives aux moteurs pas à pas

– *SCALE = 4000* - (hal) Spécifie le nombre d'impulsions qui correspond à un mouvement d'une unité machine comme indiqué dans la section [TRAJ]. Pour les systèmes à moteurs pas à pas, c'est le nombre d'impulsions de pas nécessaires pour avancer d'une unité machine. Pour un axe linéaire, une unité machine sera égale à la valeur de LINEAR_UNITS. Pour un axe angulaire, une unité machine sera égale à la valeur de ANGULAR_UNITS. Pour les systèmes à servomoteurs, c'est le nombre d'impulsions de retour signifiant que le mobile a avancé d'une unité machine. Un second nombre, si spécifié, sera ignoré. Par exemple, un pas moteur de 1.8 degré, en mode demi-pas, avec une réduction de 10 tours/pouce et des unités souhaitées en pouces, nous avons:
`scale = 2 pas/1.8 degrés * 360 degrés/tour * 10 tour/pouce = 4000 pas/pouce`

(D'anciens fichiers .ini et .hal utilisaient INPUT_SCALE pour cette valeur.)
– *STEPGEN_MAXACCEL = 21.0* - (hal) Limite d'accélération pour le générateur de pas. Elle doit être 1% à 10% supérieure à celle de l'axe MAX_ACCELERATION. Cette valeur améliore les réglages de la *boucle de position* de stepgen. Si une correction de jeu a été appliquée sur un axe, alors STEPGEN_MAXACCEL doit être 1,5 à 2 fois plus grande que MAX_ACCELERATION.
– *STEPGEN_MAXVEL = 1.4* - (hal) Les anciens fichiers de configuration avaient également une limite de vitesse du générateur de pas. Si spécifiée, elle doit aussi être 1% à 10% supérieure à celle de l'axe MAX_VELOCITY. Des tests ultérieurs ont montré que l'utilisation de STEPGEN_MAXVEL n'améliore pas le réglage de la boucle de position de stepgen.

3.5.14 Section [EMCIO]

– *CYCLE_TIME = 0.100* - La période en secondes, à laquelle EMCIO va tourner. La mettre à 0.0 ou à une valeur négative fera que EMCIO tournera en permanence. Il est préférable de ne pas modifier cette valeur.
– *TOOL_TABLE = tool.tbl* - Ce fichier contient les informations des outils, décrites dans le Manuel de l'utilisateur.
– *TOOL_CHANGE_POSITION = 0 0 2* - Quand trois digits sont utilisés, spécifie la position XYZ ou le mobile sera déplacé pour le changement d'outil. Si six digits sont utilisés, spécifie l'emplacement ou sera envoyé le mobile pour réaliser le changement d'outil sur une machine de type XYZABC et de même, sur une machine de type XYZABCUVW lorsque 9 digits

sont utilisés. Les variables relatives à la position du changement d'outil peuvent être combinées, par exemple; en combinant TOOL_CHANGE_POSITION avec TOOL_CHANGE_QUILL_UP il est possible de déplacer d'abord Z puis X et Y.

– *TOOL_CHANGE_WITH_SPINDLE_ON = 1* - Avec cette valeur à 1, la broche reste en marche pendant le changement d'outil. Particulièrement utile sur les tours.
– *TOOL_CHANGE_QUILL_UP = 1* - Avec cette valeur à 1, l'axe Z sera déplacé sur son origine machine avant le changement d'outil. C'est l'équivalent d'un G0 G53 Z0.
– *TOOL_CHANGE_AT_G30 = 1* - Avec cette valeur à 1, le mobile sera envoyé sur un point de référence prédéfini par G30 dans les paramètres 5181-5186. Pour plus de détails sur les paramètres de G30, voir le chapitre relatif au G-code dans le Manuel de l'utilisateur.
– *RANDOM_TOOLCHANGER = 1* - C'est pour des machines qui ne peuvent pas placer l'outil dans la poche il vient. Par exemple, les machines qui change l'outil dans la poche active avec l'outil dans la broche.

Chapitre 4

Prise d'origine

4.1 La prise d'origine

La prise d'origine semble assez simple, il suffit de déplacer chaque axe à un emplacement connu et de positionner l'ensemble des variables internes de LinuxCNC en conséquence. Toutefois, les machines étant différentes les unes des autres, la prise d'origine est maintenant devenue assez complexe.

4.2 Séquences de prise d'origine

Il existe quatre séquences de prise d'origine possibles. Elles sont définies par le signe des variables SEARCH_VEL et LATCH_VEL ainsi que les paramètres de configuration associés, la figure suivante donne le détail de ces séquences.

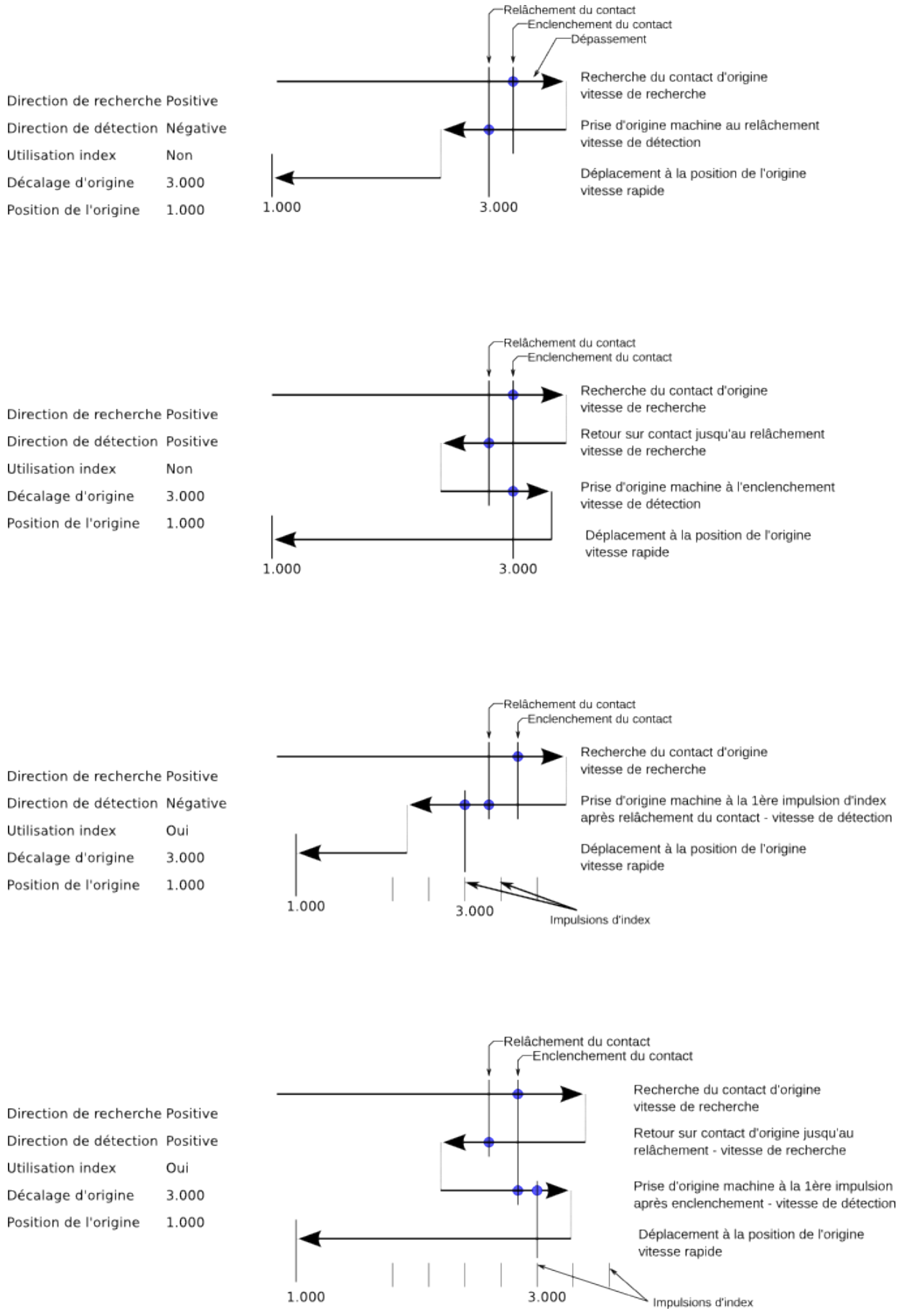


FIGURE 4.1 – Les séquences de POM possibles

1. Comme on le voit sur la figure, les deux conditions de base sont les suivantes:
   a. La direction de recherche (SEARCH_VEL) et la direction de détection (LATCH_VEL) sont de même signe.
   b. La direction de recherche (SEARCH_VEL) et la direction de détection (LATCH_VEL) sont de signe opposé.

4.3 Configuration

Le tableau suivant détermine exactement comment se déroule la séquence de prise d'origines définie dans la section [AXIS] du fichier ini.

TABLE 4.1: Combinaisons des variables de la POM

Type de POM	SEARCH_VEL	LATCH_VEL	USE_INDEX
Immediate	0	0	NON
Index-seul	0	nonzero	OUI
Contact-seul	nonzero	nonzero	NO
Contact et Index	nonzero	nonzero	OUI

Note
Toute autre combinaison produira une erreur.

4.3.1 Vitesse de recherche (HOME_SEARCH_VEL)

Vitesse de la phase initiale de prise d'origine, pendant la recherche du contact d'origine machine. Une valeur différente de zéro indique à LinuxCNC la présence d'un contact d'origine machine. LinuxCNC va alors commencer par vérifier si ce contact est déjà attaqué. Si oui, il le dégagera à la vitesse établie par *HOME_SEARCH_VEL*, la direction du dégagement sera de signe opposé à celui de *HOME_SEARCH_VEL*. Puis, il va revenir vers le contact en se déplaçant dans la direction spécifiée par le signe de *HOME_SEARCH_VEL* et à la vitesse déterminée par sa valeur absolue. Quand le contact d'origine machine est détecté, le mobile s'arrête aussi vite que possible, il y aura cependant toujours un certain dépassement dû à l'inertie et dépendant de la vitesse. Si celle-ci est trop élevée, le mobile peut dépasser suffisamment le contact pour aller attaquer un fin de course de limite d'axe, voir même aller se crasher dans une butée mécanique. À l'opposé, si *HOME_SEARCH_VEL* est trop basse, la prise d'origine peut durer très longtemps.

Une valeur égale à zéro indique qu'il n'y a pas de contact d'origine machine, dans ce cas, les phases de recherche de ce contact seront occultées. La valeur par défaut est zéro.

4.3.2 Vitesse de détection (HOME_LATCH_VEL)

Spécifie la vitesse et la direction utilisée par le mobile pendant la dernière phase de la prise d'origine, c'est la recherche précise du contact d'origine machine, si il existe et de l'emplacement de l'impulsion d'index, si elle est présente. Cette vitesse est plus lente que celle de la phase de recherche initiale, afin d'améliorer la précision. Si *HOME_SEARCH_VEL* et *HOME_LATCH_VEL* sont de mêmes signes, la phase de recherche précise s'effectuera dans le même sens que la phase de recherche initiale. Dans ce cas, le mobile dégagera d'abord le contact en sens inverse avant de revenir vers lui à la vitesse définie ici. L'acquisition de l'origine machine se fera sur la première impulsion de changement d'état du contact. Si *HOME_SEARCH_VEL* et *HOME_LATCH_VEL* sont de signes opposés, la phase de recherche précise s'effectuera dans le sens opposé à celui de la recherche initiale. Dans ce cas, LinuxCNC dégagera le contact à la vitesse définie ici. L'acquisition de l'origine machine se fera sur la première impulsion de changement d'état du contact lors de son dégagement. Si *HOME_SEARCH_VEL* est à zéro, signifiant qu'il n'y a pas de contact et que *HOME_LATCH_VEL* et différente de zéro, le mobile continuera jusqu'à la prochaine impulsion d'index, l'acquisition de l'origine machine se fera à cet position. Si *HOME_SEARCH_VEL* est différent de zéro et que *HOME_LATCH_VEL* est égale à zéro, c'est une cause d'erreur, l'opération de prise d'origine échouera. La valeur par défaut est zéro.

4.3.3 HOME_IGNORE_LIMITS

Peut contenir les valeurs YES ou NO. Cette variable détermine si LinuxCNC doit ignorer les fins de course de limites d'axe. Certaines machines n'utilisent pas un contact d'origine séparé, à la place, elles utilisent un des interrupteurs de fin de course comme contact d'origine. Dans ce cas, LinuxCNC doit ignorer l'activation de cette limite de course pendant la séquence de prise d'origine. La valeur par défaut de ce paramètre est NO.

4.3.4 HOME_USE_INDEX

Spécifie si une impulsion d'index doit être prise en compte (cas de règles de mesure ou de codeurs de positions). Si cette variable est vraie (HOME_USE_INDEX = YES), LinuxCNC fera l'acquisition de l'origine machine sur le premier front de l'impulsion d'index. Si elle est fausse (=NO), LinuxCNC fera l'acquisition de l'origine sur le premier front produit par le contact d'origine (dépendra des signes de *HOME_SEARCH_VEL* et *HOME_LATCH_VEL*). La valeur par défaut est NO.

4.3.5 HOME_OFFSET

Contient l'emplacement du point d'origine ou de l'impulsion d'index, en coordonnées relatives. Il peut aussi être traité comme le décalage entre le point d'origine machine et le zéro de l'axe. A la détection du point d'origine ou de l'impulsion d'origine, LinuxCNC ajuste les coordonnées de l'axe à la valeur de *HOME_OFFSET*. La valeur par défaut est zéro.

4.3.6 Position de l'origine (HOME)

C'est la position sur laquelle ira le mobile à la fin de la séquence de prise d'origine machine. Après avoir détecté le contact d'origine, avoir ajusté les coordonnées de ce point à la valeur de *HOME_OFFSET*, le mobile va se déplacer sur la valeur de *HOME*, c'est le point final de la séquence de prise d'origine. La valeur par défaut est zéro. Notez que même si ce paramètre est égal à la valeur de *HOME_OFFSET*, le mobile dépassera très légérement la position du point d'aquisition de l'origine machine avant de s'arrêter. Donc il y aura toujours un petit mouvement à ce moment là (sauf bien sûr si *HOME_SEARCH_VEL* est à zéro, et que toute la séquence de POM a été sautée). Ce mouvement final s'effectue en vitesse de déplacement rapide. Puisque l'axe est maintenant référencé, il n'y a plus de risque pour la machine, un mouvement rapide est donc la façon la plus rapide de finir la séquence de prise d'origine.

4.3.7 HOME_IS_SHARED

Si cet axe n'a pas un contact d'origine séparé des autres, mais plusieurs contacts câblés sur la même broche d'entrée, mettre cette valeur à 1 pour éviter de commencer la prise d'origine si un de ces contacts partagés est déjà activé. Mettez cette valeur à 0 pour permettre la prise d'origine même si un contact est déjà attaqué.

4.3.8 HOME_SEQUENCE

Utilisé pour définir l'ordre des séquences *HOME_ALL* de prise d'origine des différents axes (exemple: la POM de l'axe X ne pourra se faire qu'après celle de Z). La POM d'un axe ne pourra se faire qu'après tous les autres en ayant la valeur la plus petite de *HOME_SEQUENCE* et après qu'ils soient déjà tous à *HOME_OFFSET*. Si deux axes ont la même valeur de *HOME_SEQUENCE*, leurs POM s'effectueront simultanément. Si *HOME_SEQUENCE* est égale à -1 ou n'est pas spécifiée, l'axe ne sera pas compris dans la séquence *HOME_ALL*. Les valeurs de *HOME_SEQUENCE* débutent à 0, il ne peut pas y avoir de valeur inutilisée.

4.3.9 VOLATILE_HOME

Si ce paramètre est vrai, l'origine machine de cet axe sera effacée chaque fois que la machine sera mise à l'arrêt. Cette variable est appropriée pour les axes ne maintenant pas la position si le moteur est désactivé (gravité de la broche par exemple). Certains moteurs pas à pas, en particulier fonctionnant en micropas, peuvent se comporter de la sorte.

4.3.10 LOCKING_INDEXER

Si cet axe comporte un verrouillage d'indexeur rotatif, celui-ci sera déverrouillé avant le début de la séquence de prise d'origine, et verrouillé à la fin.

Chapitre 5

Tours

5.1 Plan par défaut

Quand l'interpréteur de LinuxCNC à été créé, il à été écrit pour les fraiseuses. C'est pourquoi le plan par défaut est le plan XY (G17). Sur un tour standard on utilise seulement les axes du plan XZ (G18). Pour changer le plan par défaut d'un tour, mettez la ligne suivante dans la section RS274NGC du fichier ini.

```
RS274NGC_STARTUP_CODE = G18
```

La ligne de commande ci-dessus peut être remplacées par des G codes placés dans le préambule du programme G-code.

5.2 Réglages INI

Les réglages du fichier .ini suivants sont nécessaires dans Axis en mode tour, en remplacement ou ajout au fichier .ini normal.

```
[DISPLAY]
DISPLAY = axis
LATHE = 1
[TRAJ]
AXES = 3
COORDINATES = X Z
[AXIS_0]
...
[AXIS_2]
...
```

Chapitre 6

Fichiers TCL pour HAL

Le langage de halcmd excelle pour spécifier des composants et des connexions mais il n'offre aucune possibilité de calcul. Par conséquent, les fichiers ini sont limités en clarté et n'ont pas la concision qu'ils pourraient avoir avec un langage de haut niveau.

Haltcl fourni un moyen facile d'utiliser les scripts tcl et leurs fonctions pour les calculs, les boucles, les branchements, les procédures, etc dans les fichiers ini. Pour utiliser cette fonctionnalité, il est nécessaire d'utiliser le langage TCL ainsi que l'extension .tcl pour les fichiers de HAL.

L'extension .tcl est comprise par le script principal (linuxcnc) qui traite les fichiers ini. Les fichiers haltcl sont identifiés dans la section [HAL] des fichiers ini (comme les fichiers .hal).

Exemple

```
[HAL]
HALFILE = fichier_conventionnel.hal
HALFILE = fichier_tcl.tcl
```

Avec quelques précautions appropriées, les fichiers .hal et .tcl peuvent être mélangés.

6.1 Compatibilité

Le langage utilisé dans les fichiers .tcl a une syntaxe simple qui est un sous-ensemble du puissant et polyvalent langage de script Tcl.

6.2 Commandes haltcl

Les fichiers haltcl utilisent le langage de script Tcl, auquel s'ajoutent les commandes spécifiques de la couche d'abstraction matériel de LinuxCNC (HAL). Les commandes spécifiques sont les suivantes:

```
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
```

Il existe deux cas particuliers, les commandes *gets* et *list* en raison de conflits avec les commandes internes de Tcl. Pour haltcl, ces commandes doivent être précédées du mot clé *hal*, comme ci-dessous:

```
halcmd   haltcl
------   ------
gets     hal gets
list     hal list
```

6.3 Variables du fichier ini et haltcl

Les variables du fichier ini sont accessibles par halcmd comme par haltcl mais avec une syntaxe différente.

Les fichiers ini de LinuxCNC utilisent des spécificateurs *SECTION* et *ITEM* pour identifier les items de configuration:

```
[SECTION_A]
ITEM1 = value_1
ITEM2 = value_2
...
[SECTION_B]
...
```

Les valeurs du fichier ini sont accessibles par substitution de texte dans les fichiers .hal en utilisant la forme:

```
[SECTION]ITEM
```

Les mêmes valeurs de fichiers ini sont accessibles dans les fichiers .tcl sous la forme de tableau de variables globales.

```
$::SECTION(ITEM)
```

Par exemple, un item de fichier ini comme:

```
[AXIS_0]
MAX_VELOCITY = 4
```

est exprimé comme [AXIS_0]MAX_VELOCITY dans les fichiers .hal pour halcmd et comme $::AXIS_0(MAX_VELOCITY) dans les fichiers .tcl pour haltcl.

6.4 Conversion des fichiers .hal en fichiers .tcl

Les fichiers .hal existants peuvent être convertis manuellement en fichiers .tcl en les éditant pour adapter les différences mentionnées précédemment. Ce processus peut être automatisé à l'aide de scripts de conversion procédant à ces substitutions:

```
[SECTION]ITEM ---> $::SECTION(ITEM)
gets          ---> hal gets
list          ---> hal list
```

6.5 Notes à propos de haltcl

Dans haltcl, la valeur de l'argument pour les commandes *sets* et *setp* est implicitement traitée comme une expression dans le langage Tcl.

exemple

```
# set gain to convert deg/sec to units/min for AXIS_0 radius
setp scale.0.gain 6.28/360.0*$::AXIS_0(radius)*60.0
```

Les espaces blancs ne sont pas autorisés dans les expressions, utiliser des guillemets doubles pour s'en affranchir:

```
setp scale.0.gain "6.28 / 360.0 * $::AXIS_0(radius) * 60.0"
```

Dans d'autres contextes, tels que *loadrt*, il est nécessaire d'utiliser explicitement la commande Tcl expr ([expr {}]) pour des expressions de calcul.

exemple

```
loadrt motion base_period=[expr {500000000/$::TRAJ(MAX_PULSE_RATE)}]
```

6.6 Exemples pour haltcl

Prenons la question de la marge haute de stepgen (*stepgen headroom*). Le générateur de pas logiciel stepgen fonctionne mieux avec une contrainte d'accélération légèrement supérieure à celle du planificateur de mouvement. Ainsi, lorsqu'on utilise des fichiers halcmd, on force une valeur calculée manuellement dans le fichier ini.

```
[AXIS_0]
MAXACCEL = 10.0
STEPGEN_MAXACCEL = 10.5
```

Avec haltcl, il est possible d'utiliser des commandes Tcl pour effectuer le calcul et éliminer totalement l'item STEPGEN_MAXACCEL du fichier ini.

```
setp stepgen.0.maxaccel $::AXIS_0(MAXACCEL)*1.05
```

Autres caractéristiques de haltcl, les boucles et les tests. Par exemple, beaucoup de configurations utilisent les fichiers .hal *core_sim.hal* ou *core_sim9.hal*. Ceux-ci diffèrent du fait de la nécessité de connecter plus ou moins d'axes. Le code haltcl suivant devrait fonctionner pour n'importe quelle combinaison d'axes dans une machine à cinématique triviale (trivkins).

```
# éCre les signaux position, vitesse et ééacclration pour chaque axe
set ddt 0
foreach axis {X Y Z A B C U V W} axno {0 1 2 3 4 5 6 7 8} {
  # 'list pin' retourne une liste vide si la pin n'existe pas
  if {[hal list pin axis.$axno.motor-pos-cmd] == {}} {
    continue
  }
  net ${axis}pos axis.$axno.motor-pos-cmd => axis.$axno.motor-pos-fb \
                                          => ddt.$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]
```

6.7 Interactivité de haltcl

La commande halrun reconnaît les fichiers halctl. Avec l'option -T, haltcl peut être exécuté interactivement comme un interpréteur Tcl. Cette fonctionnalité est utile pour les tests et pour les applications hal autonomes.

exemple

```
$ halrun -T fichierhaltcl.tcl
```

6.8 Exemples pour haltcl fournis avec la distribution (sim)

Le répertoire *configs/sim/axis/simtcl* contient un fichier ini qui utilise un fichier .tcl pour démontrer une configuration haltcl en conjonction avec l'utilisation du processus "twopass". L'exemple montre l'utilisation des procédures Tcl, les boucles, l'utilisation des commentaires avec sortie sur le terminal.

Chapitre 7

LinuxCNC et HAL

Voir également la man page *motion(9)*.

7.1 motion (temps réel)

Ces pins et paramètres sont créés par le module temps réel *motmod*. Ce module fourni une interface vers HAL pour le planificateur de mouvements de LinuxCNC. En gros, motmod prends dans une liste de points de cheminement et génère un flux de positions respectant les limites de contrainte des articulations. Ce flux sera reçu simultanément par tous les pilotes de moteurs.

Optionnellement le nombre d'E/S numériques est fixé avec *num_dio*. Le nombre d'E/S analogiques est fixé avec *num_aio*. Le nombre par défaut est 4 de chaque.

Les noms de pin commençant par *axis* sont actuellement des valeurs d'articulations, mais les pins et les paramètres sont également appelés *axis.N*. Ils sont lus et mis à jour par la fonction *motion-controller*.

motion est chargé par la commande *motmod*. La cinématique doit être chargée avant motion.

```
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])]
```

– *base_period_nsec = 50000* - période de *Base* des tâches, en nanosecondes. C'est le *thread* le plus rapide de la machine.

Note
Sur les systèmes à base de servomoteurs, il n'y a généralement aucune raison d'avoir une valeur *base_period_nsec* inférieure à celle de *servo_period_nsec*. Sur les machines avec une génération de pas logicielle, la valeur de *base_period_nsec* détermine le nombre maximum de pas par seconde. En l'absence de la nécessité d'une grande durée de pas ou d'un grand écart entre pas, le taux maximum de pas est de un pas par *base_period_nsec*. Ainsi, la *base'period'nsec* ci-dessus donnera un taux maximum absolu de 20000 pas par seconde. 50000ns (50 μs) est une valeur assez prudente. La plus petite valeur utilisable est relative au résultat du test de latence, à la longueur de pas nécessaire et à la vitesse du processeur. Choisir une *base_period_nsec* trop basse peut entrainer l'arrivée du message "Unexpected real time delay" ou "délai temps réel inattendu", le blocage de la machine ou son redémarrage spontané.

– *servo_period_nsec = 1000000* - C'est la période de la tâche *Servo* en nanosecondes. Cette valeur doit être arrondie à un entier multiple de *base_period_nsec*. Cette période est utilisée même sur les systèmes à moteurs pas à pas.
C'est la vitesse à laquelle sont calculées les nouvelles positions des moteurs, que l'erreur de suivi est calculée, que les valeurs de sortie des PID sont rafraichies et ainsi de suite. Les valeurs par défaut conviennent pour la plupart des systèmes. C'est le taux de rafraichissement du planificateur de mouvement de bas niveau.
– *traj_period_nsec = 100000* - C'est la période, en nanosecondes, du planificateur de mouvement. Cette valeur doit être arrondie à un entier multiple de *servo_period_nsec*. Excepté pour les machines ayant une cinématique particulière (ex: hexapodes) cette valeur n'a pas de raison d'être supérieure à celle de *servo_period_nsec*.

7.1.1 Options

Si le nombre d'entrées/sorties numériques demandées est supérieur à la valeur par défaut de 4, il est possible d'en ajouter jusqu'à 64 en utilisant l'option num_dio au chargement de motmod.

Si le nombre d'entrées/sorties analogiques demandées est supérieur à la valeur par défaut de 4, il est possible d'en ajouter jusqu'à 16 en utilisant l'option num_aio au chargement de motmod.

7.1.2 Pins

- *motion.adaptive-feed* - (float, in) Quand la vitesse est placée en mode adaptatif avec *M52 P1* la vitesse commandée est multipliée par cette valeur. Cet effet est multiplicatif avec *motion.feed-hold* et la valeur du correcteur de vitesse du niveau NML.
- *motion.analog-in-00* - (float, in) Ces pins (00, 01, 02, 03 ou plus si configurées) sont contrôlées par *M66*.
- *motion.analog-out-00* - (float, out) Ces pins (00, 01, 02, 03 ou plus si configurées) sont contrôlées par *M67* ou *M68*.
- *motion.coord-error* - (bit, out) TRUE quand le mouvement est en erreur, ex: dépasser une limite logicielle.
- *motion.coord-mode* - (bit, out) TRUE quand le mouvement est en *mode coordonnées* par opposistion au *mode téléopération*.
- *motion.current-vel* - (float, out) La vitesse courante de l'outil.
- *motion.digital-in-00* - (bit, in) Ces pins (00, 01, 02, 03 ou plus si configurées) sont contrôlées par *M62* à *M65*.
- *motion.digital-out-00* - (bit, out) Ces pins (00, 01, 02, 03 ou plus si configurées) sont controlées par *M62* à *M65*.
- *motion.distance-to-go* - (float, out) Distance restante pour terminer le mouvement courant.
- *motion.enable* - (bit, in) Si ce bit devient FALSE, les mouvements s'arrêtent, la machine est placée dans l'état "machine arrêtée" et un message est affiché pour l'opérateur. En fonctionnement normal, ce bit devra être mis TRUE.
- *motion.feed-hold* - (bit, in) Quand la vitesse est placée en mode arrêt contrôlé avec *M53 P1* et que ce bit est TRUE, la vitesse est fixée à 0.
- *motion.motion-inposition* - (bit, out) TRUE si la machine est en position.
- *motion.motion-enabled* - (bit, out) TRUE quand l'état de la machine est *machine on*.
- *motion.on-soft-limit* - (bit, out) TRUE quand la machine est sur une limite logicielle.
- *motion.probe-input* - (bit, in) *G38.x* utilise la valeur de cette pin pour déterminer quand la sonde de mesure a touché. TRUE le contact de la sonde est fermé (touche), FALSE le contact de la sonde est ouvert.
- *motion.program-line* - (s32, out) La ligne en cours d'exécution pendant le déroulement du programme. Zéro si pas en marche ou entre deux lignes, pendant le changement de pas de programme.
- *motion.requested-vel* - (float, out) La vitesse courante requise en unités utilisateur par seconde selon le réglage F=n du fichier G-code. Les correcteurs de vitesse et autres ajustements ne s'appliquent pas à cette pin.
- *motion.spindle-at-speed* - (bit, in) Les mouvements passent en pause tant que cette pin est TRUE, sous les conditions suivantes: avant le premier mouvement d'avance suivant chaque démarrage de broche ou changement de vitesse; après le démarrage de tout enchainement de mouvements avec broche synchronisée; et si en mode CSS, à chaque transition avance rapide -> avance travail. Cette entrée peut être utilisée pour s'assurer que la broche a atteint sa vitesse, avant de lancer un mouvement d'usinage. Elle peut également être utilisée sur un tour travaillant en mode CSS, au passage d'un grand diamètre à un petit, pour s'assurer que la vitesse a été suffisamment réduite avant la prise de passe sur le petit diamètre et inversement, lors du passage d'un petit diamètre vers un grand, pour s'assurer que la vitesse a été suffisamment augmentée. Beaucoup de variateurs de fréquence disposent d'une sortie *vitesse atteinte*. Sinon, il est facile de générer ce signal avec le composant *near*, par comparaison entre la vitesse de broche demandée et la vitesse actuelle.
- *motion.spindle-brake* - (bit, out) TRUE quand le frein de broche doit être activé.
- *motion.spindle-forward* - (bit, in) TRUE quand la broche doit tourner en sens horaire.
- *motion.spindle-index-enable* - (bit, I/O) Pour les mouvements avec broche synchronisée, ce signal doit être raccordé à la pin *index-enable* du codeur de broche.
- *motion.spindle-on* - (bit, out) TRUE quand la broche doit tourner.
- *motion.spindle-reverse* - (bit, out) TRUE quand la broche doit tourner en sens anti-horaire.
- *motion.spindle-revs* - (float, in) Pour le bon fonctionnement des mouvements avec broche synchronisée, ce signal doit être raccordé à la broche *position* du codeur de broche. La position donnée par le codeur de broche doit être étalonnée pour que *spindle-revs* augmente de 1.0 pour chaque tour de broche dans le sens horaire (*M3*).
- *motion.spindle-speed-in* - (float, in) Donne la vitesse actuelle de rotation de la broche exprimée en tours par seconde. Elle est utilisée pour les mouvements en unités par tour (*G95*). Si le pilote du codeur de broche ne dispose pas d'une sortie *vitesse*, il est possible d'en générer une en passant la position de la broche au travers d'un composant ddt. Si la machine n'a pas de codeur de broche, il est possible d'utiliser *motion.spindle-speed-out-rps*.
- *motion.spindle-speed-out* - (float, out) Consigne de vitesse de rotation de la broche, exprimée en tours par minute. Positive pour le sens horaire (*M3*), négative pour le sens anti-horaire (*M4*).

– *motion.spindle-speed-out-abs* - (float, out) Consigne de vitesse absolue de rotation de la broche, exprimée en tours par minute. Toujours positive, quel que soit le sens de rotation.
– *motion.spindle-speed-out-rps* - (float, out) Consigne de vitesse de rotation de la broche, exprimée en tours par seconde. Positive pour le sens horaire (*M3*), négative pour le sens anti-horaire (*M4*).
– *motion.spindle-speed-out-rps-abs* - (float, out) Consigne de vitesse absolue de rotation de la broche, exprimée en tours par seconde. Toujours positive, quel que soit le sens de rotation.
– *motion.teleop-mode* - (bit, out) TRUE quand motion est en *mode téléopération*, par opposition au *mode coordonné*.
– *motion.tooloffset.x à motion.tooloffset.w* - (float, out; un par axe) montre l'offset d'outil courant. Il peut provenir de la table d'outils (*G43 actif*), ou du G-code (*G43.1 actif*)
– *motion.spindle-orient-angle* - (float,out) Orientation souhaitée par M19. Contient la valeur du paramètre R du M19 plus la valeur du paramètre [RS274NGC]ORIENT_OFFSET du fichier ini.
– *motion.spindle-orient-mode* - (s32,out) Broche en mode *recherche d'orientation* par M19. Par défaut 0.
– *motion.spindle-orient* - (out,bit) Indique le début d'un cycle d'orientation de la broche. Activé par M19. Révoqué par M3, M4 ou M5. Si *spindle-orient-fault* est différent de zéro pendant que *spindle-orient* est vrai, la commande M19 échoue avec un message d'erreur.
– *motion.spindle-is-oriented* - (in, bit) Pin d'acquittement pour *spindle-orient*. Achève le cycle d'orientation. Si *spindle-orient* est vraie quand *spindle-is-oriented* est actif, la pin *spindle-orient* est relâchée et les pins *spindle-locked* et *spindle-brake* sont activées.
– *motion.spindle-orient-fault* - (s32, in) Entrée d'erreur du cycle d'orientation. Toute valeur autre que zéro produira l'abandon du cycle d'orientation.
– *motion.spindle-lock* - (bit, out) Pin indiquant que l'orientation est atteinte et le cycle achevé. Relâchée par M3, M4 ou M5.

7.1.2.1 Utilisation des pins de HAL pour l'orientation broche avec M19

Par convention, la broche est dans un des trois modes suivants:
– mode rotation (mode par défaut).
– mode recherche d'orientation.
– mode orientation atteinte.
Quand un M19 est exécuté, la broche passe en mode *recherche d'orientation* et la pin de HAL *spindle-orient* est activée. L'orientation cible est spécifiée par les pins *spindle-orient-angle* et *spindle-orient-fwd* et pilotée par les paramètres R et P du M19.

La logique de HAL réagi à l'ordre sur la pin *spindle-orient* en déplaçant la broche dans la position souhaitée. Quand cette orientation est atteinte, la logique de HAL l'indique en activant la pin *spindle-is-oriented*.

En réponse, *motion* désactive la pin *spindle-orient* et active la pin *spindle-locked* indiquant le passage en mode *orientation atteinte*. Il active également la pin *spindle-brake*. La broche est alors en mode *orientation atteinte*.

Si, pendant que *spindle-orient* est vraie, et que *spindle-is-oriented* est fausse, la pin *spindle-orient-fault* a une valeur autre que zéro, la commande M19 est abandonnée, un message incluant le code d'erreur est affiché et la file d'attente de *motion* est vidée. La broche repasse en mode rotation.

Les commandes M3, M4 ou M5 annulent les modes *recherche d'orientation* ou *orientation atteinte*. Cet état est indiqué par la désactivation des broches *spindle-orient* et *spindle-locked*.

La pin *spindle-orient-mode* reflète le paramètre P du M19, ce qui sera interprété comme ci-dessous:
– 0: rotation, quel que soit le sens, pour petit mouvement angulaire (défaut)
– 1: rotation toujours en sens horaire (même direction qu'avec M3)
– 2: rotation toujours en sens anti-horaire (même direction qu'avec M4)
Il est possible d'utiliser le composant de HAL *orient* qui fournit une boucle de commande PID, basée sur la position du codeur de broche, *spindle-orient-angle* et sur *spindle-orient-mode*.

7.1.3 Paramètres

Beaucoup de ces paramètres servent d'aide au déboguage et sont sujets aux changements ou au retrait à tout moment.
– *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) Utilisé pour le déboguage.
– *motion.debug-bit-1* - (bit, RO) Utilisé pour le déboguage.
– *motion.debug-float-0* - (float, RO) Utilisé pour le déboguage.
– *motion.debug-float-1* - (float, RO) Utilisé pour le déboguage.
– *motion.debug-float-2* - (float, RO) Utilisé pour le déboguage.
– *motion.debug-float-3* - (float, RO) Utilisé pour le déboguage.
– *motion.debug-s32-0* - (s32, RO) Utilisé pour le déboguage.
– *motion.debug-s32-1* - (s32, RO) Utilisé pour le déboguage.
– *motion.servo.last-period* - Le nombre de cycle du processeur entre les invoquations du thread servo. Typiquement, ce nombre divisé par la vitesse du processeur donne un temps en secondes. Il peut être utilisé pour determiner si le contrôleur de mouvement en temps réel respecte ses contraintes de timing.
– *motion.servo.last-period-ns* - (float, RO)

7.1.4 Fonctions

Généralement, ces fonctions sont toutes les deux ajoutées à servo-thread dans l'ordre suivant:
– *motion-command-handler* - Processus des commandes de mouvement provenant de l'interface utilisateur.
– *motion-controller* - Lance le contrôleur de mouvement de LinuxCNC.

7.2 axis.N (temps réel)

Ces pins et paramètres sont créés par le module temps réel *motmod*. Ce sont en fait des valeurs d'articulations, mais les pins et les paramètres sont toujours appelés *axis.N*. [1] Ils sont lus et mis à jour par la fonction *motion-controller*.

7.2.1 Pins

– *axis.N.active* - TRUE quand cet axe est actif.
– *axis.N.amp-enable-out* - (bit, out) TRUE si l'ampli de cet axe doit être activé.
– *axis.N.amp-fault-in* - (bit, in) Doit être mis TRUE si une erreur externe est détectée sur l'ampli de cet axe.
– *axis.N.backlash-corr* - (float, out)
– *axis.N.backlash-filt* - (float, out)
– *axis.N.backlash-vel* - (float, out)
– *axis.N.coarse-pos-cmd* - (float, out)
– *axis.N.error* - (bit, out)
– *axis.N.f-error* - (float, out)
– *axis.N.f-error-lim* - (float, out)
– *axis.N.f-errored* - (bit, out)
– *axis.N.faulted* - (bit, out)
– *axis.N.free-pos-cmd* - (float, out)
– *axis.N.free-tp-enable* - (bit, out)
– *axis.N.free-vel-lim* - (float, out)
– *axis.N.home-sw-in* - (bit, in) Doit être mis TRUE si le contact d'origine de cet axe est activé.
– *axis.N.homed* - (bit, out)
– *axis.N.homing* - (bit, out) TRUE si la prise d'origine de cette axe est en cours.
– *axis.N.in-position* - TRUE si cet axe, utilisant le *free planner*, a atteint un arrêt.
– *axis.N.index-enable* - (bit, I/O) Doit être reliée à la broche *index-enable* du codeur de cet axe pour activer la prise d'origine sur l'impulsion d'index.
– *axis.N.jog-counts* - (s32, in) Connection à la broche *counts* d'un codeur externe utilisé comme manivelle.
– *axis.N.jog-enable* - (bit, in) Quand elle est TRUE (et en mode manuel), tout changement dans *jog-counts* se traduira par un mouvement. Quand elle est FALSE, *jog-counts* sera ignoré.
– *axis.N.jog-scale* - (float, in) Fixe la distance, en unités machine, du déplacement pour chaque évolution de *jog-counts*.

1. Dans une machine à *cinématique triviale*, il y a correspondance une pour une, entre les articulations et les axes. Note Du Traducteur: nous utilisons dans cette traduction le terme *axe*, dans le cas d'une cinématique non triviale il devra être remplacé par le terme *articulation* (joint).

- *axis.N.jog-vel-mode* - (bit, in) Quand elle est FALSE (par défaut), la manivelle fonctionne en mode position. L'axe se déplace exactement selon l'incrément de jog sélectionné pour chaque impulsion, sans s'occuper du temps que prendra le mouvement. Quand elle est TRUE, la manivelle fonctionne en mode vitesse. Le mouvement s'arrête quand la manivelle s'arrête, même si le mouvement commandé n'est pas achevé.
- *axis.N.joint-pos-cmd* - (float, out) La position commandée de l'articulation (par opposition à celle du moteur). Ca peut être un écart entre les positions articulation et moteur. Par exemple; la procédure de prise d'origine fixe cet écart.
- *axis.N.joint-pos-fb* - (float, out) Le retour de position de l'articulation (par opposition à celui du moteur).
- *axis.N.joint-vel-cmd* - (float, out)
- *axis.N.kb-jog-active* - (bit, out)
- *axis.N.motor-pos-cmd* - (float, out) Position commandée pour cette articulation.
- *axis.N.motor-pos-fb* - (float, in) Position actuelle de cette articulation.
- *axis.N.neg-hard-limit* - (bit, out)
- *axis.N.pos-lim-sw-in* - (bit, in) Doit être mis TRUE si le fin de course de limite positive de cette articulation est activé.
- *axis.N.pos-hard-limit* - (bit, out)
- *axis.N.neg-lim-sw-in* - (bit, in) Doit être mis TRUE si le fin de course de limite négative de cette articulation est activé.
- *axis.N.wheel-jog-active* - (bit, out)

7.2.2 Paramètres

- *axis.N.home-state* - Reflète l'étape de la prise d'origine en cours actuellement.

7.3 iocontrol (espace utilisateur)

Ces pins sont créées par le contrôleur d'entrées/sorties de l'espace utilisateur, habituellement appelé *io*.

7.3.1 Pins

- *iocontrol.0.coolant-flood* - (bit, out) TRUE quand l'arrosage est demandé.
- *iocontrol.0.coolant-mist* - (bit, out) TRUE quand le brouillard est demandé.
- *iocontrol.0.emc-enable-in* - (bit, in) Doit être mise FALSE quand un arrêt d'urgence externe est activé.
- *iocontrol.0.lube* - (bit, out) TRUE quand le graissage centralisé est commandé.
- *iocontrol.0.lube_level* - (bit, in) Doit être mise TRUE quand le niveau d'huile est correct.
- *iocontrol.0.tool-change* - (bit, out) TRUE quand un changement d'outil est demandé.
- *iocontrol.0.tool-changed* - (bit, in) Doit être mise TRUE quand le changement d'outil est terminé.
- *iocontrol.0.tool-number* - (s32, out) Numéro de l'outil courant.
- *iocontrol.0.tool-prep-number* - (s32, out) Numéro du prochain outil, donné par le mot **T** selon RS274NGC.
- *iocontrol.0.tool-prepare* - (bit, out) TRUE quand une préparation d'outil est demandée.
- *iocontrol.0.tool-prepared* - (bit, in) Doit être mise TRUE quand une préparation d'outil est terminée.
- *iocontrol.0.user-enable-out* - (bit, out) FALSE quand un arrêt d'urgence interne est activé.
- *iocontrol.0.user-request-enable* - (bit, out) TRUE quand l'arrêt d'urgence est relâché.

Chapitre 8

Configuration d'un système pas/direction (dir/step)

8.1 Introduction

Ce chapitre décrit quelques uns des réglages les plus fréquents, sur lesquels l'utilisateur aura à agir lors de la mise au point de LinuxCNC. En raison de l'adaptabilité de LinuxCNC, il serait très difficile de les documenter tous en gardant ce document relativement concis.

Le système rencontré le plus fréquemment chez les utilisateurs de LinuxCNC est un système à moteurs pas à pas. Les interfaces de pilotage de ces moteurs recoivent de LinuxCNC des signaux de pas et de direction.

C'est le système le plus simple à mettre en oeuvre parce que les moteurs fonctionnent en boucle ouverte (pas d'information de retour des moteurs), le système nécessite donc d'être configuré correctement pour que les moteurs ne perdent pas de pas et ne calent pas.

Ce chapitre s'appuie sur la configuration fournie d'origine avec LinuxCNC appelée *stepper* et qui se trouve habituellement dans */etc/linuxcnc/sample-configs/stepper*.

8.2 Fréquence de pas maximale

Avec la génération logicielle des pas la fréquence maximale en sortie, pour les impulsions de pas et de direction, est de une impulsion pour deux BASE_PERIOD. La fréquence de pas maximale accessible pour un axe est le produit de MAX_VELOCITY et de INPUT_SCALE. Si la fréquence demandée est excessive, une erreur de suivi se produira (following error), particulièrement pendant les jog rapides et les mouvements en G0.

Si votre interface de pilotage des moteurs accepte des signaux d'entrée en quadrature, utilisez ce mode. Avec un signal en quadrature, un pas est possible pour chaque BASE_PERIOD, ce qui double la fréquence maximum admissible.

Les autres remèdes consistent à diminuer une ou plusieurs variables: BASE_PERIOD (une valeur trop faible peux causer un bloquage du PC), INPUT_SCALE (s'il est possible sur l'interface de pilotage de sélectionner une taille de pas différente, de changer le rapport des poulies ou le pas de la vis mère), ou enfin MAX_VELOCITY et STEPGEN_MAXVEL.

Si aucune combinaison entre BASE_PERIOD, INPUT_SCALE et MAX_VELOCITY n'est fonctionnelle, il faut alors envisager un générateur de pas externe (parmis les contrôleurs de moteurs pas à pas universels supportés par LinuxCNC)

8.3 Brochage

LinuxCNC est très flexible et grâce à la couche d'abstraction de HAL (Hardware Abstraction Layer) il est facile de spécifier que tel signal ira sur telle broche.

Voir le tutoriel de HAL pour plus d'informations.

Comme décrit dans l'introduction et la manuel de HAL, il comporte des composants dont il fourni les signaux, les pins et les paramètres.

Les premiers signaux et pins relatifs au brochage sont: [1]

```
signaux: Xstep, Xdir et Xen
pins: parport.0.pin-XX-out & parport.0.pin-XX-in
```

Pour configurer le fichier ini, il est possible de choisir entre les deux brochages les plus fréquents, devenus des standards de fait, le brochage standard_pinout.hal ou le brochage xylotex_pinout.hal. Ces deux fichiers indiquent à HAL comment raccorder les différents signaux aux différentes pins. Dans la suite, nous nous concentrerons sur le brochage *standard_pinout.hal*.

8.4 Le fichier standard_pinout.hal

Ce fichier contient certaines commandes de HAL et habituellement ressemble à celà:

```
# standard pinout config file for 3-axis steppers
# using a parport for I/O
#

# first load the parport driver
loadrt hal_parport cfg="0x0378"

#
# next connect the parport functions to threads
# read inputs first
addf parport.0.read base-thread 1

# write outputs last
addf parport.0.write base-thread -1

#
# finally connect physical pins to the signals
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

# create a signal for the estop loopback
net estop-loop iocontrol.0.user-enable-out iocontrol.0.emc-enable-in

# create signals for tool loading loopback
net tool-prep-loop iocontrol.0.tool-prepare iocontrol.0.tool-prepared
net tool-change-loop iocontrol.0.tool-change iocontrol.0.tool-changed

# connect "spindle on" motion controller pin to a physical pin
net spindle-on motion.spindle-on => parport.0.pin-09-out

###
### You might use something like this to enable chopper drives when machine ON
### the Xen signal is defined in core_stepper.hal
###
# net Xen => parport.0.pin-01-out

###
```

1. Note: pour rester concis, nous ne présenterons qu'un seul axe, tous les autres sont similaires.

```
### If you want active low for this pin, invert it like this:
###
# setp parport.0.pin-01-out-invert 1

###
### A sample home switch on the X axis (axis 0).  make a signal,
### link the incoming parport pin to the signal, then link the signal
### to LinuxCNC's axis 0 home switch input pin
###
# net Xhome parport.0.pin-10-in => axis.0.home-sw-in

###
### Shared home switches all on one parallel port pin?
### that's ok, hook the same signal to all the axes, but be sure to
### set HOME_IS_SHARED and HOME_SEQUENCE in the ini file.  See the
### user manual!
###
# net homeswitches <= parport.0.pin-10-in
# net homeswitches => axis.0.home-sw-in
# net homeswitches => axis.1.home-sw-in
# net homeswitches => axis.2.home-sw-in

###
### Sample separate limit switches on the X axis (axis 0)
###
# net X-neg-limit parport.0.pin-11-in => axis.0.neg-lim-sw-in
# net X-pos-limit parport.0.pin-12-in => axis.0.pos-lim-sw-in

###
### Just like the shared home switches example, you can wire together
### limit switches.  Beware if you hit one, LinuxCNC will stop but can't tell
### you which switch/axis has faulted.  Use caution when recovering from this.
###
# net Xlimits parport.0.pin-13-in => axis.0.neg-lim-sw-in axis.0.pos-lim-sw-in
```

Les lignes commençant par **#** sont des commentaires, aident à la lecture du fichier.

8.5 Vue d'ensemble du fichier standard_pinout.hal

Voici les opérations qui sont exécutées quand le fichier standard_pinout.hal est lu par l'interpréteur:

1. Le pilote du port parallèle est chargé (voir le Parport section de le Manuel de HAL pour plus de détails)
2. Les fonctions de lecture/écriture du pilote sont assignée au thread «Base thread» [2]
3. Les signaux du générateur de pas et de direction des axes X,Y,Z… sont raccordés aux broches du port parallèle
4. D'autres signaux d'entrées/sorties sont connectés (boucle d'arrêt d'urgence, boucle du changeur d'outil…)
5. Un signal de marche broche est défini et raccordé à une broche du port parallèle

8.6 Modifier le fichier standard_pinout.hal

Pour modifier le fichier standard_pinout.hal, il suffit de l'ouvrir dans un éditeur de texte puis d'y localiser les parties à modifier.

Si vous voulez par exemple, modifier les broches de pas et de direction de l'axe X, il vous suffit de modifier le numéro de la variable nommée *parport.0.pin-XX-out*:

```
net Xstep parport.0.pin-03-out
net Xdir  parport.0.pin-02-out
```

2. Le thread le plus rapide parmis les réglages de LinuxCNC, habituellement il n'y a que quelques microsecondes entre les exécutions de ce code.

peut être modifiée pour devenir:

```
net Xstep parport.0.pin-02-out
net Xdir  parport.0.pin-03-out
```

ou de manière générale n'importe quel numéro que vous souhaiteriez.

Attention: il faut être certain de n'avoir qu'un seul signal connecté à une broche.

8.7 Modifier la polarité d'un signal

Si une interface attends un signal *actif bas*, ajouter une ligne avec le paramètre d'inversion de la sortie, *-invert*. Par exemple, pour inverser le signal de rotation de la broche:

```
setp parport.0.pin-09-invert TRUE
```

8.8 Ajouter le contrôle de vitesse broche en PWM

Si votre vitesse de broche peut être contrôlée par un signal de PWM, utilisez le composant *pwmgen* pour créer ce signal:

```
loadrt pwmgen output_type=0
addf pwmgen.update servo-thread
addf pwmgen.make-pulses base-thread
net spindle-speed-cmd motion.spindle-speed-out => pwmgen.0.value
net spindle-on motion.spindle-on => pwmgen.0.enable
net spindle-pwm pwmgen.0.pwm => parport.0.pin-09-out
setp pwmgen.0.scale 1800 # Change to your 'spindles top speed in RPM
```

Ce qui donnera le fonctionnement suivant, pour un signal PWM à: 0% donnera une vitesse de 0tr/mn, 10% une vitesse de 180tr/mn, etc. Si un signal PWM supérieur à 0% est requis pour que la broche commence à tourner, suivez l'exemple du fichier de configuration *nist-lathe* qui utilise un composant d'échelle (*scale*).

8.9 Ajouter un signal de validation enable

Certains pilotes de moteurs requiert un signal de validation *enable* avant d'autoriser tout mouvement du moteur. Pour celà des signaux sont déjà définis et appelés *Xen*, *Yen*, *Zen*.

Pour les connecter vous pouvez utilisez l'exemple suivant:

```
net Xen parport.0.pin-08-out
```

Il est possible d'avoir une seule pin de validation pour l'ensemble des pilotes, ou plusieurs selon la configuration que vous voulez. Notez toutefois qu'habituellement quand un axe est en défaut, tous les autres sont invalidés aussi de sorte que, n'avoir qu'un seul signal/pin de validation pour l'ensemble est parfaitement sécurisé.

8.10 Ajouter un bouton d'Arrêt d'Urgence externe

Comme vous pouvez le voir ici, par défaut la configuration standard n'utilise pas de bouton d'Arrêt d'Urgence externe. footnote:[Une explication complète sur la manière de gérer les circuiteries d'Arrêt d'Urgence se trouve sur le wiki(en) et dans le Manuel de l'intégrateur.

Pour ajouter un simple bouton d'AU externe (ou plusieurs en série) vous devez remplacer la ligne suivante:

```
net estop-loop iocontrol.0.user-enable-out iocontrol.0.emc-enable-in
```

par

```
net estop-loop parport.0.pin-01-in iocontrol.0.emc-enable-in
```

Ce qui implique qu'un bouton d'Arrêt d'Urgence soit connecté sur la broche 01 du port parallèle. Tant que le bouton est enfoncé (le contact ouvert) [3], LinuxCNC restera dans l'état *Arrêt d'Urgence* (ESTOP). Quand le bouton externe sera relâché, LinuxCNC passera immédiatement dans l'état *Arrêt d'Urgence Relâché* (ESTOP-RESET) vous pourrez ensuite mettre la machine en marche en pressant le bouton *Marche machine* et vous êtes alors prêt à continuer votre travail avec LinuxCNC. :leveloffset: 0 = Interfaces graphiques utilisateur :leveloffset: 1 :lang: fr :toc:

3. Utiliser exclusivement des contacts normalement fermés pour les A/U.

Chapitre 9

PyVCP

9.1 Introduction

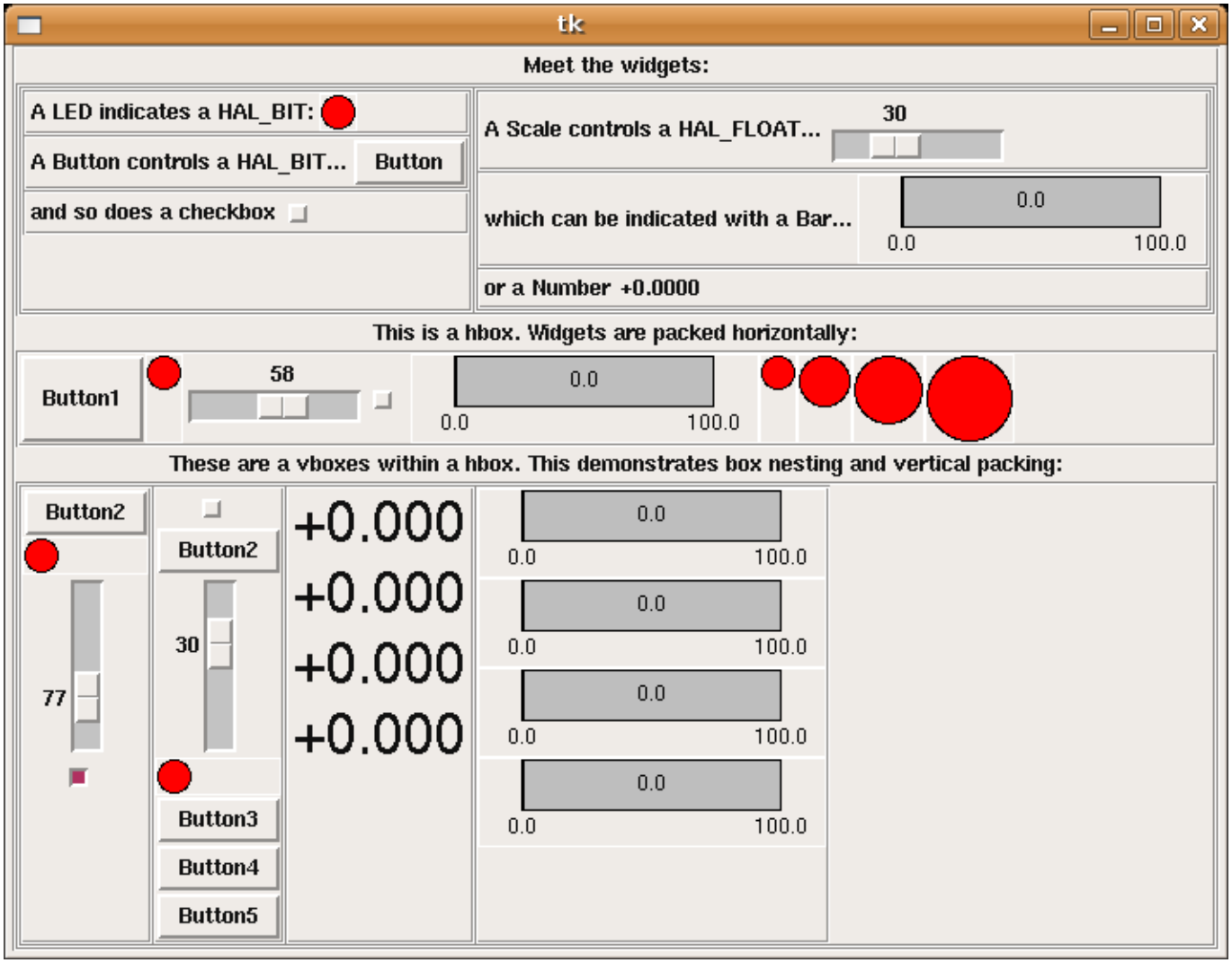
Panneau virtuel de contrôle en python (**Py**thon **V**irtual **C**ontrol **P**anel)

Le panneau de contrôle virtuel pyVCP a été créé pour donner à l'intégrateur la possibilité de personnaliser l'interface graphique d'AXIS avec des boutons et des indicateurs destinés aux tâches spéciales.

Le coût d'un panneau de contrôle physique est très élevé et il peut utiliser un grand nombre de broches d'entrées/sorties. C'est là que le panneau virtuel prends l'avantage car il ne coûte rien d'utiliser pyVCP.

Les panneaux de contrôle virtuels peuvent être utilisés pour tester ou monitorer le matériel, les entrées/sorties, remplacer temporairement d'autres matériels d'entrées/sorties pendant le déboguage d'une logique ladder ou pour simuler un panneau physique, avant de le construire et de le câbler vers les cartes électroniques.

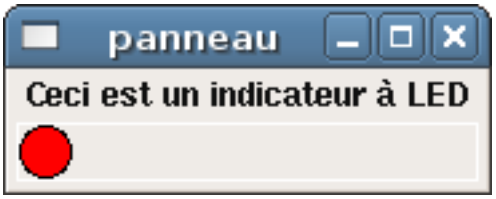
L'image suivante montre quelques widgets pyVCP.



9.2 Construction d'un panneau pyVCP

La disposition d'un panneau pyVCP est spécifiée dans un fichier XML qui contient les balises des widgets entre <pyvcp> et </pyvcp>. Par exemple:

```
<pyvcp>
   <label text="Ceci est un indicateur à LED"/>
   <led/>
</pyvcp>
```



Si vous placez ce texte dans un fichier nommé tiny.xml et que vous le lancez avec:

```
pyvcp -c panneau tiny.xml
```

pyVCP va créer le panneau pour vous, il y inclut deux widgets, un Label avec le texte *Ceci est un indicateur à LED* et une LED rouge, utilisée pour afficher l'état d'un signal HAL de type BIT. Il va aussi créer un composant HAL nommé *panneau* (tous les widgets dans ce panneau sont connectés aux pins qui démarrent avec *panneau*). Comme aucune balise <halpin> n'était présente à l'intérieur de la balise <led>, pyVCP nomme automatiquement la pin HAL pour le widget LED panneau.led.0

Pour obtenir la liste des widgets, leurs balises et options, consultez la documentation des widgets.

Un fois que vous avez créé votre panneau, connectez lui les signaux HAL, vers et à partir des pins pyVCP et avec la commande habituelle:

```
net <signal-name> <pin-name> <opt-direction> <opt-pin-name>signal-name
```

Si vous débutez avec HAL, le tutoriel de HAL est vivement recommandé.

9.3 Sécurité avec pyVCP

Certaines parties de pyVCP sont évaluées comme du code Python, elles peuvent donc exécuter n'importe quelle action disponible dans les programmes Python. N'utilisez que des fichiers pyVCP en .xml à partir d'une source de confiance.tag

9.4 Utiliser pyVCP avec AXIS

Puisque AXIS utilise le même environnement graphique et les même outils (Tkinter) que pyVCP, il est possible d'inclure un panneau pyVCP sur le côté droit de l'interface utilisateur normale d'AXIS. Un exemple typique est présenté ci-dessous.

Placer le fichier pyVCP XML décrivant le panneau dans le même répertoire que le fichier .ini. Nous voulons afficher la vitesse courante de la broche sur un widget barre de progression. Copier le code XML suivant dans un fichier appelé broche.xml:

```
<pyvcp>
    <label>
        <text>"Vitesse broche:"</text>
    </label>
    <bar>
        <halpin>"spindle-speed"</halpin>
        <max_>5000</max_>
    </bar>
</pyvcp>
```

Ici nous avons fait un panneau avec un label *Vitesse broche:* et un widget barre de progression. Nous avons spécifié que la pin HAL connectée à la barre de progression devait s'appeler *spindle-speed* et régler la valeur maximum de la barre à 5000 (se reporter à la documentation des widgets, pour toutes les options disponibles). Pour faire connaître ce fichier à AXIS et qu'il l'appelle au démarrage, nous devons préciser ce qui suit dans la section [DISPLAY] du fichier .ini:

```
PYVCP = broche.xml
```

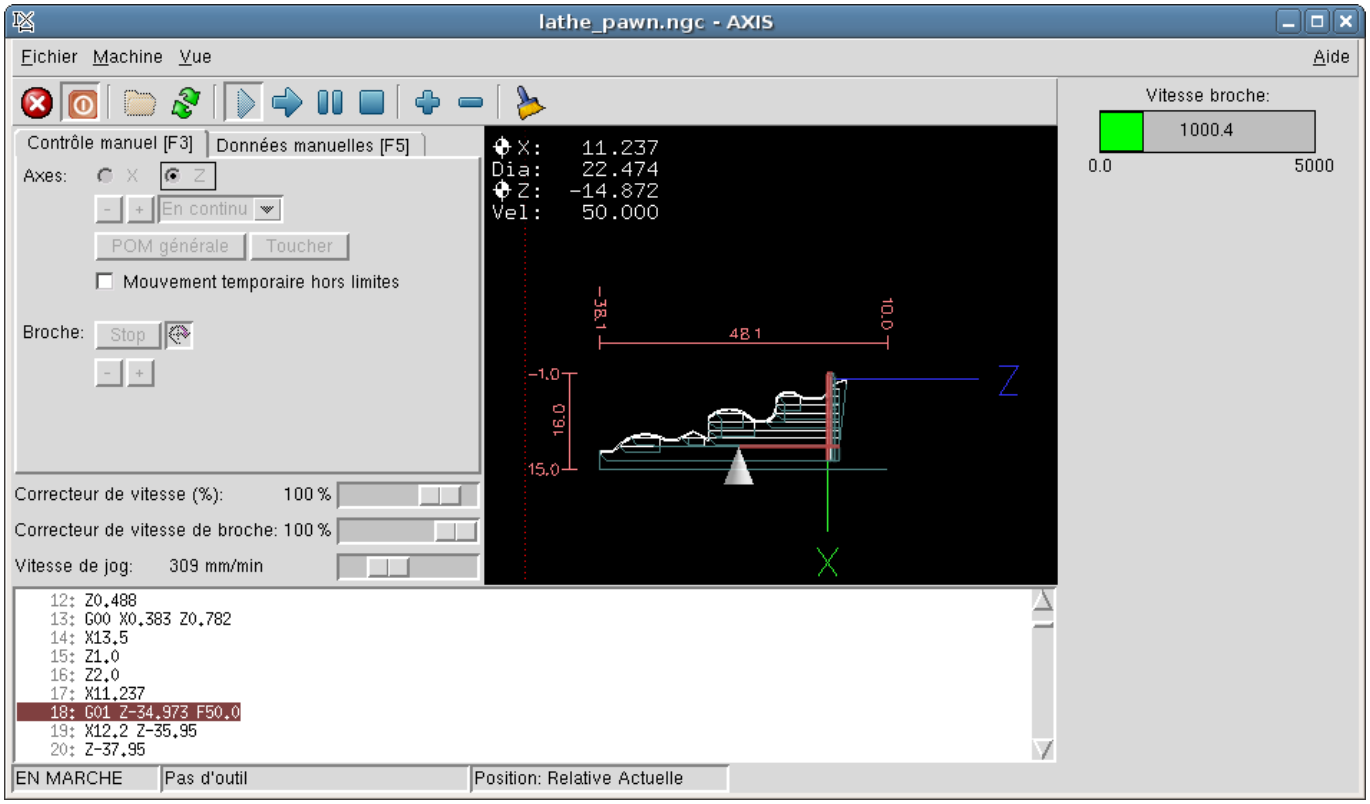
Pour que notre widget affiche réellement la vitesse de la broche *spindle-speed*, il doit être raccordé au signal approprié de HAL. Le fichier .hal qui sera exécuté quand AXIS et pyVCP démarreront doit être spécifié, de la manière suivante, dans la section [HAL] du fichier .ini:

```
POSTGUI_HALFILE = broche_vers_pyvcp.hal
```

Ce changement lancera la commande HAL spécifiée dans *broche_vers_pyvcp.hal*. Dans notre exemple, ce fichier contiendra juste la commande suivante:

```
net spindle-rpm-filtered  => pyvcp.spindle-speed
```

ce qui suppose que le signal appelé *spindle-rpm-filtered* existe aussi. Noter que lors de l'exécution avec AXIS, toutes les pins des widgets de pyVCP ont des noms commençant par *pyvcp.*.



Voila à quoi ressemble le panneau pyVCP que nous venons de créer, incorporé à AXIS. La configuration *sim/lathe* fournie en exemple, est configurée de cette manière.

9.5 Panneaux PyVCP autonomes

Cette section va décrire comment les panneaux PyVCP peuvent être affichés par eux même, par l'intermédiaire ou non des contrôleurs machine de LinuxCNC.

Pour charger un panneau PyVCP autonome avec LinuxCNC utiliser cette commande:

```
loadusr -Wn monpanneau pyvcp -g WxH+X+Y -c monpanneau <path/>fichier_panneau.xml
```

Vous l'utiliserez pour avoir un panneau flottant ou un panneau avec une interface graphique autre que Axis.

– *-Wn monpanneau* - Fait attendre à HAL que le composant *monpanneau* soit chargé (devienne *ready* en langage HAL), avant d'exécuter d'autres commandes HAL. C'est important parce-que les panneaux PyVCP exportent des pins de HAL ainsi que d'autres composants de HAL qui doivent être présents pour pouvoir se connecter à eux. Noter la lettre **W** en majuscule et la lettre **n** en minuscule. Si vous utilisez l'option -Wn vous devez également utiliser l'option -c pour nommer le panneau.
– *pyvcp* < *-g*> < *-c*> *panneau.xml* - Construit le panneau avec la géométrie optionnelle et/ou le nom de panneau depuis le fichier panneau.xml. Le fichier panneau.xml peut avoir n'importe quel nom avec l'extension .xml. Le fichier .xml décrit comment construire le panneau. Il est nécessaire d'ajouter le nom du chemin si le panneau n'est pas dans le répertoire dans lequel se trouve le script HAL.
– *-g <WxH><+X+Y>* - Spécifie la géométrie à utiliser quand le panneau est construit. La syntaxe est *Largeur x Hauteur + Ancrage X + Ancrage Y*. La taille ou la position, ou les deux peuvent être fixés. Le point d'ancrage est le coin supérieur gauche du panneau. Par exemple; -g 250x500+800+0 fixe le panneau à 250 pixels de large, 500 pixels de haut avec le point d'ancrage placé en X800 Y0.
– *-c nompanneau* - Indique à PyVCP quel composant appeler et le titre de la fenêtre. Le nom du fichier *nompanneau* peut être n'importe quel nom sans espace.

Pour charger un panneau PyVCP autonome, sans LinuxCNC utiliser cette commande:

```
loadusr -Wn monpanneau pyvcp -g 250x500+800+0 -c monpanneau monpanneau.xml
```

La commande minimale pour charger un panneau pyvcp est la suivante:

```
loadusr pyvcp monpanneau.xml
```

Vous pourrez utiliser cette commande si vous voulez un panneau sans passer par un des contrôleurs machine de LinuxCNC, par exemple pour des tests ou une visu autonome.

La commande loadusr est utilisée quand vous chargez aussi un composant qui stoppera HAL depuis la fermeture jusqu'à ce qu'il soit prêt. Si vous avez chargé un panneau puis chargé Classic Ladder en utilisant la commande *loadusr -w classicladder*, CL maintiendra HAL et le panneau ouverts jusqu'à ce que vous fermiez Classic Ladder. Le *-Wn* signifie d'attendre que le composant *-Wn "nom"* devienne prêt. (*nom* peut être n'importe quel nom. Noter la lettre **W** en majuscule et le **n** en minuscule.) Le -c indique à PyVCP de construire un panneau avec le nom *monpanneau* en utilisant les infos contenues dans le fichier *monpanneau.xml*. Le nom du fichier *monpanneau.xml* est sans importante mais doit porter l'extension .xml. C'est le fichier qui décrit comment construire le panneau. Il est nécessaire d'ajouter le nom du chemin si le panneau n'est pas dans le répertoire dans lequel se trouve le script HAL.

Une commande optionnelle à utiliser si vous voulez que le panneau stoppe HAL depuis les commandes *Continuer* / *Quitter*. Après avoir chargé n'importe quelles autres composants la dernière commande HAL sera:

```
waituser nompanneau
```

Cette commande indique à HAL d'attendre que le composant *nompanneau* soit fermé avant de continuer avec d'autres commandes. C'est généralement défini comme étant la dernière commande, de sorte que HAL s'arrêtera si le panneau est fermé.

9.6 Documentation des widgets de pyVCP

Les signaux de HAL existent en deux variantes, BIT et FLOAT. pyVCP peut afficher la valeur d'un signal avec un widget indicateur, ou modifier la valeur d'un signal avec un widget de contrôle. Ainsi, il y a quatre classes de widgets pyVCP connectables aux signaux de HAL. Une cinquième classe de widgets d'aide permet d'organiser et d'appliquer des labels aux panneaux.
– Widgets de signalisation, signaux de type bit: led, rectled
– Widgets de contrôle, signaux de type bit: button, checkbutton, radiobutton
– Widgets de signalisation de type nombre: number, s32, u32, bar, meter
– Widgets de contrôle de type nombre: spinbox, scale, jogwheel
– Widgets d'aide: hbox, vbox, table, label, labelframe

9.6.1 Syntaxe

Chaque widget sera décrit brièvement, suivi par la forme d'écriture utilisée et d'une capture d'écran. Toutes les balises contenues dans la balise du widget principal, sont optionnelles.

9.6.2 Notes générales

Á l'heure actuelle, les deux syntaxes, basée sur les balises et basée sur les attributs, sont supportées. Par exemple, les deux fragments de code XML suivants sont traités de manière identique:

```
<led halpin="ma-led"/>
```

et

```
<led><halpin>"ma-led"</halpin></led>
```

Quand la syntaxe basée sur les attributs est utilisée, les règles suivantes sont utilisées pour convertir les valeurs des attributs en valeurs Python:

1. Si le premier caractère de l'attribut est un des suivants: *{([" '* , il est évalué comme une expression Python.
2. Si la chaine est acceptée par int(), la valeur est traitée comme un entier.

3. Si la chaine est acceptée par float(), la valeur est traitée comme un flottant.
4. Autrement, la chaine est acceptée comme une chaine.

Quand la syntaxe basée sur les balises est utilisée, le texte entre les balises est toujours évalué comme un expression Python.

Les exemples ci-dessous montrent un mélange des deux formats.

9.6.3 Commentaires

Pour ajouter un commentaire utiliser la syntaxe de xml.

```
<!-- Mon commentaire -->
```

9.6.4 Editer un fichier XML

Editer le fichier XML avec un éditeur de texte. La plupart du temps un double click sur le nom de fichier permet de choisir *ouvrir avec l'editeur de texte* ou similaire.

9.6.5 Couleurs

Les couleurs peuvent être spécifiées en utilisant les couleurs RGB de X11 soit par le nom, par exemple: *gray75* ou soit en hexa décimal, par exemple: *#0000ff*. Une liste complète est consultable ici: http://sedition.com/perl/rgb.html.

Couleurs les plus courantes (les numéros suivant la couleur indiquent la nuance de la couleur)
– white (blanc)
– black (noir)
– blue et blue1 - blue4 (bleu)
– cyan et cyan1 - cyan4 (cyan)
– green et green1 - green4 (vert)
– yellow et yellow1 - yellow4 (jaune)
– red et red1 - red4 (rouge)
– purple et purple1 - purple4 (violet/pourpre)
– gray et gray0 - gray100 (gris)

9.6.6 Pins de HAL

Les pins de HAL fournisse le moyen de connecter les widgets aux autres éléments. Quand une pin de HAL est créée pour un widget, il est possible de la *connecter* à une autre pin de HAL avec une commande *net* dans un fichier .hal. Pour plus de détails, voir la commande *net* dans le manuel de HAL.

9.6.7 Label

Un label est un texte qui s'affiche sur le panneau.

Le label a une pin optionnelle de désactivation en ajoutant: *<disable_pin>True</disable_pin>*.

```
<label>
    <text>"Ceci est un label:"</text>
    <font>("Helvetica",20)</font>
</label>
```

Ce code produira:

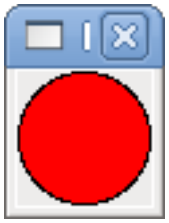
9.6.8 Les leds

Une led est utilisée pour indiquer l'état d'une pin de HAL de type bit. La couleur de la led sera on_color quand le signal est vrai et off_color autrement. * <*halpin*> définit le nom de la pin, par défaut: *led.n*, où n est un entier. * <*size*> définit la taille de la led, par défaut: 20. * <*on_color*> définit la couleur de la led led quand la pin est vraie, par défaut: *green* * <*off_color*> définit la couleur de la led quand la pin est fausse, par défaut: *ref*

9.6.9 La led ronde

```
<led>
    <halpin>"ma-led"</halpin>
    <size>50</size>
    <on_color>"verte"</on_color>
    <off_color>"rouge"</off_color>
</led>
```

Le résultat du code ci-dessus.



9.6.10 La led rectangulaire

C'est une variante du widget *led*.

```
<vbox>
    <relief>RIDGE</relief>
    <bd>6</bd>
    <rectled>
        <halpin>"ma-led-rect"</halpin>
        <height>"50"</height>
        <width>"100"</width>
        <on_color>"green"</on_color>
        <off_color>"red"</off_color>
    </rectled>
</vbox>
```

Le code ci-dessus produit cette led, entourée d'un relief.

9.6.11 Le bouton (button)

Un bouton permet de contrôler une pin de type bit. La pin sera mise vraie quand le bouton sera pressé et maintenu enfoncé, elle sera mise fausse quand le bouton sera relâché.

Les boutons peuvent suivre les options de formatage suivantes:
– <padx>n</padx> où *n* est le nombre d'espaces horizontaux supplémentaires
– <pady>n</pady> où *n* est le nombre d'espaces verticaux supplémentaires
– <activebackground>"color"</activebackground> Couleur au survol du curseur
– <bg>"color"</bg> Couleur du bouton

9.6.11.1 Bouton avec texte (Text Button)

```
<button>
    <halpin>"Bouton-OK"</halpin>
    <text>"OK"</text>
</button>
<button>
    <halpin>"Bouton-Abandon"</halpin>
    <text>"Abort"</text>
</button
```

Le code ci-dessus produit:

9.6.11.2 Case à cocher (checkbutton)

Une case à cocher contrôle une pin de type bit. La pin sera mise vraie quand la case est cochée et fausse si la case est décochée.

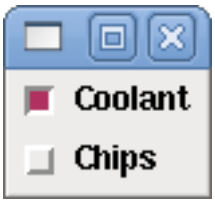
Une case non cochée:

et une case cochée:

Exemple de code:

```
<checkbutton>
    <halpin>"coolant-chkbtn"</halpin>
    <text>"Coolant"</text>
</checkbutton>
<checkbutton>
    <halpin>"chip-chkbtn"</halpin>
    <text>"Chips    "</text>
</checkbutton>
```

Le code ci-dessus produit:



9.6.11.3 Bouton radio (radiobutton)

Un bouton radio placera une seule des pins vraie. Les autres seront mises fausses.

```
<radiobutton>
    <choices>["un","deux","trois"]</choices>
    <halpin>"mon-radiobtn"</halpin>
</radiobutton>
```

Le code ci-dessus donne ce résultat:

Noter que dans l'exemple ci-dessus, les pins de HAL seront nommées mon-radiobtn.un, mon-radiobtn.deux et mon-radiobtn.trois. Dans l'image précédente, *trois* est la valeur sélectionnée courante.

9.6.12 Affichage d'un nombre (number)

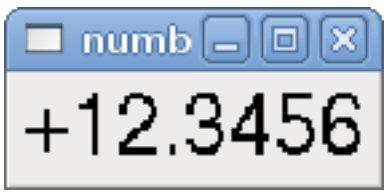
L'affichage d'un nombre peux recevoir les options de formatage suivantes:
– ("Font Name",n) où *n* est la taille de la police
– <width>n</width> où *n* est la largeur totale utilisée
– <justify>pos</justify> où "pos" peut être LEFT, CENTER ou RIGHT (devrait marcher)
– <padx>n</padx> où "n" est le nombre d'espaces horizontaux supplémentaires
– <pady>n</pady> où "n" est le nombre d'espaces verticaux supplémentaires

9.6.12.1 Number

Le widget *number* affiche la valeur d'un signal de type flottant.

```
<number>
    <halpin>"number"</halpin>
    <font>("Helvetica",24)</font>
    <format>"+4.4f"</format>
</number>
```

Le code ci-dessus donne ce résultat:



9.6.12.2 Flottant

Le widget number affiche la valeur d'un signal de type flottant.

```
<number>
    <halpin>"my-number"</halpin>
    <font>("Helvetica",24)</font>
    <format>"+4.4f"</format>
</number>
```

<font> est une police de caractères de type Tkinter avec la spécification de sa taille. Une police qui peut être agrandie jusqu'à la taille 200 est la police *courier 10 pitch*, que vous pouvez spécifier de la manière suivante, pour afficher des chiffres réellement grands:

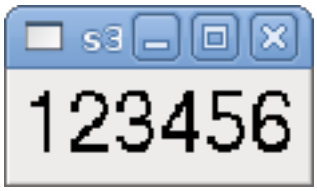
```
<font>('courier 10 pitch',100)</font>
```

<format> est un format *style C*, spécifié pour définir le format d'affichage du nombre.

9.6.12.3 Nombre s32

Le widget s32 affiche la valeur d'un nombre s32. La syntaxe est la même que celle de *number* excepté le nom qui est <s32>. Il faut prévoir une largeur suffisante pour afficher le nombre dans sa totalité.

```
<s32>
    <halpin>"simple-number"</halpin>
    <font>("Helvetica",24)</font>
    <format>"6d"</format>
    <width>6</width>
</s32>
```



9.6.12.4 Nombre u32

Le widget u32 affiche la valeur d'un nombre u32. La syntaxe est la même que celle de *number* excepté le nom qui est <u32>.

9.6.13 Affichage d'images

Seul l'affichage d'images au format gif est possible. Toutes les images doivent avoir la même taille. Les images doivent être toutes dans le même répertoire que le fichier ini (ou dans le répertoire courant pour un fonctionnement en ligne de commande avec halrun/halcmd).

9.6.13.1 Image Bit

La bascule *image_bit* bascule entre deux images selon la position vraie ou fausse de halpin.

```
<pyvcp>
    <image name='fwd' file='fwd.gif'/>
    <image name='rev' file='rev.gif'/>
    <vbox>
        <image_bit halpin='selectimage' images='fwd rev'/>
    </vbox>
</pyvcp>
```

En utilisant les deux images fwd.gif et rev.gif. FWD est affiché quand *selectimage* est fausse et REV est affiché quand *selectimage* est vraie.

FIGURE 9.1 – selectimage fausse

FIGURE 9.2 – selectimage vraie

9.6.13.2 Image u32

La bascule *image_u32* est la même que *image_bit* excepté que le nombre d'images n'est pratiquement plus limité, il suffit de *selectionner* l'image en ajustant halpin à une valeur entière commençant à 0 pour la première image de la liste, à 1 pour la seconde image etc.

```
<pyvcp>
    <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>
</pyvcp>
```

Même résultat mais en ajoutant l'image stb.gif.

FIGURE 9.3 – Halpin = 0

FIGURE 9.4 – Halpin = 1

FIGURE 9.5 – Halpin = 2

9.6.14 Barre de progression (bar)

Le widget barre de progression affiche la valeur d'un signal FLOAT, graphiquement dans une barre de progression et simultanément, en numérique.

```
<bar>
    <halpin>"bar"</halpin>
    <min_>0</min_>
    <max_>123</max_>
    <bgcolor>"grey"</bgcolor>
    <fillcolor>"red"</fillcolor>
</bar>
```

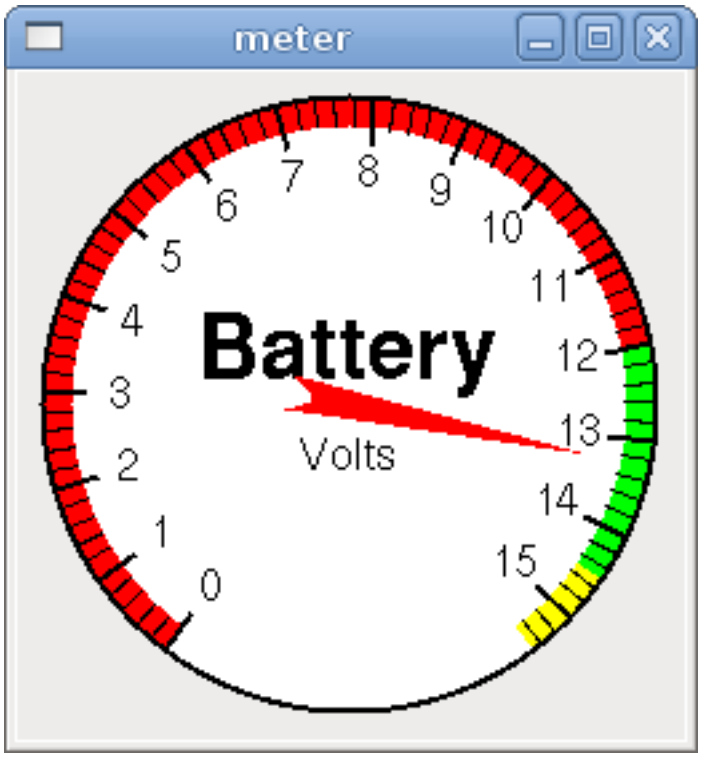
Le code ci-dessus donne ce résultat:

9.6.15 Galvanomètre (meter)

Le galvanomètre affiche la valeur d'un signal FLOAT dans un affichage à aiguille *à l'ancienne*.

```
<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>
```

Le code ci-dessus donne ce résultat:



9.6.16 Boîte d'incrément (spinbox)

La boîte d'incrément contrôle une pin FLOAT. La valeur de la pin est augmentée ou diminuée de la valeur de *resolution*, à chaque pression sur une flèche, ou en positionnant la souris sur le nombre puis en tournant la molette de la souris.

```
<spinbox>
    <halpin>"my-spinbox"</halpin>
    <min_>-12</min_>
    <max_>33</max_>
    <inival>0</inival>
    <resolution>0.1</resolution>
    <format>"2.3f"</format>
    <font>("Arial",30)</font>
</spinbox>
```

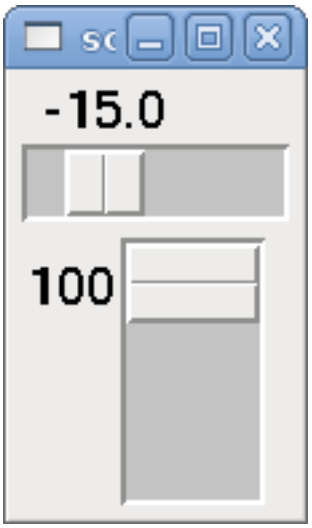
Le code ci-dessus donne ce résultat:

9.6.17 Curseur (scale)

Le curseur contrôle une pin FLOAT. La valeur de la pin est augmentée ou diminuée en déplaçant le curseur, ou en positionnant la souris sur le curseur puis en tournant la molette de la souris.

```
<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_>
</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_>
</scale>
```

Le code ci-dessus donne ce résultat:



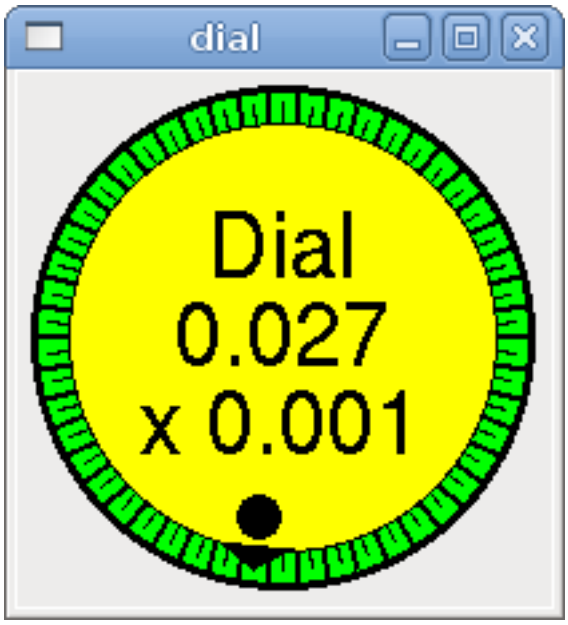
Noter que par défaut c'est min qui est affiché même si il est supérieur à max, à moins que min ne soit négatif.

9.6.18 Bouton tournant (dial)

Le bouton tournant imite le fonctionnement d'un vrai bouton tournant, en sortant sur un FLOAT HAL la valeur sur laquelle est positionné le curseur, que ce soit en le faisant tourner avec un mouvement circulaire, ou en tournant la molette de la souris. Un double click gauche augmente la résolution et un double click droit la diminue d'un digit. La sortie est limitée par les valeurs min et max. La variable cpr fixe le nombre de graduations sur le pourtour du cadran (prudence avec les grands nombres).

```
<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>
</dial>
```

Le code ci-dessus donne ce résultat:

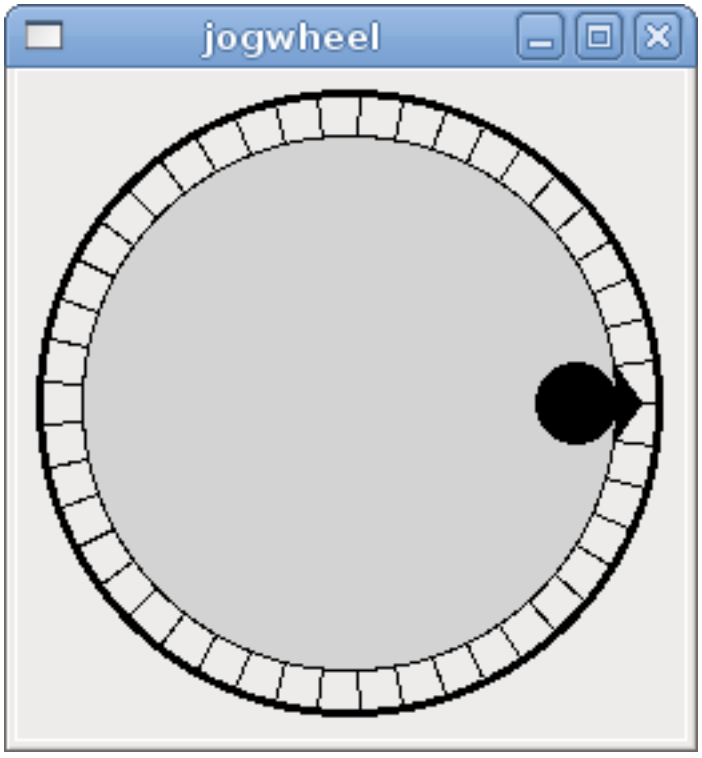


9.6.19 Manivelle (jogwheel)

La manivelle imite le fonctionnement d'une vraie manivelle, en sortant sur une pin FLOAT la valeur sur laquelle est positionné le curseur, que ce soit en le faisant tourner avec un mouvement circulaire, ou en tournant la molette de la souris.

```
<jogwheel>
    <halpin>"my-wheel"</halpin>
    <cpr>45</cpr>
    <size>250</size>
</jogwheel>
```

Le code ci-dessus donne ce résultat:



9.7 Documentation des containers de pyVCP

Les containers sont des widgets qui contiennent d'autres widgets.

9.7.1 Bordures

Le container bordure est spécifié avec deux balises utilisées ensembles. La balise <relief> spécifie le type de bordure et la balise <bd> spécifie la largeur de la bordure.

<relief>type</relief>
La valeur de *type* peut être: FLAT, SUNKEN, RAISED, GROOVE, ou RIDGE

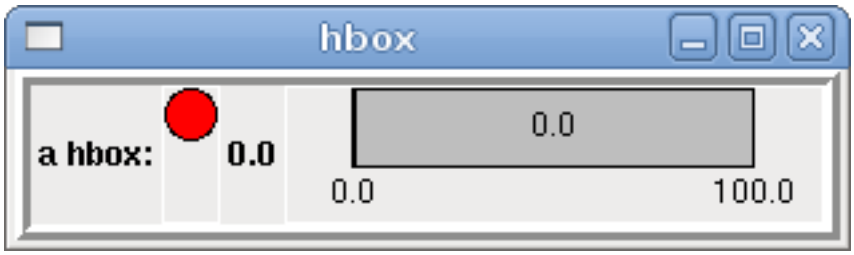
<bd>n</bd>
La valeur de **n** fixe la largeur de la bordure.

```
<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>
```

9.7.2 Hbox

Utilisez une Hbox lorsque vous voulez aligner les widgets, horizontalement, les uns à côtés des autres.

```
<hbox>
    <relief>RIDGE</relief>
    <bd>6</bd>
    <label><text>"a hbox:"</text></label>
    <led></led>
    <number></number>
    <bar></bar>
</hbox>
```

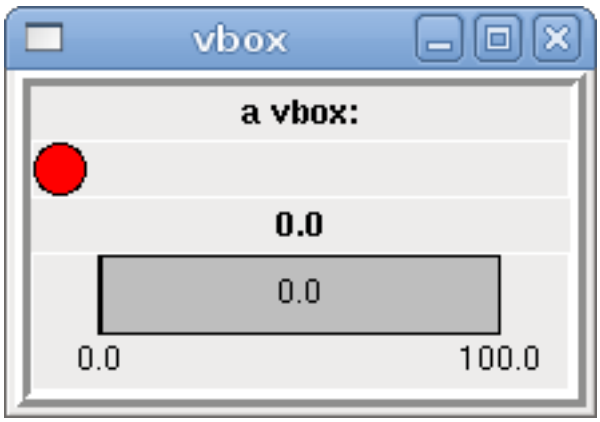


Á l'intérieur d'une Hbox, il est possible d'utiliser les balises <*boxfill fill=/*>, <*boxanchor anchor=/*> et <*boxexpand expand=/*> pour choisir le comportement des éléments contenus dans la boîte, lors d'un redimensionnement de la fenêtre. Pour des détails sur le comportement de fill, anchor, et expand, référez vous au manuel du pack Tk, *pack(3tk)*. Valeurs par défaut, *fill='y'*, *anchor='center'*, *expand='yes'*.

9.7.3 Vbox

Utilisez une Vbox lorsque vous voulez aligner les widgets verticalement, les uns au dessus des autres.

```
<vbox>
    <relief>RIDGE</relief>
    <bd>6</bd>
    <label><text>"a vbox:"</text></label>
    <led></led>
    <number></number>
    <bar></bar>
</vbox>
```



Á l'intérieur d'une Vbox, vous pouvez utiliser les balises <*boxfill fill=/*>, <*boxanchor anchor=/*> et <*boxexpand expand=/*> pour choisir le comportement des éléments contenus dans la boîte, lors d'un redimensionnement de la fenêtre. Pour des détails sur le comportement de fill, anchor, et expand, référez vous au manuel du pack Tk, *pack(3tk)*. Valeurs par défaut, *fill='y'*, *anchor='center'*, *expand='yes'*.

9.7.4 Labelframe

Un labelframe est un cadre entouré d'un sillon et un label en haut à gauche.

```
<labelframe text="Label: Leds égroupes">
```

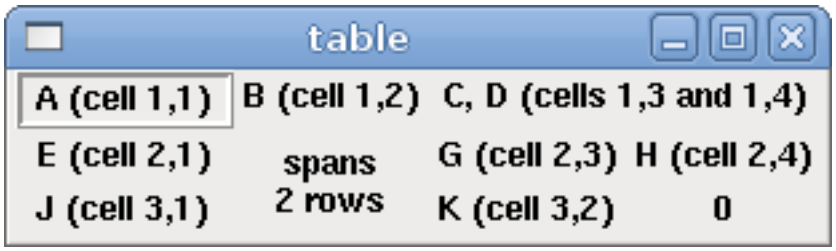
```
<labelframe text="Label: Leds égroupes">
    <font>("Helvetica",16)</font>
    <hbox>
    <led/>
    <led/>
    <led/>
    </hbox>
</labelframe>
```

9.7.5 Table

Une table est un container qui permet d'écrire dans une grille de lignes et de colonnes. Chaque ligne débute avec la balise <*tablerow*/> . Un widget contenu peut être en lignes ou en colonnes par l'utilisation de la balise <*tablespan rows*= *cols*=/>. Les bordures des cellules contenant les widgets *sticky* peuvent être réglées grâce à l'utilisation de la balise <*tablesticky sticky*=/>. Une table flexible peut s'étirer sur ses lignes et ses colonnes (sticky).

Exemple:

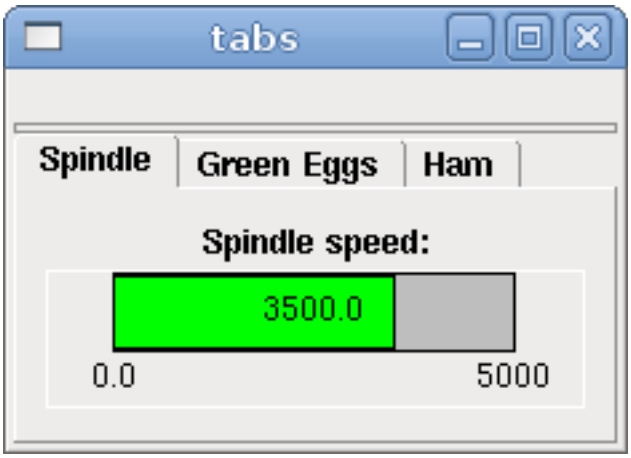
```
<table flexible_rows="[2]" flexible_columns="[1,4]">
<tablesticky sticky="new"/>
<tablerow/>
    <label>
        <text>" A (cell 1,1) "</text>
        <relief>RIDGE</relief>
        <bd>3</bd>
    </label>
    <label text="B (cell 1,2)"/>
    <tablespan columns="2"/>
    <label text="C, D (cells 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>
```



9.7.6 Onglets (Tabs)

Une interface à onglets permet d'économiser l'espace en créant un container pour chaque nom d'onglet (tabs). Une seule section *tabs* peut exister, les *tabs* ne peuvent pas être imbriqués ni empilés. La largeur de l'onglet le plus large, determine la largeur des onglets.

```
<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>
```



Chapitre 10

Exemples d'utilisation de PyVCP

10.1 Panneau PyVCP dans AXIS

Procédure pour créer un panneau PyVCP et l'utiliser, attaché dans la partie droite de l'interface AXIS.
– Créer un fichier .xml contenant la description du panneau et le placer dans le répertoire de la configuration.
– Ajouter une entrée, avec le nom du fichier .xml, dans la section [DISPLAY] du fichier ini.
– Ajouter une entrée POSTGUI_HALFILE, avec le nom du fichier postgui HAL.
– Ajouter les liens vers les pins de HAL pour le panneau dans le fichier postgui.hal pour "connecter" le panneau PyVCP avec LinuxCNC.

10.2 Panneaux flottants

Pour créer des panneaux flottants PyVCP pouvant être utilisés avec n'importe quelle interface, suivre les points suivants:
– Créer un fichier .xml contenant la description du panneau et le placer dans le répertoire de configuration.
– Ajouter les lignes *loadusr* dans le fichier.hal pour charger chaque panneau.
– Ajouter les liens vers les pins de HAl du panneau dans le fichier postgui.hal, pour "connecter" les panneaux PyVCP à LinuxCNC.
L'exemple suivant montre le chargement de deux panneaux PyVCP.

```
loadusr -Wn btnpanel pyvcp -c btnpanel panel1.xml
loadusr -Wn sppanel pyvcp -c sppanel panel2.xml
```

Les paramètres -Wn font que HAL **W**ait for **n**ame, attends le composant nommé *btnpanel*.

Les paramètres pyvcp -c font que PyVCP nomme le panneau.

Les pins de HAL de panel1.xml seront nommées *btnpanel.<pin name>*

Les pins de HAL de panel2.xml seront nommées *sppanel.<pin name>*

Bien s'assurer qu'aucune ligne *loadusr* ne fasse déjà appel à une de ces pins PyVCP.

10.3 Boutons de Jog

Dans cet exemple nous allons créer un panneau PyVCP avec 3 boutons utilisables pour déplacer en manuel les axes X, Y et Z. Cette configuration sera réalisée avec l'assistant Stepconf qui générera la configuration de la machine. Premièrement, nous lançons l'assistant Stepconf et le configurons pour la machine, ensuite dans la page *Advanced Configuration Options* nous effectuons les sélections pour ajouter un panneau PyVCP vierge, comme indiqué sur l'image suivante. Pour cet exemple nous avons nommé la configuration *pyvcp_xyz* sur la page *Basic Machine Information* de l'assistant Stepconf.

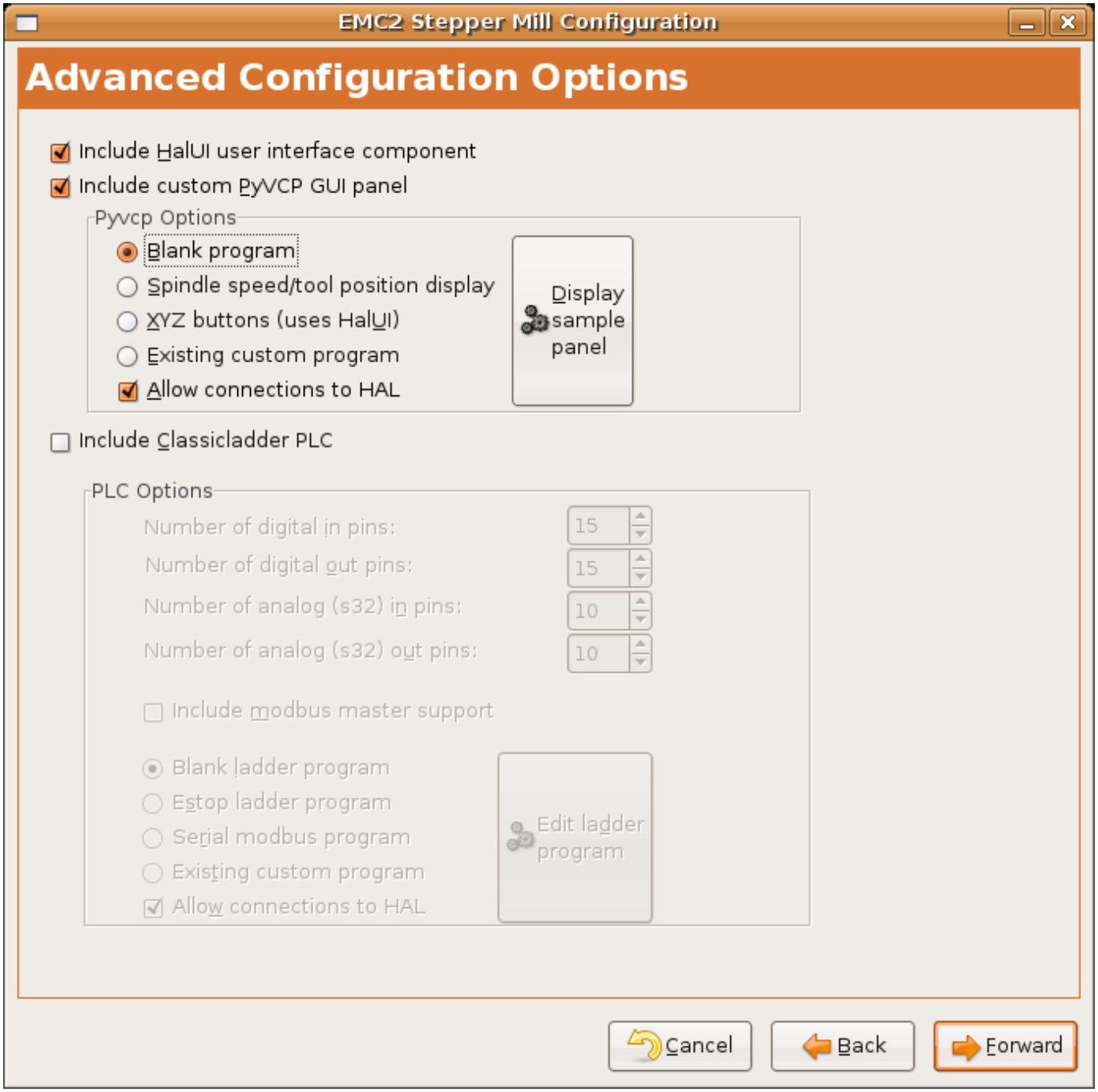


FIGURE 10.1 – Assistant de configuration XYZ

L'assistant Stepconf Wizard va créer plusieurs fichiers et les placer dans le répertoire */emc/configs/pyvcp_xyz*. Si la case *Créer un lien* est cochée, un lien vers ces fichiers sera créé sur le bureau.

10.3.1 Créer les Widgets

Ouvrir le fichier custompanel.xml par un clic droit sur son nom, puis en sélectionnant *Ouvrir avec l'éditeur de texte*. Entre les balises *<pyvcp>* et *</pyvcp>* nous ajouterons les widgets pour le panneau.

Tous les détails sur chacun des Widgets de PyVCP sont donnés dans la documentation des widgets.

Dans le fichier custompanel.xml nous ajoutons la description des widgets.

```
<pyvcp>

    <labelframe text="Boutons de Jog">
       <font>("Helvetica",16)</font>

        <!-- le bouton de jog de l'axe X -->
        <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-moins"</halpin>
                <text>"X-"</text>
            </button>
        </hbox>

        <!-- le bouton de jog de l'axe Y -->
        <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-moins"</halpin>
                <text>"Y-"</text>
            </button>
        </hbox>

        <!-- le bouton de jog de l'axe Z -->
        <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-moins"</halpin>
                <text>"Z-"</text>
            </button>
        </hbox>

        <!-- le curseur de vitesse de jog -->
        <vbox>
```

```
            <relief>RAISED</relief>
            <bd>3</bd>
            <label>
                <text>"Vitesse de Jog"</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>
```

Après les ajouts précédents, nous avons un panneau PyVCP tel que celui de l'image suivante, attaché à droite d'Axis. Il est beau mais ne fait rien tant que les boutons ne sont pas "connectés" à halui. Si, à ce stade, une erreur se produit lors du déplacement de la fenêtre vers le bas, c'est généralement dû à une erreur de syntaxe ou d'écriture, elle est donc dans cette partie qu'il conviendra tout d'abord de vérifier soigneusement.

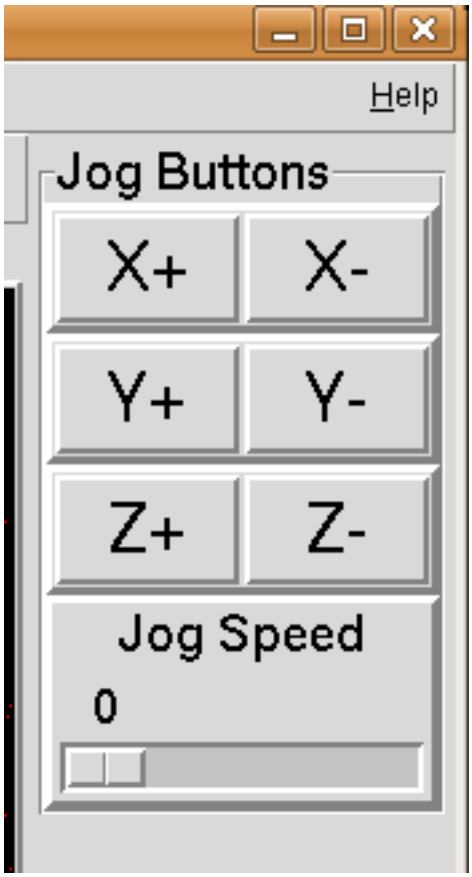


FIGURE 10.2 – Boutons de Jog

10.3.2 Effectuer les connections

Pour effectuer les connections nécessaires, ouvrir le fichier custom_postgui.hal et y ajouter le code suivant:

```
# connecte les boutons PyVCP pour X
net my-jogxmoins halui.jog.0.minus <= pyvcp.x-moins
net my-jogxplus halui.jog.0.plus <= pyvcp.x-plus

# connecte les boutons PyVCP pour Y
net my-jogymoins halui.jog.1.minus <= pyvcp.y-moins
net my-jogyplus halui.jog.1.plus <= pyvcp.y-plus

# connecte les boutons PyVCP pour Z
net my-jogzmoins halui.jog.2.minus <= pyvcp.z-moins
net my-jogzplus halui.jog.2.plus <= pyvcp.z-plus

# connecte le curseur de vitesse de jog PyVCP
net my-jogspeed halui.jog-speed <= pyvcp.jog-speed-f
```

Après avoir désactivé l'A/U (E-Stop) et activé la marche machine en mode Jog, le déplacement du curseur du panneau PyVCP devrait agir dès qu'il est placé au delà de zéro et les boutons de jog devraient fonctionner. Il est impossible de jogger alors qu'un fichier G-code s'exécute ou pendant qu'il est en pause ni quand l'onglet *Données manuelles [F5]* du (MDI), est ouvert.

10.4 Testeur de port

Cet exemple montre comment faire un simple testeur de port parallèle en utilisant PyVCP et HAL.

Premièrement, créer le fichier ptest.xml qui contiendra le code suivant pour créer la description du panneau.

```
<!-- Panneau de test pour la config. du port èparallle -->
<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>
```

Le panneau flottant contenant deux pins de HAL d'entrée et deux pins de HAL de sortie.

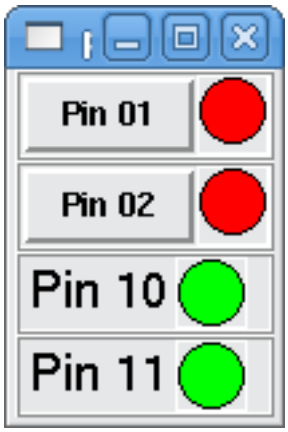


FIGURE 10.3 – Panneau flottant testeur de port parallèle

Pour lancer les commandes de HAL dont nous avons besoin et démarrer tout ce qi'il nous faut, nous avons mis le code suivant dans notre fichier ptest.hal.

```
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
```

Pour lancer le fichier HAL, nous utilisons, dans un terminal, les commandes suivantes:

```
~$ halrun -I -f ptest.hal
```

La figure suivante montre à quoi ressemble le panneau complet.

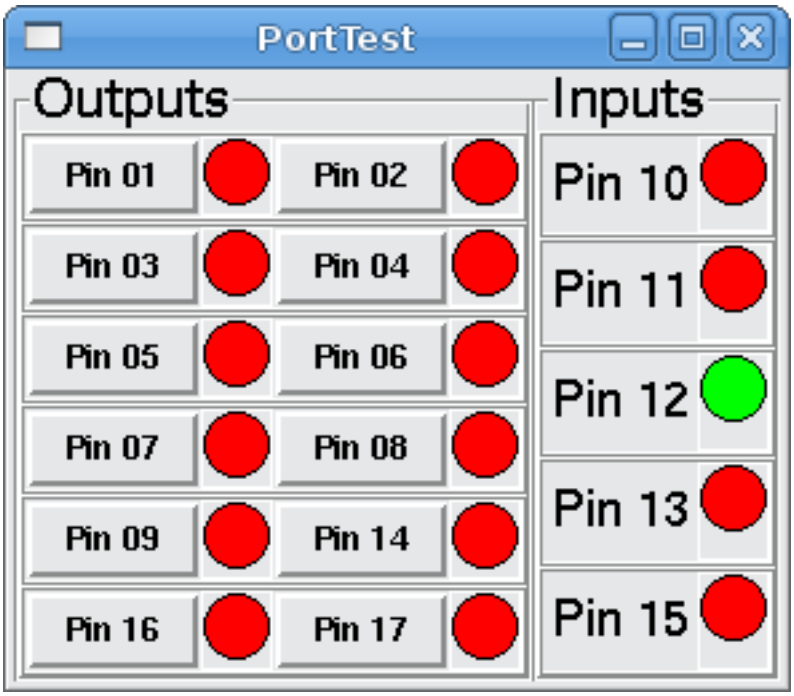


FIGURE 10.4 – Testeur de port parallèle, complet

Pour ajouter le reste des pins du port parallèle, il suffi de modifier les fichiers .xml et .hal.

Pour visualiser les pins après avoir lancé le script HAL, utiliser la commande suivante au prompt *halcmd:*

```
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
```

Cela montre quelles pins sont IN est lesquelles sont OUT, ainsi que toutes les connections.

10.5 Compte tours pour GS2

L'exemple suivant utilise un variateur de fréquence GS2 de la société Automation Direct. [1] Il permet le pilotage du moteur, la visualisation de la vitesse ainsi que d'autres informations dans un panneau PyVCP. Cet exemple est basé sur un autre, relatif au variateur GS2 et se trouvant dans la section des exemples matériels de ce manuel. Ce dernier exemple s'appuie lui même sur la description du composant de HAL gs2_vfd.

10.5.1 Le panneau

Pour créer le panneau nous ajoutons ce code au fichier .xml.

```
<pyvcp>

    <!-- Compte tours -->
    <hbox>
        <relief>RAISED</relief>
        <bd>3</bd>
        <meter>
            <halpin>"spindle_rpm"</halpin>
            <text>"Broche"</text>
            <subtext>"tr/mn"</subtext>
            <size>200</size>
            <min_>0</min_>
            <max_>3000</max_>
            <majorscale>500</majorscale>
            <minorscale>100</minorscale>
            <region1>0,10,"yellow"</region1>
        </meter>
    </hbox>

    <!-- La Led On -->
    <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"/> <!-- éutilis pour centrer la Led -->
                <rectled>
                    <halpin>"on-led"</halpin>
                    <height>"30"</height>
```

1. En Europe on trouve ce type de variateur sous la marque Omron.

```
                    <width>"30"</width>
                    <on_color>"green"</on_color>
                    <off_color>"red"</off_color>
                </rectled>
            </hbox>
        </vbox>

        <!-- La Led Sens horaire -->
        <vbox>
            <relief>RAISED</relief>
            <bd>2</bd>
            <label>
                <text>"Sens horaire"</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>

        <!-- La Led Sens inverse -->
        <vbox>
            <relief>RAISED</relief>
            <bd>2</bd>
            <label>
                <text>"Sens inverse"</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>
```

L'image ci-dessous montre notre panneau PyVCP en fonctionnement.

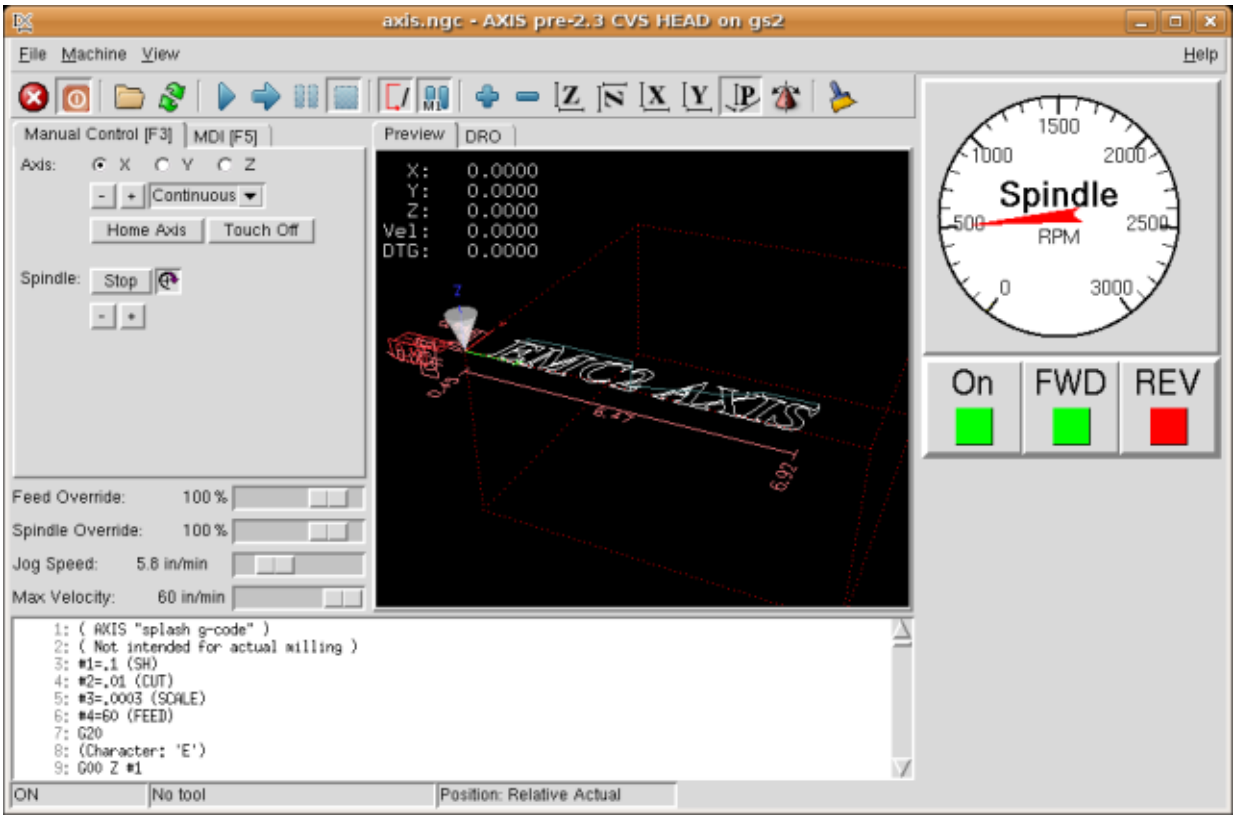


FIGURE 10.5 – Panneau pour GS2

10.5.2 Les connections

Pour qu'il fonctionne, il est nécessaire d'ajouter le code suivant au fichier custom_postgui.hal, il réalise les connections entre PyVCP et LinuxCNC.

```
# affiche le compte tours, calcul ébas sur freq * rpm par 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

# la led On
net gs2-run => pyvcp.on-led

# la led Sens horaire
net gs2-fwd => pyvcp.fwd-led

# la led Sens anti-horaire
net running-rev spindle-vfd.spindle-rev => pyvcp.rev-led
```

Certaines lignes demandent quelques explications.

– La ligne de la led Sens horaire utilise le signal créé dans le fichier custom.hal dans lequel la led Sens inverse doit utiliser le bit *spindle-rev*.
– On ne *peut pas* lier deux fois le bit *spindle-fwd* pour utiliser le signal auquel il est déjà lié.

Chapitre 11

Création d'interfaces graphiques avec GladeVCP

11.1 Qu'est-ce que GladeVCP?

GladeVCP est un composant de LinuxCNC qui donne la possibilité d'ajouter de nouvelles interfaces graphiques utilisateur à LinuxCNC telles qu'Axis ou Touchy. À la différence de PyVCP, GladeVCP n'est pas limité à l'affichage et aux réglages des pins de HAL, toutes les actions peuvent être exécutées en code Python. En fait, une interface utilisateur LinuxCNC complète peut être construite avec GladeVCP et Python.

GladeVCP utilise l'environnement graphique et WYSIWYG Glade qui simplifie l'édition et la création visuelle de panneaux esthétiquement très réussis. Il s'appuie sur les liaisons entre PyGTK et le riche jeu de widgets GTK+, finalement, tous peuvent être utilisés dans une application GladeVCP et pas seulement les widgets spécialisés pour interagir avec HAL et LinuxCNC présentés ici.

11.1.1 PyVCP par rapport à GladeVCP

Tous les deux supportent la création de panneaux avec des *widgets de HAL*, des éléments utilisateur visuels tels que boutons, Leds, curseurs etc. dont les valeurs sont liées à des pins de HAL qui à leur tour, sont des interfaces pour le reste de LinuxCNC.

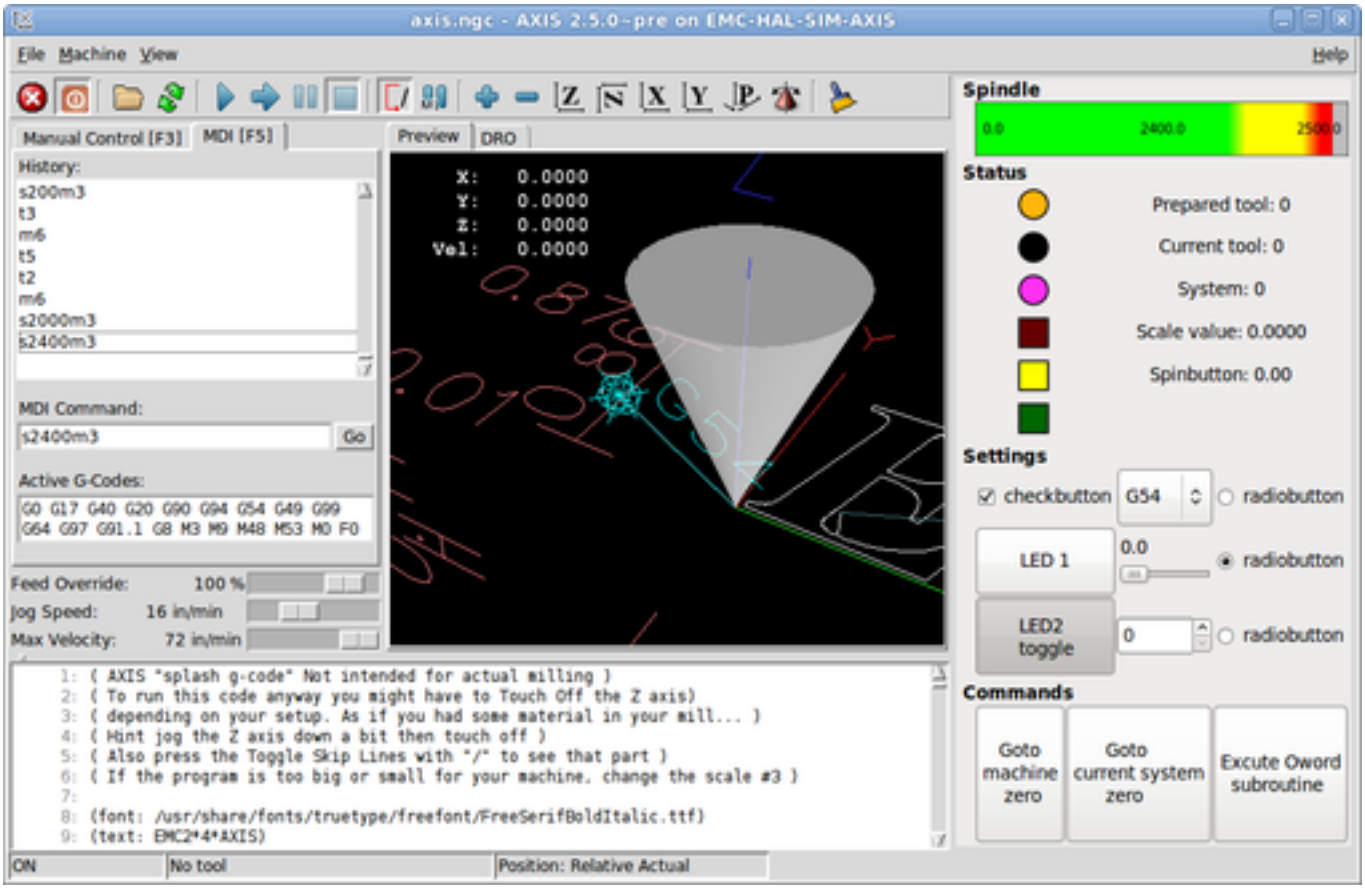
PyVCP
- Jeu de widgets: utilise les widgets TkInter.
- Cycle de création d'interfaces utilisateur:
  - Éditer les fichiers XML
  - Lancer
  - Évaluer le look.
- Pas de support pour intégrer une gestion des événements définie par l'utilisateur.
- Pas d'interaction avec LinuxCNC au-delà des interactions avec les pins d'E/S de HAL supportées.

GladeVCP
- Jeu de widgets: Liaison avec le jeu de widgets de GTK+.
- Création d'interface utilisateur: utilise l'interface graphique Glade qui est un éditeur WYSIWYG.
- Tout changement sur une pin de HAL peut diriger un appel vers une gestion d'événements définie en Python par l'utilisateur.
- Tous les signaux GTK (touches/appui sur un bouton, fenêtre, E/S, timer, événements réseau) peuvent être associés avec la gestion d'événements définie en Python par l'utilisateur.
- Interaction directe avec LinuxCNC: exécution de commandes, telle qu'initialiser une commande MDI pour appeler un sous-programme G-code.
- Plusieurs panneaux GladeVCP indépendants peuvent tourner dans des onglets différents.
- Séparation entre l'apparence de l'interface et les fonctionnalités: change d'apparence sans passer par aucun code.

11.2 Description du fonctionnement, avec un exemple de panneau

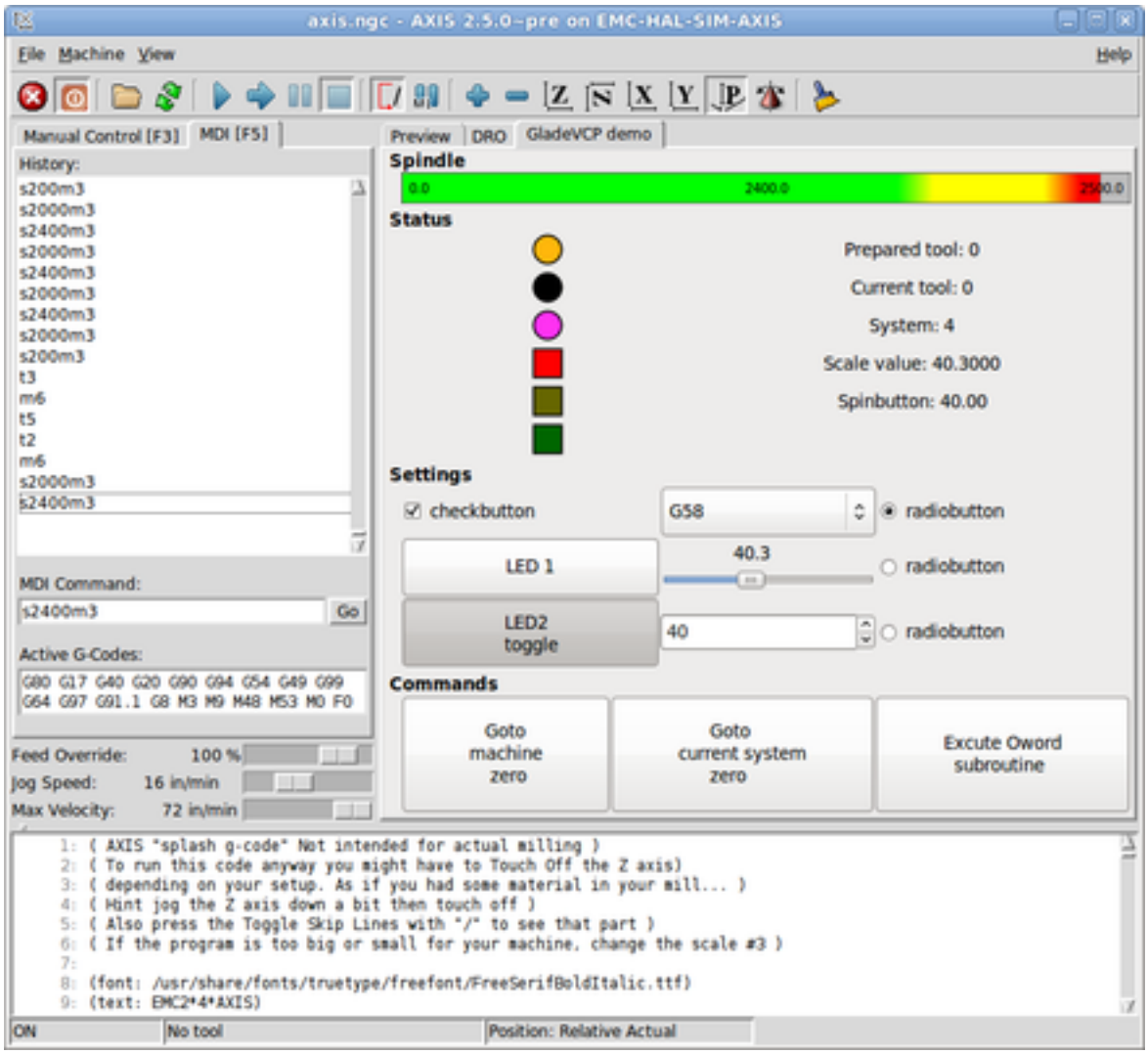
Une fenêtre de panneau GladeVCP peut démarrer avec trois différentes configuration:
– Toujours visible, intégré dans Axis, du côté droit, exactement comme un panneau PyVCP.
– Dans un onglet dans Axis ou Touchy; dans Axis un troisième onglet sera créé à côté des deux d'origine, ils doivent être choisis explicitement.
– Comme une fenêtre indépendante, qui peut être iconisée ou agrandie, indépendamment de la fenêtre principale.
Lancer un panneau GladeVCP simple, intégré dans Axis comme PyVCP, taper les commandes suivantes:

```
$ cd configs/sim/gladevcp

$ linuxcnc gladevcp_panel.ini
```



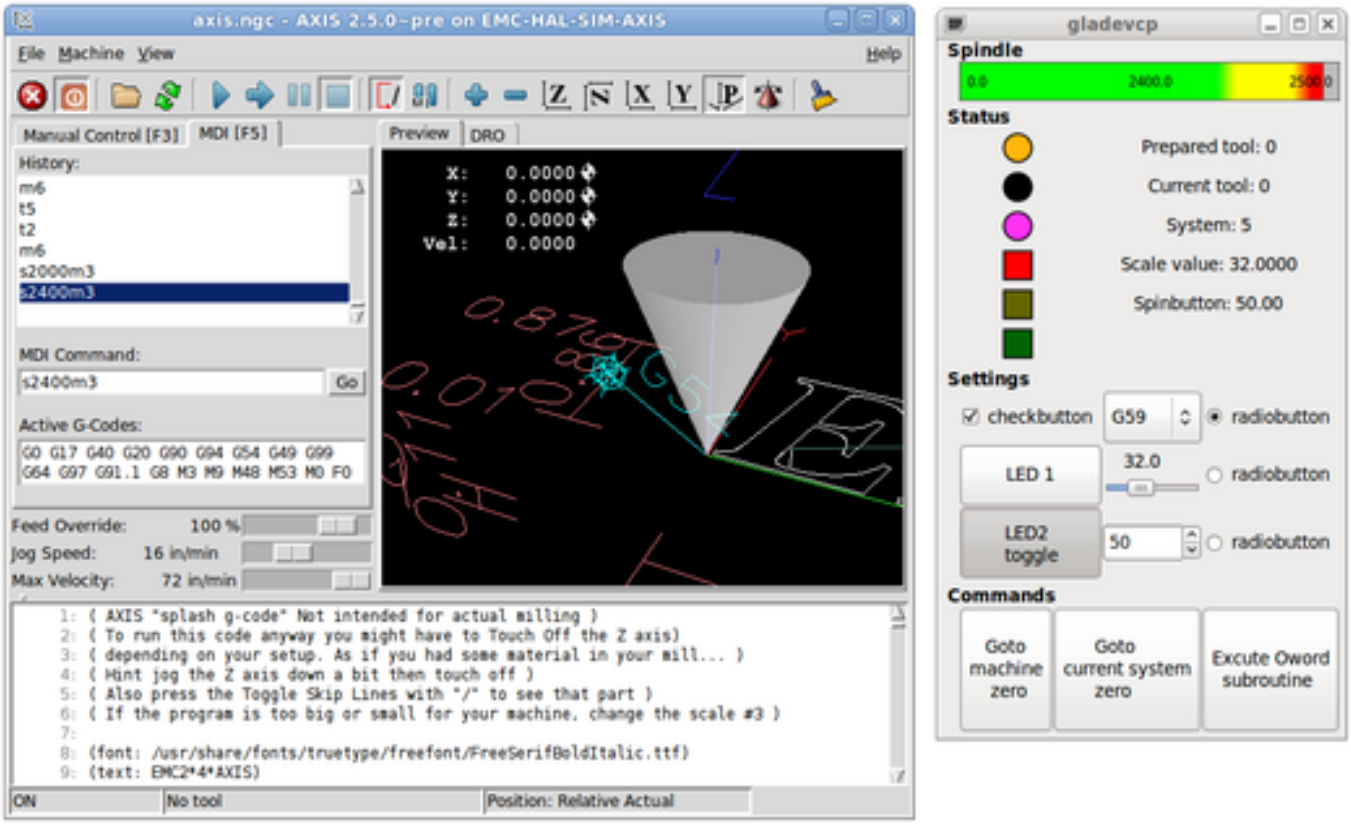
Lancer le même panneau, mais dans un onglet d'Axis avec:

```
$ cd configs/sim/gladevcp

$ linuxcnc gladevcp_tab.ini
```



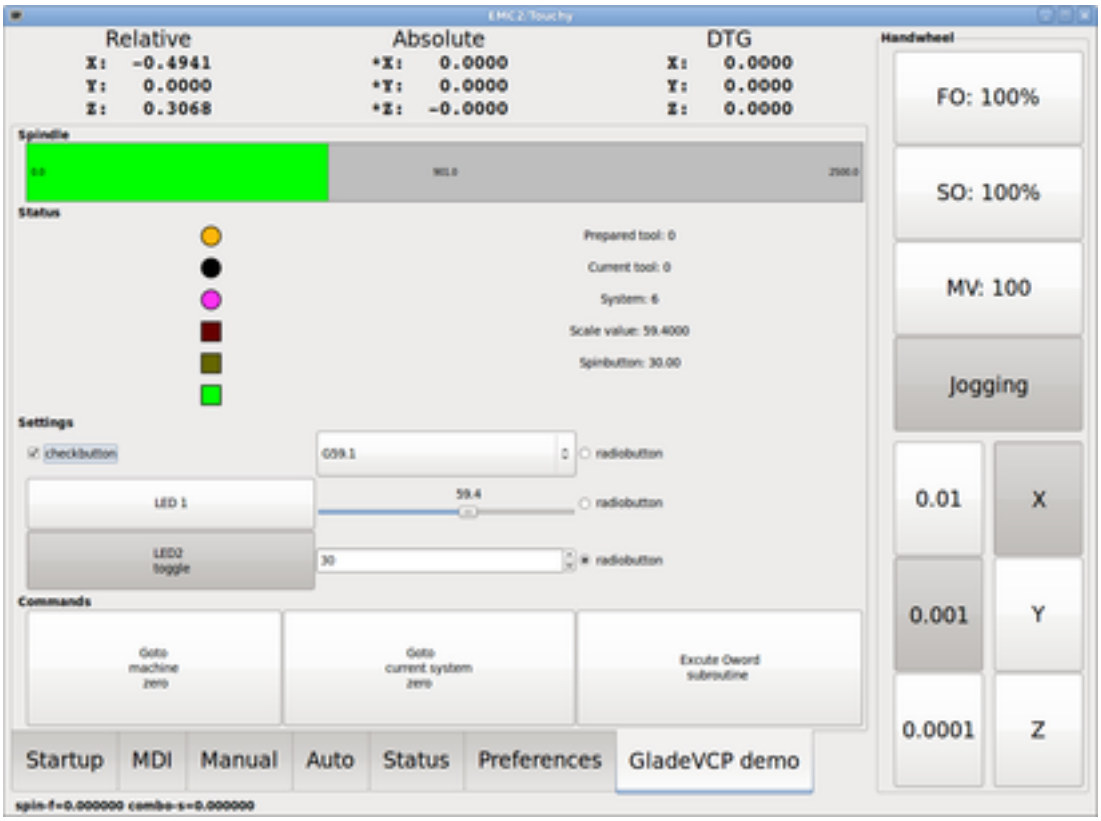
Pour lancer ce même panneau comme une fenêtre autonome à côté d'Axis, démarrer Axis en arrière plan puis démarrer gladevcp de la manière suivante:

```
$ cd configs/sim/gladevcp

$ linuxcnc axis.ini &

$ gladevcp -c gladevcp -u ../gladevcp/hitcounter.py -H
../gladevcp/manual-example.hal ../gladevcp/manual-example.ui
```



Pour lancer ce panneau dans *Touchy*:

```
$ cd configs/sim

$ linuxcnc gladevcp_touchy.ini
```



Fonctionnellement, ces configurations sont identiques. La seule différence porte sur l'état et la visibilité de l'écran. Puisqu'il est possible de lancer plusieurs composants GladeVCP en parallèle (avec des noms de modules de HAL différents), le mélange des configurations est également possible. Par exemple, un panneau sur le côté droit et un ou plusieurs en onglets pour des parties d'interface moins souvent utilisées.

11.2.1 Description de l'exemple de panneau

Pendant qu'Axis est en marche, explorons *Afficher configuration de HAL* dans lequel nous trouvons le composant de HAL *gladevcp* et dont nous pouvons observer la valeur des pins pendant l'interaction avec les widgets du panneau. La configuration de HAL peut être trouvée dans *configs/gladevcp/manual-example.hal*.

Usage des deux cadres en partie basse. Le panneau est configuré pour que, quand l'Arrêt d'Urgence est désactivé, le cadre *Settings* s'active et mette la machine en marche, ce qui active à son tour le cadre *Commandes* du dessous. Les widgets de HAL du cadre *Settings* sont liés aux Leds et labels du cadre *Status* ainsi qu'au numéros de l'outil courant et à celui de l'outil préparé. Les utiliser pour bien voir leur effet. L'exécution des commandes *T<numéro d'outil>* et *M6* dans la fenêtre du MDI aura pour effet de changer les numéros de l'outil courant et de l'outil préparé dans les champs respectifs.

Les boutons du cadre *Commandes* sont des *widgets d'action MDI*. Les presser exécutera une commande MDI dans l'interpréteur. Le troisième bouton *Execute Oword subroutine* est un exemple avancé, il prends plusieurs pins de HAL du cadre *Settings* et leur passe comme paramètres, le *sous-programme Oword*. Les paramètres actuels reçus par la routine sont affichés par une commande *(DEBUG,)*. Voir *configs/gladevcp/nc_files/oword.ngc* pour le corps du sous-programme.

Pour voir comment le panneau est intégré dans Axis, voir la déclaration de *[DISPLAY]GLADEVCP* dans gladevcp_panel.ui, ainsi que les déclarations de *[DISPLAY]EMBED* et de *[HAL]POSTGUI_HALFILE* dans *gladevcp_tab.ini*, respectivement.

11.2.2 Description de l'éditeur de Glade

L'interface utilisateur est créée avec l'éditeur graphique de Glade. Pour l'essayer il faut avoir le pré-requis nécessaire, que glade soit installé. Pour éditer l'interface utilisateur, lancer la commande:

```
$ glade configs/gladevcp/manual-example.ui
```

La zone centrale de la fenêtre montre l'apparence de l'interface en création. Tous les objets de l'interface et les objets supportés se trouvent dans la partie haute à droite de la fenêtre, où il est possible de choisir un widget spécifique (ou en cliquant sur lui au centre de la fenêtre). Les propriétés du widget choisi sont affichées et peuvent être modifiées, dans le bas à droite de la fenêtre.

Pour voir comment les commandes MDI sont passées depuis les widgets d'action MDI, explorer la liste des widgets sous *Actions* en haut à droite de la fenêtre, et dans le bas à droite de la fenêtre, sous l'onglet *Général*, les propriétés des *commandes MDI*.

11.2.3 Explorer la fonction de rappel de Python

Voici comment une fonction de rappel Python est intégrée dans l'exemple:
– Dans glade, regarder le label du widget `hits` (un widget GTK+).
– Dans le widget `button1`, regarder dans l'onglet *Signaux* et trouver le signal *pressed* associé avec le gestionnaire *on_button_press*.
– Dans ../gladevcp/hitcounter.py, regarder la méthode *on_button_press* et comment elle place la propriété du label dans l'objet *hits*.
C'était juste pour toucher le concept du doigt. Le mécanisme de fonction de rappel sera détaillé plus en détails dans la section Programmation de GladeVCP.

11.3 Créer et intégrer une interface utilisateur Glade

11.3.1 Pré-requis: Installation de Glade

Pour visualiser ou modifier les fichiers d'une interface Glade, Glade doit être installé. Ce n'est pas nécessaire pour seulement essayer un panneau GladeVCP. Si la commande *glade* est manquante, l'installer de la manière suivante:

```
$ sudo apt-get install glade
```

Vérifier ensuite la version installée, qui doit être égale ou supérieure à 3.6.7:

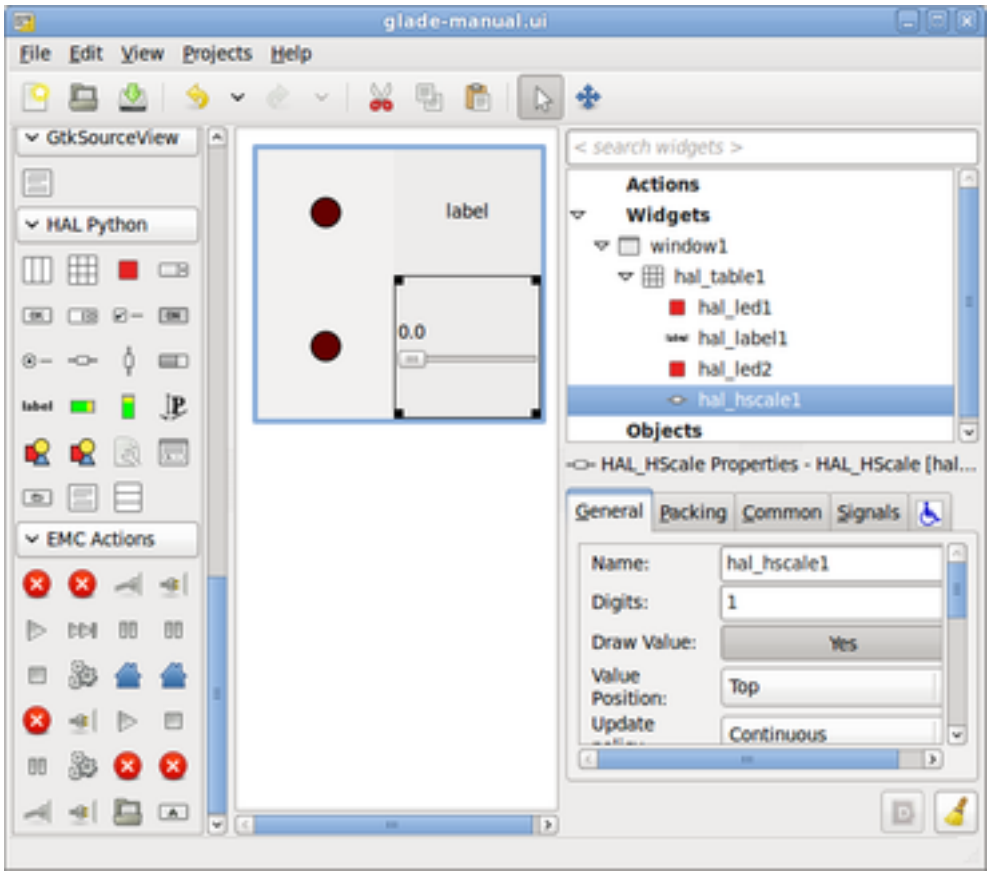
```
$ glade --version
```

```
glade3 3.6.7
```

11.3.2 Lancer Glade pour créer une nouvelle interface utilisateur

Cette section souligne juste les étapes initiales spécifiques à LinuxCNC. Pour plus d'informations et un tutoriel sur Glade, voir http://glade.gnome.org. Certains trucs & astuces sur Glade, peuvent aussi être trouvés sur youtube.

Soit modifier une interface existante en lançant `glade <fichier>.ui` ou, démarrer une nouvelle en lançant juste la commande `glade` depuis un terminal.
– Si LinuxCNC n'a pas été installé depuis un paquetage, l'environnement LinuxCNC du shell doit être configuré avec . *<linuxcncdir>/sc environment*, autrement Glade ne trouverait pas les widgets spécifiques à LinuxCNC.
– Quand l'éditeur demande pour enregistrer les préférences, accepter ce qui est proposé par défaut et presser *Close*.
– Depuis les *Niveaux supérieurs* (cadre de gauche), choisir *Fenêtre* (première icône) en haut des Niveaux supérieurs, par défaut cette fenêtre sera nommée *window1*. Ne pas changer ce nom, GladeVCP lui est relié.
– Dans le bas des onglets de gauche, dérouler *HAL Python* et *LinuxCNC Actions*.
– Ajouter au nouveau cadre, un conteneur comme une boîte HAL_Box ou une HAL_Table depuis *HAL Python*.
– Pointer et placer dans un conteneur d'autres éléments, comme une LED, un bouton, etc.
Le résultat pourrait ressembler à cela:



Glade a tendance à écrire beaucoup de messages dans la fenêtre du terminal, la plupart peuvent être ignorés. Sélectionner *Fichier → Enregistrer sous*, donner lui un nom comme *myui.ui* et bien vérifier qu'il sera enregistré comme un fichier *GtkBuilder* (bouton radio en bas à gauche du dialogue d'enregistrement). GladeVCP peut aussi traiter correctement l'ancien format *libglade* mais il n'y a aucune raison de l'utiliser. Par convention, l'extension des fichier GtkBuilder est *.ui*.

11.3.3 Tester un panneau

Vous êtes maintenant prêt à faire un essai (avec LinuxCNC, par exemple Axis en marche) faites:

```
gladevcp myui.ui
```

GladeVCP crée le composant de HAL portant le nom qui a été donné au fichier, par exemple, le très original *myui.ui* dans notre cas, à moins qu'il n'ait été surchargé pat l'option `-c <nom du composant>`. Si Axis est en marche, essayer de trouver le composant dans *Afficher configuration de HAL* et inspecter ses pins.

Vous vous demandez peut être pourquoi les widgets conteneurs comme *HAL_Hbox* ou *HAL_Table* apparaissent grisés (inactifs). Les conteneurs HAL ont une pin de HAL associée qui est désactivée par défaut, c'est ce qui cause ce rendu grisé des widgets conteneurs inactifs. Un cas d'utilisation courante pourrait être pour associer les pins de HAL du conteneur `halui.machine.is-on` ou un des signaux `halui.mode.`, pour s'assurer que certains widgets n'apparaissent actifs que dans un certain état.

Pour activer un conteneur, exécuter la commande HAL `setp gladevcp.<nom-du-conteneur> 1`.

11.3.4 Préparer le fichier de commande HAL

La voie suggérée pour lier les pins de HAL dans un panneau GladeVCP consiste à les collecter dans un fichier séparé portant l'extension `.hal`. Ce fichier est passé via l'option `POSTGUI_HALFILE=`, dans la section `[HAL]` du fichier de configuration.

ATTENTION: Ne pas ajouter le fichier de commandes HAL de GladeVCP à la section ini d'Axis `[HAL]HALFILE=`, ça n'aurait pas l'effet souhaité. Voir les sections suivantes.

11.3.5 Intégration dans Axis, comme pour PyVCP

Pour placer le panneau GladeVCP dans la partie droite d'Axis, ajouter les lignes suivantes dans le fichier ini:

```
[DISPLAY]
# ajouter le panneau GladeVCP à l'emplacement de PyVCP:
GLADEVCP= -u ../gladevcp/hitcounter.py ../gladevcp/manual-example.ui

[HAL]
# Les commandes HAL pour les composants GladeVCP dans un onglet, doivent êtreéé
excutes via POSTGUI_HALFILE
POSTGUI_HALFILE =  ../gladevcp/manual-example.hal

[RS274NGC]
# les sous-programmes Oword éspcifiques à gladevcp se placent ici
SUBROUTINE_PATH = ../gladevcp/nc_files/
```

Le nom de composant HAL d'une application GladeVCP lancé avec l'option GLADEVCP est toujours: `gladevcp`. La ligne de commande actuellement lancée par Axis dans la configuration ci-dessous est la suivante:

```
halcmd loadusr -Wn gladevcp gladevcp -c gladevcp -x {XID} <arguments pour GLADEVCP>
```

Ce qui veux dire que n'importe quelle option gladevcp, peut être ajoutée ici, tant qu'elle n'entre pas en collision avec les options des lignes de commande suivantes.

Note
L'option `[RS274NGC]SUBROUTINE_PATH=` est fixée seulement pour que l'exemple de panneau puisse trouver le sous-programme Oword pour le widget de commande MDI. Il n'est peut être pas nécessaire dans votre configuration.

11.3.6 Intégration dans un nouvel onglet d'Axis, à la suite des autres

Pour cela, éditer le fichier .ini et ajouter dans les sections DISPLAY et HAL, les lignes suivantes:

```
[DISPLAY]
# ajoute le panneau GladeVCP dans un nouvel onglet:
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]
# commandes HAL pour le composant GladeVCP dans un onglet doit être éexcut via
POSTGUI_HALFILE
POSTGUI_HALFILE =  ../gladevcp/manual-example.hal

[RS274NGC]
# les sous-programmes Oword éspcifiques à gladevcp se placent ici
SUBROUTINE_PATH = ../gladevcp/nc_files/
```

Noter le *halcmd loadusr* pour charger la commande d'onglet, elle assure que *POSTGUI_HALFILE* ne sera lancé que seulement après que le composant de HAL ne soit prêt. Dans de rares cas, une commande pourrait être lancée ici, pour utiliser un onglet sans être associée à un composant de HAL. Une telle commande pourrait être lancée sans *halcmd loadusr*, ce qui indiquerait à Axis qu'il ne doit plus attendre un composant de HAL, puisqu'il n'existe pas.

Noter que quand le nom du composant est changé dans l'exemple suivant, les noms utilisés dans `-Wn <composant>` et `-c <composant>` doivent être identiques.

Essayer en lançant Axis, il doit avoir un nouvel onglet appelé *GladeVCP demo* à droite de l'onglet de la visu. Sélectionner cet onglet, le panneau de l'exemple devrait être visible, bien intégré à Axis.

Note
Bien vérifier que le fichier de l'interface est la dernière option passée à GladeVCP dans les deux déclarations `GLADEVCP=` et `EMBED_TAB_COMMAND=`.

11.3.7 Intégration dans Touchy

Pour ajouter un onglet GladeVCP à *Touchy*, éditer le fichier .ini comme cela:

```
[DISPLAY]
# ajoute un panneau GladeVCP dans un onglet
EMBED_TAB_NAME=GladeVCP demo
EMBED_TAB_COMMAND=gladevcp -c gladevcp -x {XID} -u ../gladevcp/hitcounter.py -H
../gladevcp/gladevcp-touchy.hal ../gladevcp/manual-example.ui

[RS274NGC]
# les sous-programmes Oword éspcifiques à gladevcp se placent ici
SUBROUTINE_PATH = ../gladevcp/nc_files/
```

Noter les différences suivantes avec la configuration de l'onglet d'Axis:
– Le fichier de commandes HAL est légèrement modifié puisque *Touchy* n'utilise pas le composant *halui*, ses signaux ne sont donc pas disponibles et certains raccourcis ont été pris.
– Il n'y a pas d'option *POSTGUI_HALFILE=*, mais il est correct, de passer le fichier de commandes HAL, par la ligne *EMBED_TAB_COMMAND=*.
– L'appel *halcmd loaduser -Wn ...* n'est pas nécessaire.

11.4 Options de GladeVCP en ligne de commande

Voir également, *man gladevcp*. Ce sont les options pour cette ligne de commande:

Usage: gladevcp [options] myfile.ui

Options:

-h, --help
Affiche ce message d'aide et sort.

-c NAME
Fixe le nom du composant à NAME. Par défaut, le nom de base des fichiers UI

-d
Active la sortie débogage

-g GEOMETRY
Fixe la géométrie à WIDTHxHEIGHT+XOFFSET+YOFFSET. Les valeurs sont en pixels,
XOFFSET/YOFFSET est référencé à partir du coin haut, à gauche de l'écran.
Utilise -g WIDTHxHEIGHT pour fixer une taille ou -g +XOFFSET+YOFFSET pour fixer une position

-H FILE
exécute les déclarations de HAL depuis FILE, avec halcmd après que le composant soit chargé et prêt

-m MAXIMUM
force la fenêtre du panneau à se maximiser. Toutefois avec l'option -g geometry le panneau est déplaçable d'un moniteur à un autre en le forçant à utiliser toute l'écran

-t THEME
fixe le thème gtk. Par défaut, le thème système. Différents panneaux peuvent avoir différents thèmes. Un exemple de thème peut être trouvé sur le Wiki de LinuxCNC.

-x XID
Redonne un parent GladeVCP dans une fenêtre existante XID au lieu d'en créer une nouvelle au niveau supérieur

-u FILE
Utilise les FILE comme modules définis par l'utilisateur avec le gestionnaire

-U USEROPT
passe les modules python USEROPT

11.5 Références des Widgets HAL

GladeVcp inclus une collection de widgets Gtk qui ont des pins de HAL attachées, appelés widgets HAL, il sont destinés à contrôler, à afficher et à avoir d'autres interactions avec la couche HAL de LinuxCNC. Il sont destinés à être utilisés avec les interfaces créées par l'éditeur de Glade. Avec une installation correcte, les widgets HAL devraient être visibles, dans l'éditeur Glade, dans le groupe des Widgets *HAL Python*. Beaucoup de champs spécifiques à HAL dans l'onglet *Général* affichent une infobulle au survol de la souris.

Il y a deux variantes de signaux de HAL, bits et nombres. Les signaux bits sont les on/off. Les nombres peuvent être des "float", des "s32" ou des "u32". Pour plus d'informations sur les types de données de HAL, voir le manuel de HAL. Les widgets GladeVcp peuvent soit, afficher la valeur d'un signal avec un widget d'indication, soit, modifier la valeur d'un signal avec un widget de contrôle. Ainsi, il existe quatre classes de widgets gladvcp qui peuvent être connectés à un signal de HAL. Une autre classe de widgets d'aide permettent d'organiser et d'étiqueter les panneaux.
– Widgets d'indications "bit" signals: Led HAL
– Widgets de contrôle "bit" signals: HAL Bouton, HAL Bouton radio, HAL Case à cocher
– Widgets d'indications "nombre" signals: Section 11.5.7, HAL Barre de progression, HAL HBar, HAL VBar, HAL Indicateur
– Widgets de contrôle "nombre" signals: boîte d'incrément, HAL HScale, HAL VScale
– widgets d'aide: HAL Table, HAL HBox
– Tracé du parcours d'outil: HAL Gremlin
Les widgets HAL héritent des méthodes, propriétés et signaux des widgets Gtk sous-jacents, il est donc utile de consulter le site du GTK+ ainsi que la documentation pour les liaisons avec PyGTK.

11.5.1 Nommage des Widgets HAL et de leurs pins

La plupart des widgets HAL on une simple pin de HAL associée et portant le même nom que le widget (glade: Général→Nom).

Les exceptions à cette règle sont actuellement:
– *HAL_Spinbutton* et *HAL_ComboBox*, qui ont deux pins: une pin `<nomwidget>-f` (float) et une pin `<nomwidget>-s` (s32)
– *HAL_ProgressBar*, qui a une pin d'entrée `<nomwidget>-value`, et une pin d'entrée `<nomwidget>-scale`.

11.5.2 Donner des valeurs aux Widgets HAL et à leurs pins

En règle générale, si une valeur doit être attribuée à la sortie d'un widget HAL depuis un code Python, le faire en appelant le *setter* Gtk sous-jacent (par exemple `set_active()`, `set_value()`), ne pas essayer de donner directement la valeur à la pin associée par un `halcomp[nompin] = value`, parce-que le widget ne verra jamais le changement!.

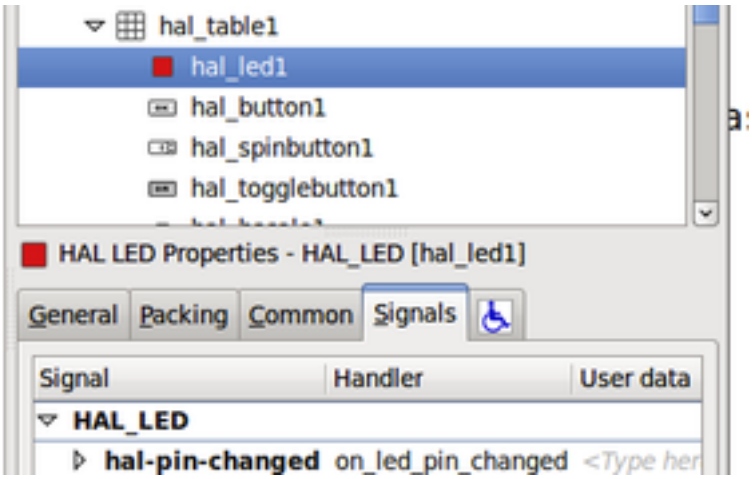
Il pourrait être tentant de *fixer une pin d'entrée de widget HAL* par programme. Noter que cela va à l'encontre du but premier d'une pin d'entrée. Elle devrait être attachée à un autre composant de HAL et réagir au signal qu'il génère. Bien qu'aucune protection, empêchant d'écrire sur les pins d'entrée HAL Python, ne soit présente actuellement, cela n'aurait aucun sens. Il faut utiliser `setp nompin valeur` dans un fichier Hal associé, pour les essais.

Il est par contre, parfaitement autorisé de mettre une valeur sur une pin de sortie de Hal avec `halcomp[nompin] = valeur` à condition que cette pin ne soit pas déjà associée avec un autre widget, ce qui aurait pu être créé par la méthode `hal_glib.GPin(halcomp.newpin(<nom>,<type>,<direction>)`. Voir la programmation de GladeVCP pour d'autres exemples.

11.5.3 Le signal *hal-pin-changed*

La programmation événementielle signifie que l'interface graphique indique au code quand "quelque chose se produit", grâce à une fonction de rappel, comme quand un bouton est pressé, la sortie du widget HAL (ceux qui affichent la valeur des pins de HAL) comme une LED, une barre, une VBar, un indicateur à aiguille etc, supportent le signal *hal-pin-changed* qui peut provoquer une fonction de rappel dans le code Python quand une pin de HAL change de valeur. Cela veut dire qu'il n'est plus nécessaire d'interroger en permanence les pins de HAL dans le code pour connaitre les changements, les widgets font ça en arrière plan et le font savoir.

Voici un exemple montrant comment régler un signal `hal-pin-changed` pour une Hal Led, dans l'éditeur de Glade:

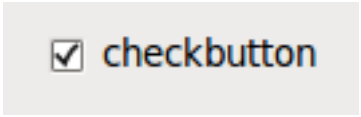


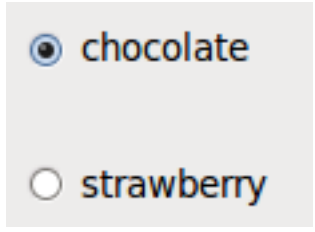
L'exemple dans `configs/gladevcp/examples/complex` montre comment c'est géré en Python.

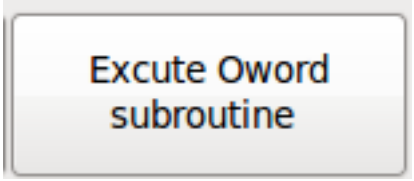
11.5.4 Les boutons (HAL Button)

Ce groupe de widgets est dérivé de divers boutons Gtk, ce sont les widgets HAL_Button, HAL_ToggleButton, HAL_RadioButton et CheckButton. Tous ont une seule pin de sortie BIT portant un nom identique au widget. Les boutons n'ont pas d'autres propriétés additionnelles, contrairement à leurs classes de base Gtk.

– HAL_Button: Action instantanée, ne retient pas l'état. Signal important: `pressed`.
– HAL_ToggleButton, HAL_CheckButton: Retiennent l'état on/off. Signal important: `toggled`.
– HAL_RadioButton: Un parmi un groupe. Signal important: `toggled` (par bouton).
– Importantes méthodes communes: `set_active()`, `get_active()`
– Importantes propriétés: `label`, `image`

Case à cocher: 

Boutons radio: 

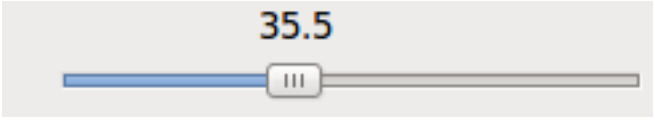
Bouton à bascule: 

ASTUCE
Définir les groupes de boutons radio dans Glade:
– Décider du bouton actif par défaut
– Dans les boutons radio, *Général→Groupe* sélectionner le nom du bouton actif par défaut dans le dialogue *Choisir un Bouton radio pour ce projet*.
Voir `configs/gladevcp/by-widget/radiobutton` pour une application GladeVCP avec un fichier d'interface utilisateur, pour travailler sur les boutons radio.

11.5.5 Les échelles (Scales)

HAL_HScale et HAL_VScale sont respectivement dérivées de GtkHScale et GtkVScale. Elles ont une pin de sortie FLOAT portant le même nom que le widget. Les échelles n'ont pas de propriété additionnelle.

Pour créer une échelle fonctionnelle dans Glade, ajouter un *Ajustement* (Général→Ajustement→Nouveau ou existant) et éditer l'objet ajustement. Il défini les valeurs défaut/min/max/incrément. Fixer la *Sensibilité de l'incrément* de l'ajustement sur automatique pour éviter les warnings.

Exemple d'échelle (HAL_hscale): 

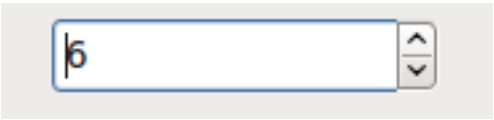
11.5.6 La boîte d'incrément (SpinButton)

La boîte d'incrément de HAL est dérivée de GtkSpinButton, elle a deux pins de sortie:

<nomwidget>-f
out FLOAT pin

<nomwidget>-s
out S32 pin

Pour être fonctionnelle, Spinbutton doit avoir une valeur d'ajustement comme l'échelle, vue précédemment.

Exemple de boîte d'incrément: 

11.5.7 Les labels

Le Label HAL est un simple widget basé sur GtkLabel qui représente la valeur d'une pin de HAL dans un format défini par l'utilisateur.

HAL pin type
Les pins de HAL sont des types (0:S32, 1:float ou 2:U32), voir aussi l'infobulle d'info sur *Général → HAL pin type*, (noter que c'est différent de PyVCP qui lui, a trois widgets label, un pour chaque type).

text template
Détermine le texte à afficher, une chaine au format Python pour convertir la valeur de la pin en texte. Par défauts, à `%s` (les valeurs sont converties par la fonction str()), mais peut contenir n'importe quel argument légal pour la méthode format() de Python. Exemple: `Distance: %.03f` va afficher le texte et la valeur de la pin avec 3 digits fractionnaires remplis avec des zéros pour une pin FLOAT.

11.5.8 Les conteneurs: HAL_HBox et HAL_Table

Comparés à leurs contreparties Gtk ils ont une pin d'entrée BIT qui contrôle si les enfants des widgets sont sensitifs ou non. Si la pin est basse, alors les widgets enfants sont inactifs, ce qui est le comportement par défaut.

ASTUCE
Si vous trouvez que certaines parties de votre application GladeVCP sont *grisées* (insensible), vérifiez que les pins d'un conteneur ne soient pas inutilisées.

11.5.9 Les Leds

La Led hal simule un vrai indicateur à Led. Elle a une seule pin d'entrée BIT qui contrôle son état: ON ou OFF. Les Leds ont quelques propriétés pour contrôler leur aspect:

on_color
: Une chaine définissant la couleur ON de la Led. Peut être tout nom valide de gtk.gdk.Color. Ne fonctionne pas sous Ubuntu 8.04.

off_color
: Un chaine définissant la couleur OFF de la Led. Peut être tout nom valide de gtk.gdk.Color ou la valeur spéciale *dark*. *dark* signifie que la couleur OFF sera fixée à 0.4 valeur de la couleur ON. Ne fonctionne pas sous Ubuntu 8.04.

pick_color_on, pick_color_off
: Couleurs pour les états ON et OFF peuvent être représentées par une chaine comme *#RRRRGGGGBBBB*. Ces propriétés optionnelles ont la précédence sur *on_color* et *off_color*.

led_size
: Rayon de la Led (pour une Led carrée, 1/2 côté)

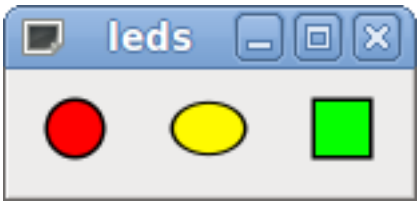
led_shape
: Forme de la Led Shape. Les valeurs permises sont 0 pour ronde, 1 pour ovale et 2 pour carrée.

led_blink_rate
: Si utilisée et que la Led est ON, alors la Led clignotera. La fréquence du clignotement est égal à la valeur de "led_blink_rate", spécifiée en millisecondes.

Comme un widget d'entrée, la Led aussi supporte le `hal-pin-changed signal`. Si vous voulez avoir une notification dans votre code quand les pins des Leds HAL ont changé d'état, alors connectez ce signal au gestionnaire, par exemple `on_led_pin_changed` et passez ce qui suit au gestionnaire:

```
def on_led_pin_changed(self,hal_led,data=None):
    print "on_led_pin_changed() - HAL pin value:",hal_led.hal_pin.get()
```

Ce code sera appelé à chaque front du signal et également au démarrage du programme pour reporter la valeur courante.

Exemple de Leds: 

11.5.10 La barre de progression (ProgressBar)

Note
Ce widget pourrait disparaître. Utilisez les widgets HAL_HBar et HAL_VBar à sa place.

La HAL_ProgressBar est dérivée de gtk.ProgressBar et a deux pins d'entrée de HAL float:

<nomwidget>
la valeur courante à afficher.

<nomwidget>-scale
la valeur maximum absolue en entrée.

Elle a les propriétés suivantes:

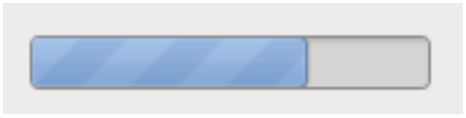
scale
Valeur d'échelle. fixe la valeur maximum absolue en entrée. Pareil que la configuration de la pin <nomwidget>.scale. Un flottant, compris entre -2^{24} et $+2^{24}$.

green_limit
Limite basse de la zone verte

yellow_limit
Limite basse de la zone jaune

red_limit
Limite basse de la zone rouge

text_template
Texte modèle pour afficher la valeur courante de la pin `<nomwidget>`. Formaté pour Python, peut être utilisé pour dict `{"valeur":valeur}`.

Exemple de barre de progression: 

11.5.11 La boîte combinée (ComboBox)

La comboBox HAL est dérivée de gtk.ComboBox. Elle valide le choix d'une valeur dans une liste déroulante.

Elle exporte deux pins de HAL:

<nomwidget>-f
La valeur courante, de type FLOAT

<nomwidget>-s
La valeur courante, de type S32

Elle a la propriété suivante, qui est configurable dans Glade:

column
L'index de colonne, type S32, défaut à -1, échelle de -1 à 100.

En mode par défaut, ces réglages du widget mettent les pins à la valeur d'index de l'entrée choisie dans la liste. Aussi, si le widget a trois labels, il peut seulement assumer les valeurs 0, 1 et 2.

En mode colonne (colonne > -1), la valeur reportée est choisie dans le tableau de stockage de liste défini dans Glade. Ainsi, typiquement la définition du widget devrait comprendre deux colonnes dans le tableau de stockage, une avec le texte affiché dans la liste déroulante, l'autre une valeur entière ou flottante correspondante au choix.

Il y a un exemple dans `configs/gladevcp/by-widget/combobox/combobox.{py,ui}` qui utilise le mode colonne pour prendre une valeur flottante dans un stockage de liste.

Si comme moi, vous êtes désorienté pour éditer une liste de stockage de ComboBox ou de CellRenderer, voyez http://www.youtube.com/-watch?v=Z5_F-rW2cL8.

11.5.12 Les barres

Les widgets HAL, HBar et VBar pour barres Horizontale et Verticale, représentent des valeurs flottantes. Elles ont une pin d'entrée de HAL FLOAT. Chaque barre a les propriétés suivantes:

invert
Inverse les directions min avec max. Une HBar inversée croît de la droite vers la gauche, un VBar inversée croît du haut vers le bas.

min, max
Valeurs minimum et maximum de l'étendue souhaitée. Ce n'est pas une erreur si la valeur courante dépasse cette étendue.

zero
Point le plus bas de l'étendue. Si il est entre min et max, alors la barre croît à partir de cette valeur et non de la gauche du widget (ou de sa droite). Utile pour représenter des valeurs qui peuvent être à la fois, positives ou négatives.

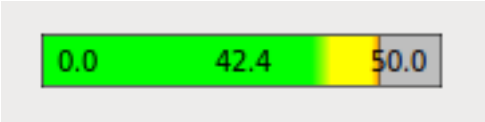
force_width, force_height
Force la largeur ou la hauteur du widget. Si inutilisés, la taille sera déduite du conteneur ou de la taille des widgets et des barres qui remplissent la zone.

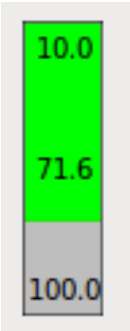
text_template
Détermine le texte à afficher, comme pour le Label, pour les valeurs min/max/courante. Peut être utilisé pour arrêter l'affichage de la valeur.

bg_color
Couleur de fond pour la barre (inactive).

z0_color, z1_color, z2_color
Couleurs des zones des différentes valeurs. Par défaut, *green*, *yellow* et *red*. Pour une description des zones voir propriétés des *z _border*.

z0_border, z1_border
Définissent les limites des zones de couleur. Par défaut, seule une zone est validée. Pour en activer plus d'une, fixer *z0_border* et *z1_border* aux valeurs souhaitées. Ainsi, zone 0 va remplir depuis 0 à la première bordure, zone 1 va remplir de la première à la seconde bordure et zone 2 depuis la dernière bordure jusqu'à 1. Les bordures se règlent comme des fractions, les valeurs vont de 0 à 1.

Barre horizontale: 

Barre verticale: 

11.5.13 L'indicateur (HAL Meter)

L'indicateur est un widget similaire à celui de PyVCP, il représente une valeur flottante et a une pin d'entrée de HAL FLOAT. L'indicateur a les deux propriétés suivantes:

min, max
Valeurs minimum et maximum de l'étendue souhaitée. Ce n'est pas une erreur si la valeur courante dépasse cette étendue.

force_size
Force le diamètre du widget. Si inutilisé, alors la taille sera déduite du conteneur ou des dimensions d'un widget à taille fixe. L'indicateur occupera alors l'espace le plus grand disponible, tout en respectant les proportions.

text_template
Détermine le texte à afficher, comme pour le Label, pour la valeur courante. Peut être utilisé pour arrêter l'affichage de la valeur.

label
Label large au dessus du centre de l'indicateur.

sublabel
Petit label, sous le centre de l'indicateur.

bg_color
Couleur de fond de l'indicateur.

z0_color, z1_color, z2_color
Valeurs des couleurs des différentes zones. Par défaut, *green*, *yellow* et *red*. For description of zones see *z _border* properties.

z0_border, z1_border
Définissent les limites externes des zones de couleur. Par défaut, une seule zone de couleur est définie. Pour en activer plus d'une, fixer *z0_border* et *z1_border* aux valeurs souhaitées. Ainsi, zone 0 va remplir depuis min à la première bordure, zone 1 va remplir de la première à la seconde bordure et zone 2 depuis la dernière bordure jusqu'à max. Les bordures se règlent sur une étendue comprise en min et max.

Exemples d'indicateurs:

11.5.14 Gremlin, visualiseur de parcours d'outil pour fichiers .ngc

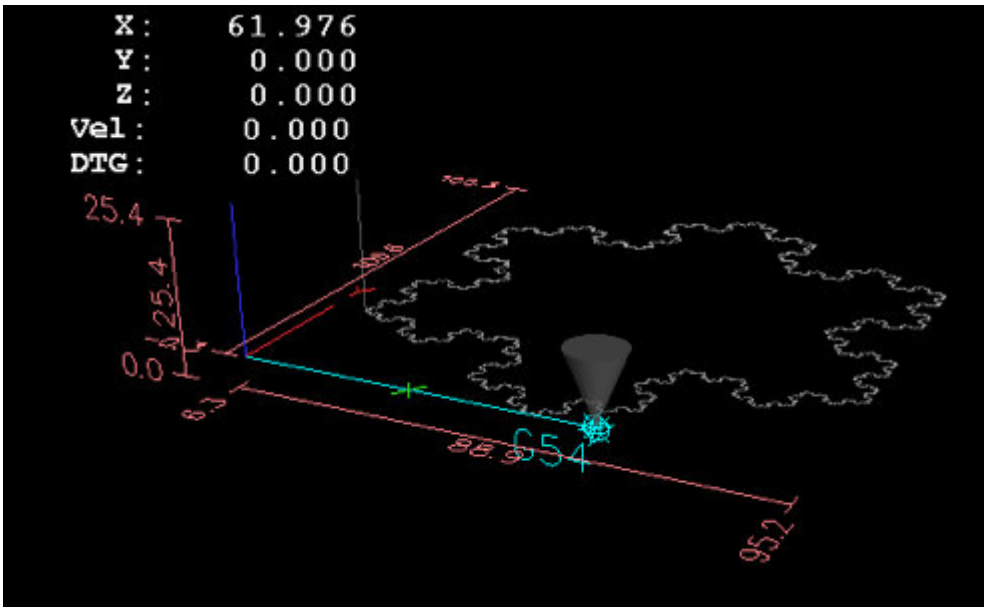
Gremlin est un traceur de parcours d'outil similaire à celui d'Axis. Il demande un environnement LinuxCNC en fonctionnement, comme Axis ou Touchy. Pour se connecter à lui, inspecter la variable d'environnement INI_FILE_NAME. Gremlin affiche le fichiers .ngc courant. Si le fichier ngc est modifié, il doit être rechargé pour actualiser le tracé. Si il est lancé dans une application GladeVCP quand LinuxCNC n'est pas en marche, un message va être affiché parce-que le widget Gremlin ne trouve pas le statut de LinuxCNC, comme le nom du fichier courant.

Gremlin n'exporte aucune pin de HAL. Il a les propriétés suivantes:

view
Peut être la vue en *x*, *y*, *z*, *p* (perspective) . Par défaut, vue en *z*.

enable_dro
Booléen; afficher une visu sur le tracé ou non. Par défaut,à *True*.

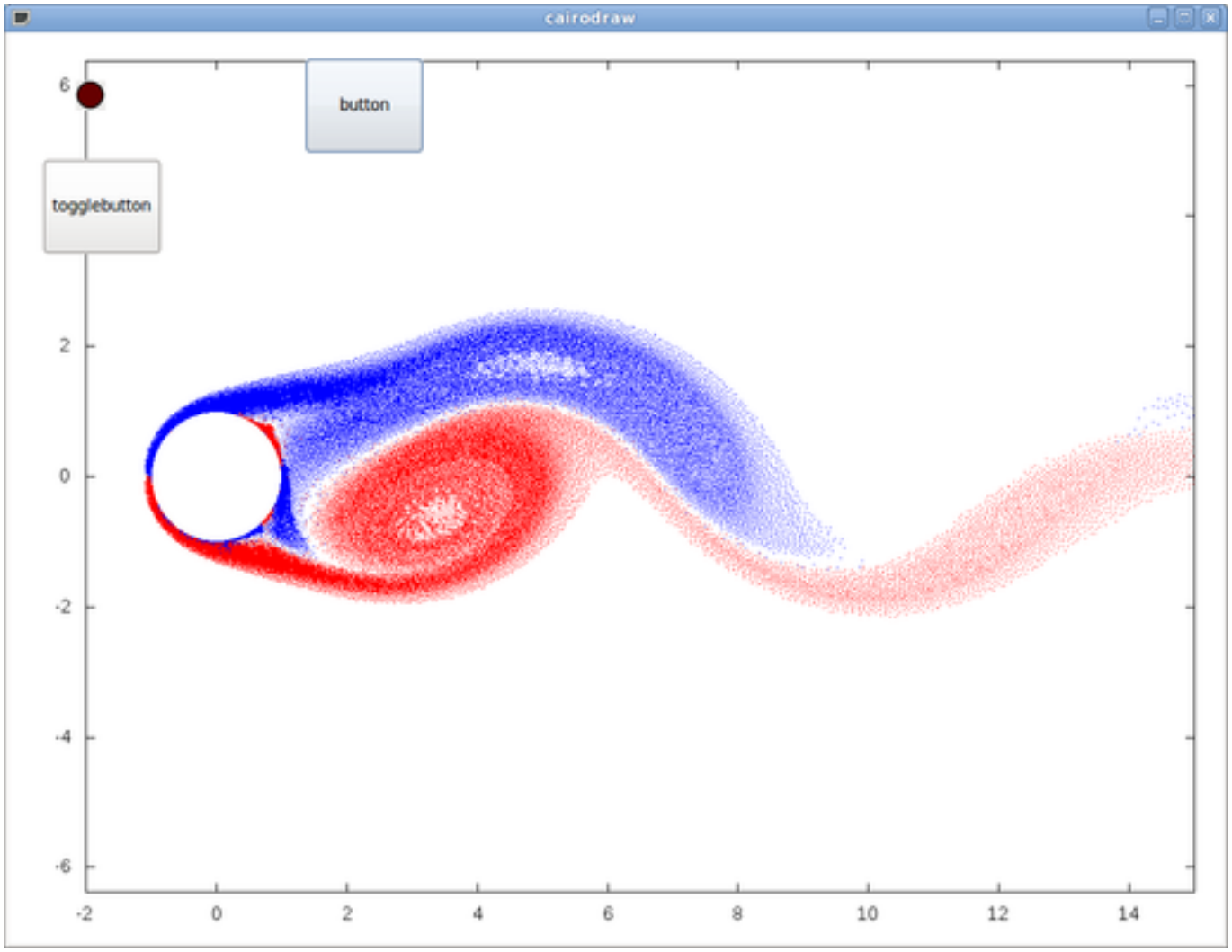
Exemple:



11.5.15 Fonction de diagrammes animés: Widgets HAL dans un bitmap

Pour certaines applications, il est intéressant d'avoir une image de fond, comme un diagramme fonctionnel et positionner les widgets aux endroits appropriés dans le diagramme. Une bonne combinaison consiste à placer une image de fond comme un fichier .png, mettre la fenêtre GladeVCP en taille fixe, et utiliser Glade pour fixer la position du widget sur cette image.

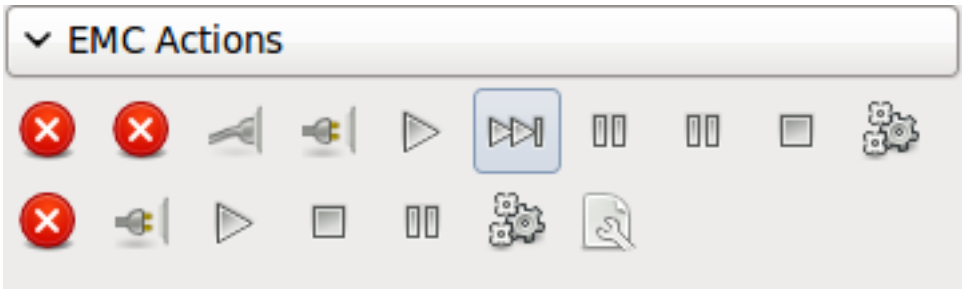
Le code pour l'exemple ci-dessus peut être trouvé dans `configs/gladevcp/animated-backdrop`:



11.6 Références des Widgets LinuxCNC Action

GladeVcp inclus une collection d'actions préprogrammées appelées widgets *LinuxCNC Action* qui sont des Widgets pour l'éditeur Glade. À la différence des widgets HAL, qui interagissent avec les pins de HAL, les widgets LinuxCNC Actions, interagissent avec LinuxCNC et son interpréteur de G-code.

Les widgets LinuxCNC Action sont dérivés du widget Gtk.Action. Le widget LinuxCNC Action en quelques mots:
– C'est un objet disponible dans l'éditeur Glade.
– Il n'a pas d'apparence visuelle par lui-même.
– Son but: associer à un composant d'interface visible, à un composant d'interface sensitif, comme un menu, un bouton outil, un bouton avec une commande. Voir les propriétés des widgets Action dans *Général* → *Related Action* de l'éditeur.
– L'action préprogrammée sera exécutée quand l'état du composant associé basculera (bouton pressé, menu cliqué…)
– Ils fournissent une voie facile pour exécuter des commandes sans avoir à faire appel à la programmation en Python.
L'apparence des LinuxCNC Actions dans Glade est approximativement la suivante:



Le survol de la souris donne une infobulle.

11.6.1 Les widgets LinuxCNC Action

Les widgets LinuxCNC Action sont des widgets de type simple état. Ils implémentent une seule action par l'usage, d'un seul bouton, d'une option de menu, d'un bouton radio ou d'une case à cocher.

11.6.2 Les widgets LinuxCNC bascule action (ToggleAction)

Ce sont des widgets double état. Ils implémentent deux actions ou utilisent un second état (habituellement, *pressé*) pour indiquer qu'une action est actuellement en cours. Les bascules action sont prévues pour être utilisées avec les boutons à bascule (ToggleButtons) et les boutons à bascule d'outil (ToggleToolButtons) ou encore, pour basculer les items de menu. Un exemple simple est le bouton à bascule d'Arrêt d'Urgence (EStop).

Actuellement, les widgets suivants sont disponibles:
– La bascule *d'Arrêt d'Urgence* (ESTOP) envoie la commande ESTOP ou ESTOP_RESET à LinuxCNC, selon l'état courant.
– La bascule *ON/OFF* envoie la commande STATE_ON ou STATE_OFF.
– La bascule *Pause/Reprise* envoie la commande AUTO_PAUSE ou AUTO_RESUME.
Les bascules action suivantes ont seulement une commande associée et utilisent l'état *pressé* pour indiquer que l'opération demandée est lancée:
– La bascule *Run* envoie la commande AUTO_RUN et attends dans l'état pressé jusqu'à ce que l'interpréteur soit de nouveau au repos.
– La bascule *Stop* est inactive jusqu'à ce que l'interpréteur passe à l'état actif (Un G-code est lancé) et permet alors à l'utilisateur d'envoyer la commande AUTO_ABORT.
– La bascule *MDI* envoie la commande passée dans le MDI et attends sa complétion dans l'état inactif *pressé*.

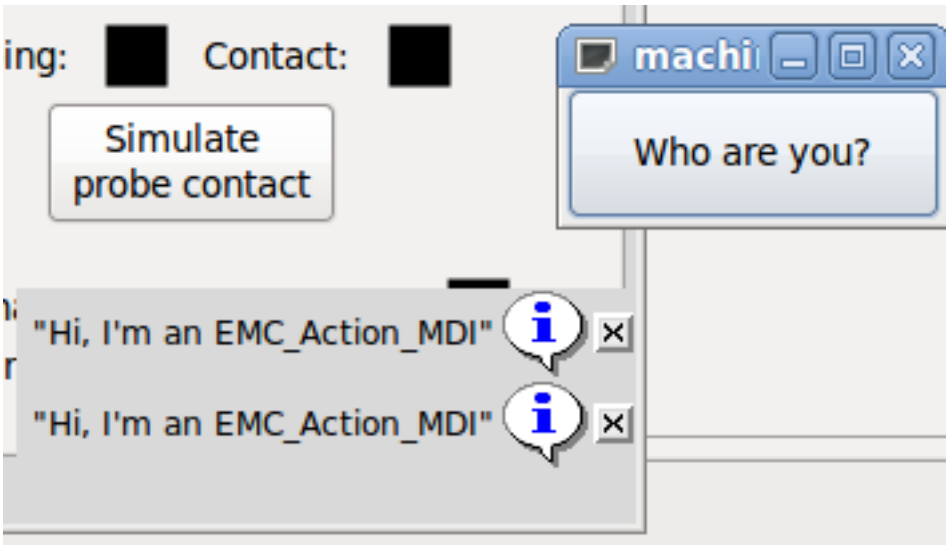
11.6.3 La bascule Action_MDI et les widgets Action_MDI

Ces widgets fournissent le moyen d'exécuter des commandes MDI. Le widget Action_MDI n'attends pas la complétion de la commande, comme le fait la bascule Action_MDI, qui reste elle, désactivée tant que la commande n'est pas terminée.

11.6.4 Un exemple simple: Exécuter une commande MDI lors de l'appui sur un bouton.

`configs/gladevcp/mdi-command-example/whoareyou.ui` est un fichier UI Glade qui transmet cette action basique:

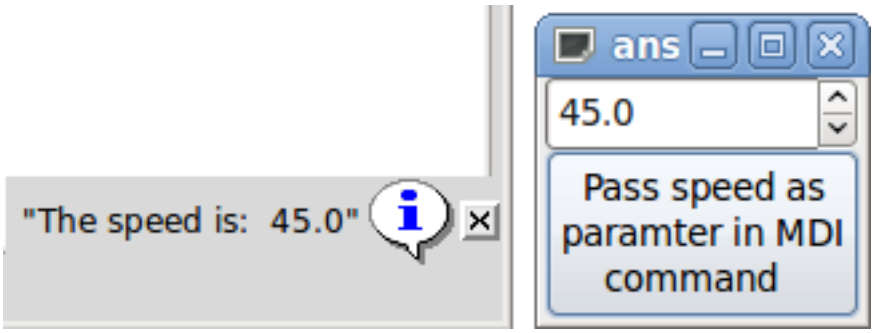
L'ouvrir dans Glade et étudier comment il est fait. Lancer Axis puis dans un terminal faire: +*gladevcp whoareyou.ui*+. Voir l'action `hal_action_mdi1` et les propriétés de `MDI command` qui exécute juste `(MSG, "Hi, I'm an LinuxCNC_Action_MDI` ce qui ouvre un popup de message dans Axis, comme ci-dessous:



Noter que le bouton, associé à l'Action_MDI, est grisé si la machine est arrêtée, en A/U ou si l'interpréteur est déjà en marche. Il deviendra automatiquement actif quand la machine sera mise en marche donc, sortie de l'A/U (E-Stop), et que le programme est au repos.

11.6.5 Paramètres passés avec les widgets Action_MDI et ToggleAction_MDI

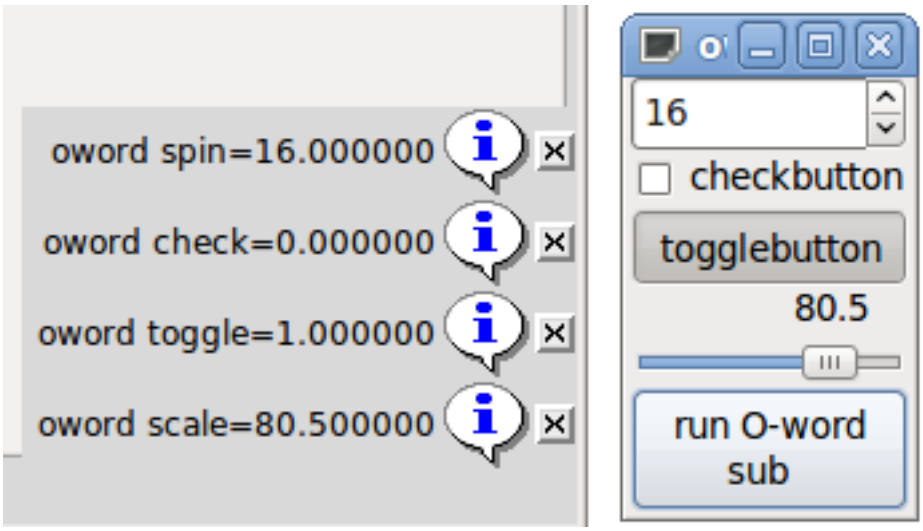
Optionnellement, la chaine *MDI command* peut avoir des paramètres substitués avant d'être passée à l'interpréteur. Ces paramètres sont actuellement les noms des pins de HAL dans les composants GladeVCP. Voici comment cela fonctionne:
– Supposons que nous avons une *SpinBox HAL* nommée `speed`, nous voulons passer sa valeur courante comme paramètre dans une commande MDI.
– La SpinBox HAL aura une pin de HAL de type flottant, nommée speed-f (voir la description des Widgets Hal).
– Pour substituer cette valeur dans la commande MDI, insérons le nom de la pin de HAL
– Pour la spinbox HAL précédente, il aurait été possible d'utiliser
L'exemple de fichier UI est `configs/gladevcp/mdi-command-example/speed.ui`. Voici ce qui ce qui est obtenu en le lançant:



11.6.6 Un exemple plus avancé: Passer des paramètres à un sous-programme O-word

Il est parfaitement permis d'appeler un sous-programme O-word dans une commande MDI et passer la valeur des pins de HAL comme paramètres actuels. Un exemple de fichier UI est dans `configs/gladevcp/mdi-command-example/owordsub.ui`.

Placer `configs/gladevcp/nc_files/oword.ngc` de sorte qu'Axis puisse le trouver, et lancer *gladevcp owordsub.ui* depuis un terminal. Ce qui devrait ressembler à celà:



11.6.7 Préparation d'une Action_MDI

L'interpréteur de G-code de LinuxCNC dispose d'un simple jeu de variables globales, comme la vitesse travail, la vitesse broche, le mode relatif/absolu et autres. Si on utilise des commandes G-code ou des sous-programmes O-word, certaines de ces variables doivent être modifiées par la commande ou le sous-programme. Par exemple, un sous-programme de sonde a très probablement besoin de définir la vitesse d'avance à une valeur très faible. Sans autres précautions, le réglage de vitesse précédent serait écrasé par la valeur du sous-programme de sonde.

Pour faire avec ce surprenant, autant qu'indésirable effet de bord produit par un sous-programme O-word ou un G-code exécuté avec une bascule Action MDI, le gestionnaire pré-MDI et post-MDI doit être associé avec une bascule Action_MDI donnée. Ces gestionnaires sont optionnels et fournissent une voie pour sauver tous les états avant d'exécuter l'action MDI et pour les restaurer ensuite aux valeurs précédentes. Les noms de signaux sont `mdi-command-start` et `mdi-command-stop`, les noms de gestionnaire peuvent être fixés dans Glade comme tout autre gestionnaire.

Voici un exemple, montrant comment la valeur de la vitesse d'avance est sauvée puis restaurée par de tels gestionnaires, noter que la commande LinuxCNC et le statut des voies sont disponibles comme `self.emc` et `self.stat` à travers la classe LinuxCNC_ActionBase:

```
    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.emc.mdi('F%.1f' % (self.start_feed))
        while action.emc.wait_complete() == -1:
            pass
```

Seule le widget de la bascule Action_MDI, supporte ces signaux.

Note
Dans une prochaine version de LinuxCNC, les nouveaux M-codes M70 à M72 seront disponibles, ils enregistreront l'état avant l'appel du sous-programme, la restauration de l'état au retour sera plus aisée.

11.6.8 Utiliser l'objet LinuxCNC Stat pour traiter les changements de statut

Beaucoup d'actions dépendent du statut de LinuxCNC, est-il en mode manuel, en mode MDI ou en mode auto ? Un programme est-il en cours d'exécution, est-il en pause ou au repos ? Il est impossible de lancer une commande MDI tant qu'un programme G-code est en cours d'exécution, cela doit donc être pris en compte. Beaucoup d'actions LinuxCNC prennent cela en compte d'elle même, les boutons et les options de menu sont désactivés quand leurs actions sont rendues impossibles.

Avec l'utilisation des gestionnaires d'événements Python, qui sont à un niveau inférieur aux Actions, on doit prendre soin de traiter les dépendances de statut soit-même. À cette fin, existe le widget *LinuxCNC Stat*, il associe les changements de statut de LinuxCNC avec les gestionnaires d'événements.

LinuxCNC Stat n'a pas de composant visible, il suffi de l'ajouter dans l'éditeur Glade. Une fois ajouté, vous pouvez associer des gestionnaires avec les signaux suivants:
– relatif au statut: émis quand l'arrêt d'urgence est activé, ou désactivé,
  – `state-estop` la machine est totalement arrêtée, puissance coupée.
  – `state-estop-reset` la machine passe à l'arrêt.
  – `state-on`, la machine est mise en marche
  – `state-off` la machine passe à l'arrêt.
– relatif au mode: émis quand LinuxCNC entre dans un de ces modes particuliers
  – `mode-manual`
  – `mode-mdi`
  – `mode-auto`
– relatif à l'interpréteur: émis quand l'interpréteur de G-code passe dans un de ces modes
  – `interp-run`
  – `interp-idle`
  – `interp-paused`
  – `interp-reading`
  – `interp-waiting`

11.7 Programmation de GladeVCP

11.7.1 Actions définies par l'utilisateur

La plupart des jeux de widgets, par le biais de l'éditeur Glade, supportent le concept de fonction de rappel, fonctions écrites par l'utilisateur, qui sont exécutées quand *quelque chose arrive* dans l'UI, événements tels que clics de souris, caractère tapé, mouvement de souris, événements d'horloge, fenêtre iconisée ou agrandie et ainsi de suite.

Les widgets de sortie HAL, typiquement, scrutent les événements de type *entrée*, tels qu'un bouton pressé, provoquant un changement de la valeur d'une pin HAL associée par le biais d'une telle fonction de rappel prédéfinie. Dans PyVCP, c'est réellement le seul type d'événement qui peut être défini à la main. Faire quelque chose de plus complexe, comme exécuter une commande MDI pour appeler un sous-programme G-code, n'est pas supporté.

Dans GladeVCP, les changement sur les pins de HAL sont juste un type de la classe générale d'événements (appelés signaux) dans GTK+. La plupart des widgets peuvent générer de tels signaux et l'éditeur de Glade supporte l'association de ces signaux avec une méthode Python ou nom de fonction.

Si vous décidez d'utiliser les actions définies par l'utilisateur, votre travail consistera à écrire un module Python dont la méthode, une fonction suffit dans les cas simples, peut être référencée à un gestionnaire d'événements dans Glade. GladeVCP fournit un moyen d'importer votre module au démarrage, il sera alors lié automatiquement au gestionnaire d'événements avec les signaux de widget comme un ensemble dans la description de l'éditeur Glade.

11.7.2 Un exemple: ajouter une fonction de rappel en Python

Ceci est juste un exemple minimal pour exprimer l'idée, les détails sont donnés dans le reste de cette section.

GladeVCP peut, non seulement manipuler ou afficher les pins de HAL, il est possible aussi d'écrire des gestionnaires d'événements en Python. Ce qui peut être utilisé, entre autre, pour exécuter des commandes MDI. Voici comment faire:

Écrire un module Python comme le suivant, et l'enregistrer sous le nom handlers.py

```
nhits = 0
def on_button_press(gtkobj,data=None):
    global nhits nhits += 1 gtkobj.set_label("hits: %d" % nhits)
```

Dans Glade, définir un bouton ou un bouton HAL, sélectionner l'onglet *Signal*, et dans les propriétés GtkButton sélectionner la ligne *pressed*. Entrer *on_button_press* ici, puis enregistrer le fichier Glade.

Ensuite, ajouter l'option *-u handlers.py* à la ligne de commande de gladevcp. Si les gestionnaires d'événements son répartis sur plusieurs fichiers, ajouter de multiples options *-u <pynomfichier>*.

Maintenant, presser le bouton devrait modifier son label car il est défini dans la fonction de rappel.

Que fait le drapeau `-u`: toutes les fonctions Python dans ce fichier sont collectées et configurées comme des gestionnaires de fonction de rappel potentiels pour les widgets Gtk, ils peuvent être référencés depuis l'onglet *Signaux* de Glade. Le gestionnaire de fonction de rappel est appelé avec l'instance de l'objet particulier comme paramètre, comme l'instance du GtkButton précédente, ainsi, il est possible d'appliquer n'importe quelle méthode GtkButton depuis ici.

Ou faire des choses plus utiles, par exemple, appeler une commande MDI!

11.7.3 L'événement valeur de HAL modifiée

Les widgets d'entrée HAL, comme la Led, ont l'état de leur pin de HAL (on/off), automatiquement associé avec l'apparence optique du widget (Led allumée/éteinte).

Au delà de cette fonctionnalité primitive, on peut associer n'importe quelle pin de HAL avec une fonction de rappel, y compris les widgets de HAL prédéfinis. Cela correspond bien avec la structure événementielle de l'application typique du widget: chaque activité, qu'elle soit un simple clic de souris, une touche pressée, une horloge expirée ou le changement de valeur d'une pin de HAL, générera une fonction de rappel et sera gérée par le même mécanisme.

Pour les pins de HAL définies par l'utilisateur, non associées à un widget de HAL particulier, le nom du signal est *value-changed*. Voir la section Ajouter des pins de HAL pour plus de détails.

Les widgets HAL sont fournis avec un signal prédéfini appelé *hal-pin-changed*. Voir la section sur les Widgets HAL pour d'autres détails.

11.7.4 Modèle de programmation

L'approche globale est la suivante:
– Concevoir l'interface graphique avec Glade, fixer les gestionnaires de signaux associés aux widgets action.
– Écrire un module Python qui contient des objets appelables (voir 'gestionnaire de modèles, plus loin)
– Passer le chemin du modules à gladevcp avec l'option *-u <module>*.
– gladevcp importe le module, inspecte les gestionnaires de signaux et les connecte à l'arbre des widgets.
– La boucle principale d'événements est exécutée.

11.7.4.1 Modèle du gestionnaire simple

Pour des tâches simple, il est suffisant de définir des fonctions nommées après les gestionnaires de signaux de Glade. Elles seront appelées quand l'événement correspondant se produira dans l'arbre des widgets. Voici un exemple très simple, il suppose que le signal *pressed* d'un bouton Gtk ou d'un bouton HAL est lié à une fonction de rappel appelée *on_button_press*:

```
nhits = 0
def on_button_press(gtkobj,data=None):
    global nhits
    nhits += 1
    gtkobj.set_label("hits: %d" % nhits)
```

Ajouter cette fonction dans un fichier Python et le lancer avec:

```
gladevcp -u <myhandler>.py mygui.ui
```

Noter que la communication entre les gestionnaires doit passer par des variables globales, qui s'adaptent mal est ne sont pas très "pythonique". C'est pourquoi nous en arrivons au gestionnaire de classes.

11.7.4.2 Modèle de gestionnaire basé sur les classes

L'idée ici est la suivante: les gestionnaires sont liés aux méthodes de classe. La classe sous-jacente est instanciée et inspectée durant le démarrage de GladeVCP et liée à l'arbre des widgets comme gestionnaire de signaux. Donc, la tâche est maintenant d'écrire:
– Une ou plusieurs définitions de classe avec une ou plusieurs méthodes, dans un module ou répartis sur plusieurs modules.
– Une fonction *get_handlers* dans chaque module, qui retournera la liste des instances de classe à GladeVCP, leurs noms de méthode seront liés aux gestionnaires de signaux.
Voici un exemple minimaliste de module de gestionnaire définit par l'utilisateur:

```
class MyCallbacks :
    def on_this_signal(self,obj,data=None):
        print "this_signal happened, obj=",obj
    def get_handlers(halcomp,builder,useropts):
        return [MyCallbacks ()]
```

Maintenant, *on_this_signal* est disponible comme gestionnaire de signal dans l'arbre des widgets.

11.7.4.3 Le protocole get_handlers

Si durant l'inspection du module GladeVCP trouve une fonction *get_handlers*, Il l'appelle de la manière suivante:

```
get_handlers(halcomp,builder,useropts)
```

Les arguments sont:
– halcomp - Se réfère au composant de HAL en construction.
– builder - arbre du widget - résulte de la lecture de la définition de l'UI (soit, en référence à un objet de type GtkBuilder ou de type libglade).
– useropts - Une liste de chaines collectée par l'option de la ligne de commande de gladevcp *-U <useropts>*.
GladeVCP inspecte alors la liste des instances de classe et récupère leurs noms. Les noms de méthode sont connectés à l'arbre des widgets comme gestionnaire de signaux. Seuls, les noms de méthode ne commençant pas par un _ (tiret bas) sont considérés.

Noter que peu importe si la libglade ou le nouveau format GtkBuilder est utilisé pour l'UI Glade, les widgets peuvent toujours être soumis au *builder.get_object(<nomwidget>)*. En outre, la liste complète des widgets est disponible par *builder.get_objects()*, indépendamment du format de l'UI.

11.7.5 Séquence d'initialisation

Il est important de connaitre pour quoi faire, la fonction *get_handlers()* est appelée, et connaitre ce qui est sûr et ce qui ne l'est pas. Tout d'abord, les modules sont importés et initialisés dans leur ordre d'apparition sur la ligne de commande. Après le succès de l'importation, *get_handlers()* est appelé selon les étapes suivantes:
– L'arbre du widget est créé, mais pas encore réalisé (pas tant que le niveau supérieur *window.show()* n'aura pas été exécuté)
– Le composant de HAL, halcomp, est configuré et toutes les pins de HAL des widgets lui sont ajoutées.
– Il est sûr d'ajouter plus de pins de HAL parce-que *halcomp.ready()* n'a pas encore été appelé à ce point, ainsi, on peut ajouter ses propres pins, par exemple, dans la méthode de classe init*()*.
Après que tous les modules ont été importés et que les noms des méthodes ont été extraits, les étapes suivantes se produisent:
– Tous les noms de méthode qualifiés seront connectés à l'arbre du widget avec *connect_signals() ou signal_autoconnect()* (selon le type de l'UI importée, format GtkBuilder ou l'ancien libglade).
– Le composant de HAL est finalisé avec halcomp.ready().
– Si un ID de fenêtre est passé comme argument, l'arbre du widget est re-apparenté pour démarrer dans cette fenêtre, et la fenêtre de niveau supérieur de Glade, window1 est abandonnée (voir la FAQ)
– Si un fichier de commandes de HAL, est passé avec *-H halfile*, il est exécuté avec halcmd.
– La boucle principal de Gtk est lancée.
Ainsi, lorsque le gestionnaire de classe est initialisé, tous les widgets sont existants mais pas encore réalisés (affichés à l'écran). Et le composant de HAL n'est pas prêt non plus, de sorte qu'il n'est pas sûr d'accéder aux valeurs des pins dans la méthode init*()*.

Si on doit avoir une fonction de rappel à exécuter au démarrage du programme mais, après qu'il soit sûr d'accéder aux pins de HAL, alors connecter un gestionnaire au signal de la fenêtre de niveau supérieur réalisée, window1 (qui pourrait être sa seule raison d'être). A ce point, GladeVCP en a terminé avec toutes les configurations, le halfile a bien été lancé et GladeVCP est sur le point d'entrer dans la boucle principale Gtk.

11.7.6 Multiple fonctions de rappel avec le même nom

Dans une classe, les noms de méthode doivent être unique. Cependant, il est permis d'avoir de multiples instances de classe passées à GladeVCP par get_handlers() avec des méthodes portant le même nom. Lorsque le signal correspondant survient, les méthodes sont appelées dans l'ordre dans lequel elles ont été définies, module par module et dans un module, dans l'ordre des instances de classe retourné *get_handlers()*.

11.7.7 Le drapeau GladeVCP -U <useropts>

Au lieu d'étendre GladeVCP à toutes les options concevables qui pourraient potentiellement être utilisées par un gestionnaire de classe, on peut utiliser le drapeau -U<useroption> (répétitivement si nécessaire). Ce drapeau collecte la liste des chaines de <useroption>. Cette liste est passée à la fonction get_handlers() (argument useropts). Le code est libre d'interpréter ces chaines comme bon lui semble. Un utilisation possible serait de les passer à la fonction exec de Python dans le *get_handlers()*, comme suit:

```
debug = 0
...
def get_handlers(halcomp,builder,useropts):
    ...
    global debug # suppose qu'il y a une variable globale
    pour cmd dans useropts:
        exec cmd in globals()
```

De cette façon, on peut passer des déclarations Python arbitraires au module grâce à l'option gladevcp -U, Par exemple:

```
gladevcp -U debug=42 -U "print 'debug=%d' % debug" ...
```

Debug devrait être mis à 2, et confirmer ce que le module fait actuellement.

11.7.8 Variables persistantes dans GladeVCP

Un aspect gênant de GladeVCP dans sa forme initiale avec pyvcp est le fait qu'on peut changer les valeurs des pins de HAL au travers du texte saisi, curseurs, bouton tournant, bouton à bascule etc, mais leurs paramètres ne sont pas enregistrés ni restaurés à la prochaine exécution de LinuxCNC. Ils commencent aux valeurs par défaut fixées dans le panneau ou la définition du widget.

GladeVCP dispose d'un mécanisme facile à utiliser pour enregistrer et restaurer l'état des widgets de HAL, ainsi que les variables du programme (en fait, n'importe quel attribut d'instance de type int, float, bool ou string).

Ce mécanisme utilise le format du populaire fichier *.ini* pour enregistrer et recharger les attributs persistants.

11.7.8.1 Examen de la persistance, de la version et de la signature du programme

Imaginons renommer, ajouter ou supprimer des widgets dans Glade: un fichier .ini qui traîne depuis une version précédente du programme, ou une interface utilisateur entièrement différente, ne serait pas en mesure de restaurer correctement l'état des variables et des types puisqu'ils ont changé depuis.

GladeVCP détecte cette situation par la signature qui dépends de tous les noms d'objets et de types qui ont été enregistrés et qui doivent être restaurés. Dans le cas de signatures incompatibles, un nouveau fichier .ini avec la configuration pas défaut est généré.

11.7.9 Utilisation des variables persistantes

Pour que tous les états des widgets Gtk, que toutes les valeurs des pins de sortie des widget HAL et/ou que tous les attributs de classe du gestionnaire de classe soient conservés entre les invocations, procéder comme suit:
– Importer le module `gladevcp.persistence`.
– Décider quels attributs d'instance et leurs valeurs par défaut doivent être conservés, le cas échéant,

– décider quels widgets doivent avoir leur état conservé.
– Décrire ces décisions dans le gestionnaire de classe par la méthode `init()` grâce à un dictionnaire imbriqué comme suit:

```
def __init__(self, halcomp,builder,useropts):
    self.halcomp = halcomp
    self.builder = builder
    self.useropts = useropts
    self.defaults = {
        # les noms suivants seront éenregistrs/érestaurs comme attributs de émthode,
        # le émcanisme d'enregistrement/restauration est fortement étyp,
        # les types de variables sont éédrivs depuis le type de la valeur initiale.
        # les types couramment ésupportes sont: int, float, bool, string
        IniFile.vars : { 'nhits' : 0, 'a': 1.67, 'd': True ,'c' : "a string"},
        # pour enregistrer/restaurer lé'tat de tous les widgets pour lesquels
        # c'est ésens, ajouter cela:
        IniFile.widgets : widget_defaults(builder.get_objects())
        # une alternative ésense pourrait être de ne retenir que lé'tat de
        # tous les widgets de sortie HAL:
        # IniFile.widgets: widget_defaults(select_widgets(self.builder.get_objects(),
hal_only=True,output_only = True)),
    }
```

Puis associer un fichier .ini avec ce descripteur:

```
self.ini_filename = __name__ + '.ini'
self.ini = IniFile(self.ini_filename,self.defaults,self.builder)
self.ini.restore_state(self)
```

Ensuite *restore_state()*, aura automatiquement les attributs définis si ce qui suit a été exécuté:

```
self.nhits = 0
self.a = 1.67
self.d = True
self.c = "a string"
```

Noter que les types sont enregistrés et conservés lors de la restauration. Cet exemple suppose que le fichier .ini n'existe pas ou qu'il contient les valeurs par défaut depuis self.defaults.

Après cette incantation, on peut utiliser les méthodes IniFil suivantes:

ini.save_state(obj)
: enregistre les attributs des objets depuis le dictionnaire IniFil.vars l'état du widget comme décrit par IniFile.widgets dans self.defaults

ini.create_default_ini()
: crée un fichier .ini avec les valeurs par défaut

ini.restore_state(obj)
: restaure les pins de HAL et les attributs des objets enregistrés/initialisés par défaut comme précédemment

Pour enregistrer le widget et/ou l'état des variables en quittant, connecter un gestionnaire de signal à la fenêtre de niveau supérieur `window1`, détruire l'événement:

```
def on_destroy(self,obj,data=None):
    self.ini.save_state(self)
```

La prochaine fois que l'application GladeVCP démarrera, les widgets doivent retrouver l'état qu'ils avaient à la fermeture de l'application.

11.7.10 Édition manuelle des fichiers .ini

Il est possible de faire cela, mais noter que les valeurs dans self.defaults écraseront votre édition si il y a erreur de frappe ou de syntaxe. Une erreur détectée, un message émis dans la console, donneront des indices sur ce qui s'est passé et le mauvais fichier ini sera renommé avec le suffixe .BAD. Après une mauvaise initialisation, les fichiers .BAD les plus anciens seront écrasés.

11.7.11 Ajouter des pins de HAL

Si il faut des pins de HAL non associées avec un widget HAL, les ajouter comme ci-dessous:

```
import hal_glib
...
# dans le gestionnaire de classe __init__():
self.example_trigger = hal_glib.GPin(halcomp.newpin('example-trigger', hal.HAL_BIT, hal. ↩
    HAL_IN))
```

Pour appeler une fonction de rappel quand la valeur de cette pin change il faut associer une fonction de rappel `value-changed` avec cette pin, ajouter pour cela:

```
self.example_trigger.connect('value-changed', self._on_example_trigger_change)
```

et définir une méthode de fonction de rappel (ou une fonction, dans ce cas laisser tomber le paramètre `self`):

```
# noter *_* - cette émthode n'est pas visible dans l'arbre du widget
def _on_example_trigger_change(self,pin,userdata=None):
    print "pin value changed to:" % (pin.get())
```

11.7.12 Ajout de timers

Depuis que GladeVCP utilise les widgets Gtk qui se rattachent sur les classes de base GObject, la totalité des fonctionnalités de la glib est disponible. Voici un exemple d' horloge de fonction de rappel:

```
def _on_timer_tick(self,userdata=None):
    ...
    return True # pour relancer l'horloge; return False pour un monostable
...
# édmonstration d'une horloge lente en âtche de fond - la égranularit est de une seconde
# pour une horloge rapide (égranularit 1 ms), utiliser cela:
# glib.timeout_add(100, self._on_timer_tick,userdata) # 10Hz
glib.timeout_add_seconds(1, self._on_timer_tick)
```

11.7.13 Exemples, et lancez votre propre application GladeVCP

Visiter `linuxcnc/configs/gladevcp` pour des exemples prêt à l'emploi et points de départ de vos propres projets.

11.8 Questions & réponses

1. *Je reçois un événement unmap inattendu dans ma fonction de gestionnaire juste après le démarrage, qu'est-ce que c'est?*
   C'est la conséquence d'avoir dans votre fichier d'UI Glade la propriété de la fenêtre window1 visible fixée à True, il y a changement de parents de la fenêtre GladeVCP dans Axis ou touchy. L'arbre de widget de GladeVCP est créé, incluant une fenêtre de niveau supérieur puis *re-aparenté dans Axis*, laissant trainer les orphelins de la fenêtre de niveau supérieur. Pour éviter d'avoir cette fenêtre vide qui traine, elle est unmapped (rendue invisible) et la cause du signal unmap que vous avez eux. Suggestion pour fixer le problème: fixer window1.visible à False et ignorer le message initial d'événement unmap.
2. *Mon programme GladeVCP démarre, mais aucune fenêtre n'apparait alors qu'elle devrait.*
   La fenêtre allouée par Axis pour GladeVCP obtient la *taille naturelle* de tous ses enfants combinés. C'est au widget enfant a réclamer une taille (largeur et/ou hauteur). Cependant, toutes le fenêtres ne demandent pas une plus grande que 0, par exemple, le widget Graph dans sa forme courante. Si il y a un tel widget dans votre fichier Glade et que c'est lui qui défini la disposition vous devrez fixer sa largeur explicitement. Noter que la largeur et la hauteur de la fenêtre window1 dans Glade n'a pas de sens puisque cette fenêtre sera orpheline lors du changement de parent et donc sa géométrie n'aura aucun impact sur les mise en page (voir ci-dessus). La règle générale est la suivante: si vous exécutez manuellement un fichier UI avec *gladevcp <fichierui>* et que sa fenêtre a une géométrie raisonnable, elle devrait apparaitre correctement dans Axis.

3. *Je veux une Led clignotante, alors j'ai coché une case pour la laisser clignoter avec un intervalle de 100ms. Elle devrait clignoter, mais je reçois un :Warning: value* 0 *le type* gint *est invalide ou hors de l'étendue pour les propriétés de* led-blink-rate*, c'est quoi le type gint?*

   Il semble qu'il s'agisse d'un bug de Glade. Il faut re-saisir une valeur sur le champ de la fréquence de clignotement et enregistrer à nouveau. Ça a marché pour moi.

4. *Mon panneau gladevcp ne marche pas dans Axis, il n'enregistre pas les états quand je ferme Axis, j'ai pourtant défini un gestionnaire on_destroy attaché au signal destroy de la fenêtre.*

   Ce gestionnaire est très probablement lié à window1, qui en raison du changement de parent ne peux pas assurer cette fonction. Attachez le gestionnaire on_destroy handler au signal destroy d'une fenêtre intérieure. Par exemple: J'ai un notebook dans window1, attaché on_destroy au signal destroy de notebooks et ça marche bien. Il ne marcherait pas pour window1.

11.9 Troubleshooting

– make sure your 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/dis` file lying around besides the newer `/usr/lib/python2.6/dist-packages/_hal.so` (note 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.

11.10 Notes d'implémentation: la gestion des touches dans Axis

Nous pensons que la gestion des touches fonctionne bien, mais comme c'est un nouveau code, nous devons vous informer à ce propos pour que vous puissiez surveiller ces problèmes; S'il vous plaît, faites nous savoir si vous connaissez des erreurs ou des choses bizarres. Voici l'histoire:

Axis utilise le jeu de widget de TkInter. L'application GladeVCP utilise les widgets Gtk et démarre dans un contexte de processus différent. Ils sont attachés dans Axis avec le protocole Xembed. Ce qui permet à une application enfant comme GladeVCP de bien tenir proprement dans la fenêtre d'un parent et, en théorie, d'être intégrée au gestionnaire d'événements.

Toutefois, cela suppose que parent et enfant supportent tous les deux proprement le protocole Xembed, c'est le cas avec Gtk, pas avec TkInter. Une conséquence de cela, c'est que certaines touches ne sont pas transmises correctement dans toutes les circonstances depuis un panneau GladeVCP vers Axis. Une d'elle est la touche *Entrée*. Ou quand le widget SpinButton a le focus, dans ce cas, par exemple la touche Échap n'est pas bien transmise à Axis et cause un abandon avec des conséquences potentiellement désastreuses.

Par conséquent, les événements touches dans GladeVCP, sont traités explicitement, et sélectivement transmises à Axis, pour assurer que de telles situations ne puissent pas survenir. Pour des détails, voir la fonction *keyboard_forward()* dans la *lib/python/-gladevcp/xembed.py*. :leveloffset: 0

Chapitre 12

Notions avancées

Chapitre 13

Python Interface

This documentation describes the *linuxcnc* python module, which provides a python API for talking to LinuxCNC.

13.1 The linuxcnc Python module

User interfaces control LinuxCNC activity by sending NML messages to the LinuxCNC task controller, and monitor results by observing the LinuxCNC status structure, as well as the error reporting channel.

Programmatic access to NML is through a C++ API; however, the most important parts of the NML interface to LinuxCNC are also available to Python programs through the `linuxcnc` module.

Beyond the NML interface to the command, status and error channels, the `linuxcnc` module also contains:
– support for reading values from ini files
– support for position logging (???)

13.2 Usage Patterns for the LinuxCNC NML interface

The general pattern for `linuxcnc` usage is roughly like this:
– import the `linuxcnc` module
– establish connections to the command, status and error NML channels as needed
– poll the status channel, either periodically or as needed
– before sending a command, determine from status whether it is in fact OK to do so (for instance, there is no point in sending a *Run* command if task is in the ESTOP state, or the interpreter is not idle)
– send the command by using one of the `linuxcnc` command channel methods
To retrieve messages from the error channel, poll the error channel periodically, and process any messages retrieved.
– poll the status channel, either periodically or as needed
– print any error message FIXME: explore the exception code
`linuxcnc` also defines the `error` Python exception type to support error reporting.

13.3 Reading LinuxCNC status

Here is a Python fragment to explore the contents of the `linuxcnc.stat` object which contains some 80+ values (run while linuxcnc is running for typical values):

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys
import linuxcnc
```

```
try:
        s = linuxcnc.stat() # create a connection to the status channel
        s.poll() # get current values
except linuxcnc.error, detail:
        print "error", detail
        sys.exit(1)
for x in dir(s):
        if not x.startswith("_"):
                print x, getattr(s,x)
```

Linuxcnc uses the default compiled-in path to the NML configuration file unless overridden, see Reading ini file values for an example.

13.3.1 linuxcnc.stat attributes

acceleration
(returns float) - default acceleration, reflects the ini entry [TRAJ] DEFAULT_ACCELERATION.

active_queue
(returns integer) - number of motions blending.

actual_position
(returns tuple of floats) - current trajectory position, (x y z a b c u v w) in machine units.

adaptive_feed_enabled
(returns boolean) - status of adaptive feedrate override (0/1).

ain
(returns tuple of floats) - current value of the analog input pins.

angular_units
(returns float) - reflects [TRAJ] ANGULAR_UNITS ini value.

aout
(returns tuple of floats) - current value of the analog output pins.

axes
(returns integer) - reflects [TRAJ] AXES ini value.

axis
(returns tuple of dicts) - reflecting current axis values. See The axis dictionary.

axis_mask
(returns integer) - mask of axis available as defined by [TRAJ] COORDINATES in the ini file. Returns the sum of the axes X=1, Y=2, Z=4, A=8, B=16, C=32, U=64, V=128, W=256.

block_delete
(returns boolean) - block delete curren status.

command
(returns string) - currently executing command.

current_line
(returns integer) - currently executing line, int.

current_vel
(returns float) - current velocity in user units per second.

cycle_time
(returns float) - thread period

debug
(returns integer) - debug flag from the ini file.

delay_left
(returns float) - remaining time on dwell (G4) command, seconds.

din
(returns tuple of integers) - current value of the digital input pins.

distance_to_go
(returns float) - remaining distance of current move, as reported by trajectory planner.

dout
(returns tuple of integers) - current value of the digital output pins.

dtg
(returns tuple of floats) - remaining distance of current move for each axis, as reported by trajectory planner.

echo_serial_number
(returns integer) - The serial number of the last completed command sent by a UI to task. All commands carry a serial number. Once the command has been executed, its serial number is reflected in `echo_serial_number`.

enabled
(returns boolean) - trajectory planner enabled flag.

estop
(returns integer) - Returns either STATE_ESTOP or not.

exec_state
(returns integer) - task execution state. One of EXEC_ERROR, EXEC_DONE, EXEC_WAITING_FOR_MOTION, EXEC_WAIT
EXEC_WAITING_FOR_PAUSE,EXEC_WAITING_FOR_MOTION_AND_IO, EXEC_WAITING_FOR_DELAY, EXEC_WAI

feed_hold_enabled
(returns boolean) - enable flag for feed hold.

feed_override_enabled
(returns boolean) - enable flag for feed override.

feedrate
(returns float) - current feedrate override, 1.0 = 100%.

file
(returns string) - currently loaded gcode filename with path.

flood
(returns integer) - Flood status, either *FLOOD_OFF* or *FLOOD_ON*.

g5x_index
(returns integer) -
– replace with constants list currently active coordinate system, G54=1, G55=2 etc.

g5x_offset
(returns tuple of floats) - offset of the currently active coordinate system.

g92_offset
(returns tuple of floats) - pose of the current g92 offset.

gcodes
(returns tuple of integers) - Active G-codes for each modal group. G code constants G_0, G_1, G_2, G_3, G_4, G_5, G_5_1, G_5_2, G_5_3, G_7, G_8, G_100, G_17, G_17_1, G_18, G_18_1, G_19, G_19_1, G_20, G_21, G_28, G_28_1, G_30, G_30_1, G_33, G_33_1, G_38_2, G_38_3, G_38_4, G_38_5, G_40, G_41, G_41_1, G_42, G_42_1, G_43, G_43_1, G_43_2, G_49, G_50, G_51, G_53, G_54, G_55, G_56, G_57, G_58, G_59, G_59_1, G_59_2, G_59_3, G_61, G_61_1, G_64, G_73, G_76, G_80, G_81, G_82, G_83, G_84, G_85, G_86, G_87, G_88, G_89, G_90, G_90_1, G_91, G_91_1, G_92, G_92_1, G_92_2, G_92_3, G_93, G_94, G_95, G_96, G_97, G_98, G_99

homed
(returns tuple of integers) - 0 = not homed, 1 = homed.

id
(returns integer) - currently executing motion id.

inpos
(returns boolean) - machine-in-position flag.

input_timeout
(returns boolean) - flag for M66 timer in progress.

interp_state
(returns integer) - current state of RS274NGC interpreter. One of INTERP_IDLE, INTERP_READING, INTERP_PAUSED, INTERP_WAITING.

interpreter_errcode
(returns integer) - current RS274NGC interpreter return code. One of INTERP_OK, INTERP_EXIT, INTERP_EXECUTE_FINIS
INTERP_ENDFILE, INTERP_FILE_NOT_OPEN, INTERP_ERROR. see src/emc/nml_intf/interp_return.hh

joint_actual_position
(returns tuple of floats) - actual joint positions.

joint_position
(returns tuple of floats) - Desired joint positions.

kinematics_type
(returns integer) - identity=1, serial=2, parallel=3, custom=4 .

limit
(returns tuple of integers) - axis limit masks. minHardLimit=1, maxHardLimit=2, minSoftLimit=4, maxSoftLimit=8.

linear_units
(returns float) -

reflects [TRAJ]LINEAR_UNITS ini value.

lube
(returns integer) - *lube on* flag.

lube_level
(returns integer) - reflects *iocontrol.0.lube_level*.

max_acceleration
(returns float) - maximum acceleration. reflects [TRAJ] MAX_ACCELERATION.

max_velocity
(returns float) - maximum velocity. reflects [TRAJ] MAX_VELOCITY.

mcodes
(returns tuple of 10 integers) - currently active M-codes.

mist
(returns integer) - Mist status, either MIST_OFF' or *MIST_ON*

motion_line
(returns integer) - source line number motion is currently executing. Relation to `id` unclear.

motion_mode
(returns integer) - This is the mode of the Motion controller. One of TRAJ_MODE_COORD, TRAJ_MODE_FREE, TRAJ_MODE_TELEOP.

motion_type
(returns integer) - The type of the currently executing motion. One of:
- MOTION_TYPE_TRAVERSE
- MOTION_TYPE_FEED
- MOTION_TYPE_ARC
- MOTION_TYPE_TOOLCHANGE
- MOTION_TYPE_PROBING
- MOTION_TYPE_INDEXROTARY
- Or 0 if no motion is currently taking place.

optional_stop
(returns integer) - option stop flag.

paused
(returns boolean) - `motion paused` flag.

pocket_prepped
(returns integer) - A Tx command completed, and this pocket is prepared. -1 if no prepared pocket.

poll()
-(built-in function) method to update current status attributes.

position
(returns tuple of floats) - trajectory position.

probe_tripped
(returns boolean) - flag, True if probe has tripped (latch)

probe_val
(returns integer) - reflects value of the `motion.probe-input` pin.

probed_position
(returns tuple of floats) - position where probe tripped.

probing
(returns boolean) - flag, True if a probe operation is in progress.

program_units
(returns integer) - one of CANON_UNITS_INCHES=1, CANON_UNITS_MM=2, CANON_UNITS_CM=3

queue
(returns integer) - current size of the trajectory planner queue.

queue_full
(returns boolean) - the trajectory planner queue is full.

rapidrate
(returns float) - rapid override scale.

read_line
(returns integer) - line the RS274NGC interpreter is currently reading.

default velocity. reflects [TRAJ] DEFAULT_VELOCITY.

13.3.2 The `axis` dictionary

The axis configuration and status values are available through a list of per-axis dictionaries. Here's an example how to access an attribute of a particular axis:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import linuxcnc
s = linuxcnc.stat()
s.poll()
print "Axis 1 homed: ", s.axis[1]["homed"]
```

For each axis, the following dictionary keys are available:

axisType
(returns integer) - type of axis configuration parameter, reflects [AXIS_x]TYPE. LINEAR=1, ANGULAR=2. See Axis ini configuration for details.

backlash
(returns float) - Backlash in machine units. configuration parameter, reflects [AXIS_x]BACKLASH.

enabled
(returns integer) - non-zero means enabled.

fault
(returns integer) - non-zero means axis amp fault.

ferror_current
(returns float) - current following error.

ferror_highmark
(returns float) - magnitude of max following error.

homed
(returns integer) - non-zero means has been homed.

homing
(returns integer) - non-zero means homing in progress.

inpos
(returns integer) - non-zero means in position.

input
(returns float) - current input position.

max_ferror
(returns float) - maximum following error. configuration parameter, reflects [AXIS_x]FERROR.

max_hard_limit
(returns integer) - non-zero means max hard limit exceeded.

max_position_limit
(returns float) - maximum limit (soft limit) for axis motion, in machine units.configuration parameter, reflects [AXIS_x]MAX_LIM

max_soft_limit
non-zero means `max_position_limit` was exceeded, int

min_ferror
(returns float) - configuration parameter, reflects [AXIS_x]MIN_FERROR.

min_hard_limit
(returns integer) - non-zero means min hard limit exceeded.

min_position_limit
(returns float) - minimum limit (soft limit) for axis motion, in machine units.configuration parameter, reflects [AXIS_x]MIN_LIMI

min_soft_limit
(returns integer) - non-zero means `min_position_limit` was exceeded.

output
(returns float) - commanded output position.

override_limits
(returns integer) - non-zero means limits are overridden.

units
(returns float) - units per mm, deg for linear, angular

velocity
(returns float) - current velocity.

13.4 Preparing to send commands

Some commands can always be sent, regardless of mode and state; for instance, the `linuxcnc.command.abort()` method can always be called.

Other commands may be sent only in appropriate state, and those tests can be a bit tricky. For instance, an MDI command can be sent only if:
– ESTOP has not been triggered, and
– the machine is turned on and
– the axes are homed and
– the interpreter is not running and
– the mode is set to `MDI mode`
so an appropriate test before sending an MDI command through `linuxcnc.command.mdi()` could be:

```
#!/usr/bin/env python
# -*- 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 and (s.interp_state == linuxcnc. ↩
            INTERP_IDLE)

if ok_for_mdi():
        c.mode(linuxcnc.MODE_MDI)
        c.wait_complete() # wait until mode switch executed
        c.mdi("G0 X10 Y20 Z30")
```

13.5 Sending commands through `linuxcnc.command`

Before sending a command, initialize a command channel like so:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import linuxcnc
c = linuxcnc.command()

# Usage examples for some of the commands listed below:
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, axis)
c.jog(linuxcnc.JOG_CONTINUOUS, axis, speed)
c.jog(linuxcnc.JOG_INCREMENT, axis, speed, 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)
```

13.5.1 `linuxcnc.command` attributes

`serial`
the current command serial number

13.5.2 `linuxcnc.command` methods:

`abort()`
send EMC_TASK_ABORT message.

`auto(int[, int])`
run, step, pause or resume a program.

`brake(int)`
engage or release spindle brake.

`debug(int)`
set debug level via EMC_SET_DEBUG message.

`feedrate(float)`
set the feedrate.

`flood(int)`
turn on/off flooding.

`home(int)`
home a given axis.

`jog(int, int, [, int[,int]])`
Syntax:

jog(command, axis[, velocity[, distance]])
jog(linuxcnc.JOG_STOP, axis)
jog(linuxcnc.JOG_CONTINUOUS, axis, velocity)
jog(linuxcnc.JOG_INCREMENT, axis, velocity, distance)
Constants:
JOG_STOP (0)
JOG_CONTINUOUS (1)
JOG_INCREMENT (2)

load_tool_table()
reload the tool table.

maxvel(float)
set maximum velocity

mdi(string)
send an MDI command. Maximum 255 chars.

mist(int)
turn on/off mist.
Syntax:
mist(command)
mist(linuxcnc.MIST_ON) [(1)]
mist(linuxcnc.MIST_OFF) [(0)]
Constants:
MIST_ON (1)
MIST_OFF (0)

mode(int)
set mode (MODE_MDI, MODE_MANUAL, MODE_AUTO).

override_limits()
set the override axis limits flag.

program_open(string)
open an NGC file.

reset_interpreter()
reset the RS274NGC interpreter

set_adaptive_feed(int)
set adaptive feed flag

set_analog_output(int, float)
set analog output pin to value

set_block_delete(int)
set block delete flag

set_digital_output(int, int)
set digital output pin to value

set_feed_hold(int)
set feed hold on/off

set_feed_override(int)
set feed override on/off

set_max_limit(int, float)
set max position limit for a given axis

set_min_limit()
set min position limit for a given axis

set_optional_stop(int)
set optional stop on/off

set_spindle_override(int)
set spindle override flag

spindle(int)
set spindle direction. Argument one of SPINDLE_FORWARD, SPINDLE_REVERSE, SPINDLE_OFF, SPINDLE_INCREASE, SPINDLE_DECREASE, or SPINDLE_CONSTANT.

spindleoverride(float)
set spindle override factor

state(int)
set the machine state. Machine state should be STATE_ESTOP, STATE_ESTOP_RESET, STATE_ON, or STATE_OFF

teleop_enable(int)
enable/disable teleop mode.

teleop_vector(float, float, float [,float, float, float])
set teleop destination vector

tool_offset(int, float, float, float, float, float, int)
set the tool offset. See usage example above.

traj_mode(int)
set trajectory mode. Mode is one of MODE_FREE, MODE_COORD, or MODE_TELEOP.

unhome(int)
unhome a given axis.

wait_complete([float])
wait for completion of the last command sent. If timeout in seconds not specified, default is 1 second.

13.6 Reading the error channel

To handle error messages, connect to the error channel and periodically poll() it.

Note that the NML channel for error messages has a queue (other than the command and status channels), which means that the first consumer of an error message deletes that message from the queue; whether your another error message consumer (e.g. Axis) will *see* the message is dependent on timing. It is recommended to have just one error channel reader task in a setup.

```
#!/usr/bin/env python
# -*- 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
```

13.7 Reading ini file values

Here's an example for reading values from an ini file through the `linuxcnc.ini` object:

```
#!/usr/bin/env python
# -*- 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() returns None if the key wasnt found - the
# following idiom is useful for setting a default value:

machine_name = inifile.find("EMC", "MACHINE") or "unknown"
print "machine name: ", machine_name

# inifile.findall() returns a list of matches, or an empty list
# if the key wasnt found:

extensions = inifile.findall("FILTER", "PROGRAM_EXTENSION")
print "extensions: ", extensions

# override default NML file by ini parameter if given
nmlfile = inifile.find("EMC", "NML_FILE")
if nmlfile:
    linuxcnc.nmlfile = os.path.join(os.path.dirname(sys.argv[1]), nmlfile)
```

13.8 The `linuxcnc.positionlogger` type

Some usage hints can be gleaned from `src/emc/usr_intf/gremlin/gremlin.py`.

13.8.1 members

`npts`
number of points.

13.8.2 methods

`start(float)`
start the position logger and run every ARG seconds

`clear()`
clear the position logger

`stop()`
stop the position logger

`call()`
Plot the backplot now.

`last([int])`
Return the most recent point on the plot or None , :lang: fr :toc:

Chapitre 14

La cinématique dans LinuxCNC

14.1 Introduction

Habituellement quand nous parlons de machines CNC, nous pensons à des machines programmées pour effectuer certains mouvements et effectuer diverses tâches. Pour avoir une représentation unifiée dans l'espace de ces machines, nous la faisons correspondre à la vision humaine de l'espace en 3D, la plupart des machines (sinon toutes) utilisent un système de coordonnées courant, le système Cartésien.

Le système de coordonnées Cartésiennes est composé de 3 axes (X, Y, Z) chacun perpendiculaire aux 2 autres. [1]

Quand nous parlons d'un programme G-code (RS274/NGC) nous parlons d'un certain nombre de commandes (G0, G1, etc.) qui ont comme paramètres (X- Y- Z-). Ces positions se réfèrent exactement à des positions Cartésiennes. Une partie du contrôleur de mouvements de LinuxCNC est responsable de la translation entre ces positions et les positions correspondantes de la cinématique de la machine [2].

14.1.1 Les articulations par rapport aux axes

Une articulation, pour une machine CNC est un des degrés physiques de liberté de la machine. Elle peut être linéaire (vis à billes) ou rotative (table tournante, articulations d'un bras robotisé). Il peut y avoir n'importe quel nombre d'articulations sur une machine. Par exemple, un robot classique dispose de 6 articulations et une fraiseuse classique n'en a que 3.

Sur certaines machines, les articulations sont placées de manière à correspondre aux axes cinématiques (articulation 0 le long de l'axe X, articulation 1 le long de l'axe Y et articulation 2 le long de l'axe Z), ces machines sont appelées machines Cartésiennes (ou encore machines à cinématiques triviales . Ce sont les machines les plus courantes parmi les machines-outils mais elles ne sont pas courantes dans d'autres domaines comme les machines de soudage (ex: robots de soudage de type puma).

14.2 Cinématiques triviales

Comme nous l'avons vu, il y a un groupe de machines sur lesquelles chacun des axes est placé le long d'un des axes Cartésiens. Sur ces machines le passage, du plan de l'espace Cartésien (le programme G-code) au plan de l'espace articulation (l'actuateur actuel de la machine), est trivial. C'est un simple plan 1:1:

```
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];
```

1. Le mot *axes* est aussi communément (et incorrectement) utilisé à propos des machines CNC, il fait référence aux directions des mouvements de la machine.
2. Cinématique: une fonction à deux voies pour transformer un espace Cartésien en espace à articulations

Dans l'extrait de code ci-dessus, nous pouvons voir comment le plan est fait: la position X est identique avec la articulation 0, Y avec la articulation 1 etc. Nous nous référons dans ce cas à une cinématique directe (une transformation avant), tandis que dans l'extrait de code suivant il est fait référence à une cinématique inverse (ou une transformation inverse):

```
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;
```

Comme on peut le voir, c'est assez simple de faire la transformation d'une machine à cinématique banale (ou Cartésienne). Cela devient un peu plus compliqué si il manque un axe à la machine. [3] [4]

14.3 Cinématiques non triviales

Il peut y avoir un certain nombre de types de configurations de machine (robots: puma, scara; hexapodes etc.) Chacun d'eux est mis en place en utilisant des articulations linéaires et rotatives. Ces articulations ne correspondent pas habituellement avec les coordonnées Cartésiennes, cela nécessite une fonction cinématique qui fasse la conversion (en fait 2 fonctions: fonction en avant et inverse de la cinématique).

Pour illustrer ce qui précède, nous analyserons une simple cinématique appelée bipode (une version simplifiée du tripode, qui est déjà une version simplifiée de l'hexapode).

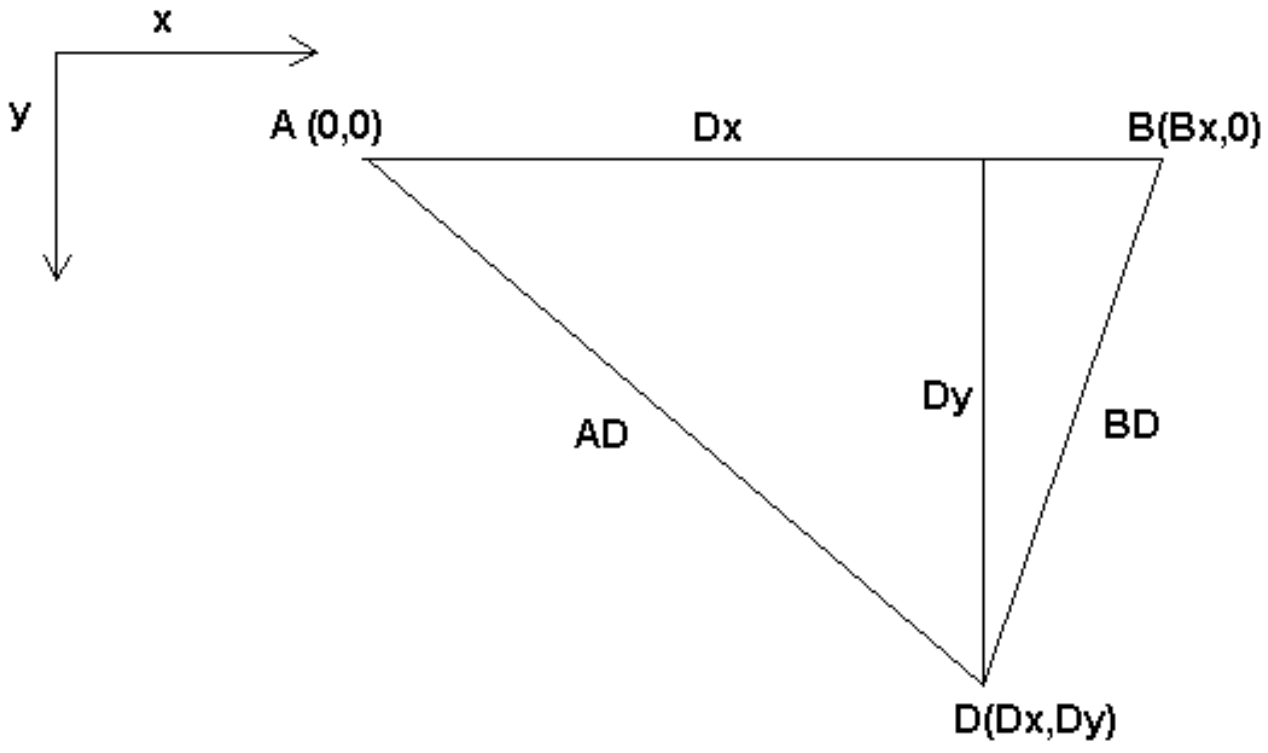


FIGURE 14.1 – Définir un bipode

3. Si la machine (par exemple un tour) est montée avec seulement les axes X, Z et A et que le fichier d'init de LinuxCNC contient uniquement la définition de ces 3 articulations, alors l'assertion précédente est fausse. Parce-que nous avons actuellement (joint0=x, joint1=Z, joint2=A) ce qui suppose que joint1=Y. Pour faire en sorte que cela fonctionne dans LinuxCNC il suffit de définir tous les axes (XYZA), LinuxCNC utilisera alors une simple boucle dans HAL pour l'axe Y inutilisé.

4. Une autre façon de le faire fonctionner, est de changer le code correspondant et recompiler le logiciel.

Le bipode dont nous parlons est un appareil, composé de deux moteurs placés sur un mur, à cet appareil un mobile est suspendu par des fils. Les articulations dans ce cas sont les distances entre le mobile et les moteurs de l'appareil (nommées AD et BD sur la figure ci-dessus.

La position des moteurs est fixée par convention. Le moteur A est en (0,0), qui signifie que sa coordonnée X est 0 et sa coordonnée Y également 0. Le moteur B est placé en (Bx, 0), se qui veut dire que sa coordonnée X est Bx.

Notre pointe mobile se trouvera au point D défini par les distances AD et BD, et par les coordonnées Cartésiennes Dx, Dy.

La tâche de la cinématique consistera à transformer les longueurs des articulations en (AD, BD) en coordonnées Cartésiennes (Dx, Dy) et vice-versa.

14.3.1 Transformation avant

Pour effectuer la transformation de l'espace articulation en espace Cartésien nous allons utiliser quelques règles de trigonomètrie (le triangle rectangle déterminé par les points (0,0), (Dx,0), (Dx,Dy) et le triangle rectangle (Dx,0), (Bx,0) et (Dx,Dy).

Nous pouvons voir aisément que $\mathbf{AD^2=x^2+y^2}$, de même que $\mathbf{BD^2=(Bx-x)^2+y^2}$.

Si nous soustrayons l'un de l'autre nous aurons: $\mathbf{AD^2-BD^2=x^2+y^2-x^2+2*x*Bx-Bx^2-y^2}$

et par conséquent: $\mathbf{x=(AD^2-BD^2+Bx^2)/(2*Bx)}$

De là nous calculons: $\mathbf{y=sqrt(AD^2-x^2)}$

Noter que le calcul inclus la racine carrée de la différence, mais qu'il n'en résulte pas un nombre réel. Si il n'y a aucune coordonnée Cartésienne pour la position de cette articulation, alors la position est dite singulière. Dans ce cas, la cinématique inverse retourne -1.

Traduction en 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;
```

14.3.2 Transformation inverse

La cinématique inverse est beaucoup plus simple dans notre exemple, de sorte que nous pouvons l'écrire directement:

$\mathbf{AD=sqrt(x^2+y^2)}$

$\mathbf{BD=sqrt((Bx-x)^2+y^2)}$

ou traduite en 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;
```

14.4 Détails d'implémentation

Un module cinématique est implémenté comme un composant de HAL, et il est permis d'exporter ses pins et ses paramètres. Il consiste en quelques fonctions "C" (par opposition au fonctions de HAL):

int kinematicsForward(const double *joint, EmcPose *world, const KINEMATICS_FORWARD_FLAGS *fflags, KINEMATIC

Implémente la fonction cinématique avant.

int kinematicsInverse(const EmcPose * world, double *joints, const KINEMATICS_INVERSE_FLAGS *iflags, KINEMATICS

Implémente la fonction cinématique inverse.

KINEMATICS_TYPE kinematicsType(void)_
Retourne l'identificateur de type de la cinématique, typiquement *KINEMATICS_BOTH*.

int kinematicsHome(EmcPose *world, double *joint, KINEMATICS_FORWARD_FLAGS *fflags, KINEMATICS_INVERSE_

La fonction prise d'origine de la cinématique ajuste tous ses arguments à leur propre valeur à une position d'origine connue. Quand elle est appelée, cette position doit être ajustée, quand elle est connue, comme valeurs initiales, par exemple depuis un fichier INI. Si la prise d'origine de la cinématique peut accepter des points de départ arbitraires, ces valeurs initiales doivent être utilisées.

int rtapi_app_main(void) , void rtapi_app_exit(void)
Il s'agit des fonctions standards d'installation et de la désinstallation des modules RTAPI.

Quand ils sont contenus dans un seul fichier source, les modules de la cinématique peuvent être compilés et installés par *comp*. Voir la manpage *comp(1)* pour d'autres informations.

Chapitre 15

Réglages des pas à pas

15.1 Obtenir le meilleur pilotage logiciel possible

Faire générer les impulsions de pas au logiciel présente un gros avantage, c'est gratuit. Quasiment chaque PC dispose d'un port parallèle capable de sortir sur ses broches les signaux de pas générés par le logiciel. Cependant, les générateurs d'impulsions logiciels ont aussi quelques inconvénients:
– La fréquence maximum des impulsions est limitée.
– Les impulsions générées sont irrégulières à cause du bruit.
– Elles sont sujettes à la charge du CPU
Ce chapitre présente certaines mesures qui vous aideront à obtenir les meilleurs résultats du logiciel.

15.1.1 Effectuer un test de latence

Le CPU n'est pas le seul facteur déterminant la latence. Les cartes mères, cartes graphiques, ports USB et nombre d'autres choses peuvent la dégrader. La meilleure façon de savoir ce que vous pouvez attendre d'un PC consiste à exécuter le test de latence RTAI.

Lancer un test comme décrit au chapitre sur le test de latence.

15.1.2 Connaître ce dont vos cartes de pilotage ont besoin

Les différente marques de cartes de pilotage de moteurs pas à pas demandent toutes des timing différents pour les impulsions de commande de pas et de direction. Aussi vous avez besoin d'accéder (ou Google) à la fiche des spécifications techniques de votre carte.

Par exemple, le manuel du Gecko G202 indique:

```
Step Frequency: 0 to 200 kHz
Step Pulse "0" Time: 0.5 µs min (Step on falling edge)
Step Pulse "1" Time: 4.5 µs min
Direction Setup: 1 µs min (20 µs min hold time after Step edge)
```

Les spécifications du Gecko G203V indiquent:

```
Step Frequency: 0 to 333 kHz
Step Pulse "0" Time: 2.0 µs min (Step on rising edge)
Step Pulse "1" Time: 1.0 µs min

Direction setup:
    200 ns (0.2µs) before step pulse rising edge
    200 ns (0.2µs) hold after step pulse rising edge
```

Un carte Xylotex donne dans ses données techniques un superbe graphe du timing nécessaire, il indique:

```
Minimum DIR setup time before rising edge of STEP Pulse 200ns Minimum
DIR hold time after rising edge of STEP pulse 200ns
Minimum STEP pulse high time 2.0µs
Minimum STEP pulse low time 1.0µs
Step happens on rising edge
```

Notez les valeurs que vous trouvez, vous en aurez besoin pour la prochaine étape.

15.1.3 Choisir la valeur de BASE_PERIOD

BASE_PERIOD est l'horloge de votre LinuxCNC. A chaque période, le générateur d'impulsions de pas décide si il est temps pour une autre impulsion. Une période plus courte vous permettra de générer plus d'impulsions par seconde, dans les limites. Mais si vous la réglez trop bas, votre ordinateur va passer autant de temps à générer des impulsions de pas que pour exécuter tous le reste de ses tâches, il finira peut-être même par se bloquer. La latence et la génération de pas exigent d'affecter la plus courte période utilisable, comme nous le verrons un peu plus loin.

Regardons l'exemple du Gecko en premier. Le G202 peut gérer des impulsions restant à l'état bas pendant 0.5µs et à l'état haut pendant 4.5µs, il a besoin que la broche de direction soit stable 1µs avant le front descendant et qu'elle reste stable pendant 20µs après le front descendant. La plus longue durée est de 20µs, c'est le temps de maintien. Une approche simple consisterait à fixer la période à 20µs. Ce qui signifierait que tous les changements d'état des lignes STEP et DIR serait espacés de 20µs. C'est tout bon, non?

Faux! Si la latence était de zéro, et que tous les fronts soient espacés de 20µs, tout irait bien. Mais tous les ordinateurs ont une latence. Si l'ordinateur a 11µs de latence, celà signifie que, ce que l'ordinateur exécute aura parfois un retard de 11µs et la fois suivante pourra être juste à l'heure, le délai entre le premier et le second sera seulement de 9µs. Si le premier génére l'impulsion de pas et le second change la broche de direction, le timing de 20µs requis par le G202 sera tout simplement violé. Cela signifie que votre moteur aura peut être fait un pas dans la mauvaise direction et que votre pièce ne sera pas à la cote.

Le côté vraiment mauvais de ce problème est qu'il peut être très très rare. Les pires latences sont celles qui ne se produisent que quelques fois par minute. Les chances qu'une mauvaise latence de ce genre arrive juste quand le moteur est en train de changer de direction sont faibles. Ainsi, vous avez de très rares erreurs qui vous ruinent une pièce de temps en temps et qui sont impossibles à résoudre.

La façon la plus simple pour éviter ce problème est de choisir une BASE_PERIOD qui soit la somme de la plus longue période requise par votre carte plus la durée de la pire latence de votre ordinateur. Si vous utilisez un Gecko avec un temps de maintien exigé de 20µs et que votre test de latence vous avait donné une latence maximum de 11µs, alors si vous définissez BASE_PERIOD à 20+11 = 31µs (31000 nanosecondes dans le fichier ini), vous aurez la garantie de répondre aux exigences de votre carte de pilotage.

Mais c'est un compromis. Faire une impulsion de pas demande au moins deux périodes. Une pour débuter l'impulsion, et une pour y mettre fin. Etant donné que la période est de 31µs, il faut 2x31 = 62µs pour créer une impulsion de pas. Ce qui signifie que la fréquence de pas maximum sera seulement de 16129 pas par seconde. Pas très bon. (Mais n'abandonnez pas, nous avons encore quelques réglages à faire dans la section suivante.)

Pour la Xylotex, la configuration demande des temps de maintien très courts de 200ns chacun (0.2µs). Le temps le plus long est de 2µs. Si vous avez 11µs de latence, alors vous pouvez définir BASE_PERIOD aussi bas que 11+2 = 13µs. Se débarrasser du long temps de maintien de 20µs aide vraiment. Avec une période de 13µs, un pas complet ne dure que 26µs = 2x13 et la fréquence maximum est de 38461 pas par seconde!

Mais ne commencez pas à célébrer celà. Notez que 13µs est une période très courte. Si vous essayez d'exécuter le générateur de pas toutes les 13µs, il ne restera peut-être pas assez de temps pour faire autre chose et votre ordinateur se bloquera. Si vous visez des périodes de moins de 25µs, vous devez commencer à 25µs ou plus, lancer LinuxCNC et voir comment les choses réagissent. Si tout va bien, vous pouvez réduire progressivement la période. Si le pointeur de la souris commence à être sacadé et que le reste du PC ralentit, votre période est un peu trop court. Retournez alors à la valeur précédente qui permettent le meilleur fonctionnement.

Dans ce cas, supposons que vous ayez commencé à 25µs, en essayant descendre à 13µs, vous trouvez que c'est autour de 16µs que se situe la limite la plus basse et qu'en dessous l'ordinateur ne répond plus très bien. Alors, vous utilisez 16µs. Avec une

période à 16μs et une latence à 11μs, le temps de sortie le plus court sera de 16-11 = 5μs. La carte demande seulement 2μs, ainsi vous aurez une certaine marge. Il est bon d'avoir une marge si vous ne voulez pas perdre de pas parce que vous auriez réglé un timing trop court.

Quel est la fréquence de pas maximum? Rappelez-vous, deux périodes pour faire un pas. Vous avez réglé la période à 16μs alors qu'un pas prend 32μs. Il fonctionnera à 31250 pas par seconde, ce qui n'est pas mal.

15.1.4 Utiliser steplen, stepspace, dirsetup, et/ou dirhold

Dans la section précédente, nous avons utilisé la carte de puissance Xylotex pour piloter nos moteurs avec une période de 16μs ce qui nous a donné une fréquence de pas de 31250 pas par seconde maximum. Alors que la Gecko a été bloquée à 31μs avec une assez mauvaise fréquence de pas de 16129 pas par seconde. L'exemple de la Xylotex est au mieux de ce que nous puissions faire. Mais la Gecko peut être ameliorées.

Le problème avec le G202 est le temps de maintien demandé de 20μs. Ca plus la latence de 11μs nous oblige à utiliser une période longue de 31μs. Mais le générateur de pas logiciel de LinuxCNC a un certain nombre de paramètres qui permettent d'augmenter les différentes durées d'une période à plusieurs autres. Par exemple, si *steplen* passe de 1 à 2, alors il y aura deux périodes entre le début et la fin de l'impulsion. De même, si *dirhold* passe de 1 à 3, il y aura au moins trois périodes entre l'impulsion de pas et un changement d'état de la broche de direction.

Si nous pouvons utiliser *dirhold* pour le temps de maintien de 20μs demandé, alors le temps le plus long suivant sera de 4.5μs. Ajoutez les 11μs de latence à ces 4.5μs, et vous obtenez une période minimale de 15.5μs. Lorsque vous essayez 15.5μs, vous trouvez que l'ordinateur est très lent, donc vous régler sur 16μs. Si nous laissons *dirhold* à 1 (par défaut), alors le temps minimum entre le pas et la direction est de 16μs moins la période de latence de 11μs = 5μs, ce qui n'est pas suffisant. Nous avons besoin de 15 autres μs, puisque la période est de 16μs, nous avons besoin d'une période de plus. Nous allons donc passer *dirhold* de 1 à 2. Maintenant, le temps minimum entre la fin de l'impulsion et l'impulsion de changement de direction est de 5+16 = 21μs et nous n'avons pas à craindre que la Gecko parte dans la mauvaise direction en raison de la latence.

Si l'ordinateur a une latence de 11μs, alors la combinaison d'une période de base de 16μs et d'une valeur de *dirhold* de 2 garanti que nous serons toujours dans le respect des délais exigés par la Gecko. Pour les pas normaux (sans changement de direction), l'augmentation de la valeur de *dirhold* n'aura aucun effet. Il faudra deux périodes d'un total de 32μs pour faire un seul pas et nous avons la même fréquence de 31250 pas par seconde que nous avions eu avec la Xylotex.

Le temps de latence de 11μs utilisé dans cet exemple est très bon. Si vous travaillez par le biais de ces exemples avec des latences plus grandes, comme 20 ou 25μs, la fréquence de pas la plus grande à la fois pour la Xylotex et la Gecko sera plus faible. Mais les mêmes formules sont applicables pour calculer un BASE_PERIOD optimal et pour régler *dirhold* ou d'autres paramètres du générateur de pas.

15.1.5 Pas de secret!

Pour un système à moteurs pas à pas avec générateur de pas logiciel rapide et fiable, vous ne pouvez pas deviner la période et les autres paramètres de configuration. Vous devez faire des mesures sur votre ordinateur et faire les calculs qui garantirons les meilleurs signaux dont les moteurs ont besoin.

Pour rendre le calcul plus facile, j'ai créé une feuille de calcul Open Office: Step Timing Calculator (en) - Calculatrice Calendrier étape (fr). Vous entrez les résultats du test de latence et les timing de votre carte de pilotage et la feuille calcule la meilleure BASE_PERIOD. Ensuite, vous testez la période pour vous assurer que votre PC ne sera pas ralenti ou bloqué. Enfin, vous entrez dans la période actuelle et la feuille de calcul vous indiquera le réglage de stepgen nécessaire pour répondre aux exigences de votre carte de pilotage. Elle calcule aussi la fréquence de pas maximum que vous serez en mesure de générer.

J'ai ajouté quelques petites choses à la feuille de calcul pour calculer la fréquence maximum et quelques autres calculs.

Chapitre 16

Réglages d'une boucle PID

16.1 Régulation à PID

Un régulateur Proportionnel Intégral Dérivé (PID) est un organe de contrôle qui permet d'effectuer une régulation en boucle fermée d'un procédé.

Le régulateur compare une valeur mesurée sur le procédé avec une valeur de consigne. La différence entre ces deux valeurs (le signal d'*erreur*) est alors utilisée pour calculer une nouvelle valeur d'entrée du procédé tendant à réduire au maximum l'écart entre la mesure et la consigne (signal d'erreur le plus faible possible).

Contrairement aux algorithmes de régulation simples, le contrôle par PID peut ajuster les sorties du procédé, en fonction de l'amplitude du signal d'erreur, et en fonction du temps. Il donne des résultats plus précis et un contrôle plus stable. (Il est montré mathématiquement qu'une boucle PID donne un contrôle plus stable qu'un contrôle proportionnel seul et qu'il est plus précis que ce dernier qui laissera le procédé osciller).

16.1.1 Les bases du contrôle en boucle

Intuitivement, une boucle PID essaye d'automatiser ce que fait un opérateur muni d'un multimètre et d'un potentiomètre de contrôle. L'opérateur lit la mesure de sortie du procédé, affichée sur le multimètre et utilise le bouton du potentiomètre pour ajuster l'entrée du procédé (l'*action*) jusqu'à stabiliser la mesure de la sortie souhaitée, affichée sur le multimètre.

Un boucle de régulation est composée de trois parties:

1. La mesure, effectuée par un capteur connecté à un procédé, par exemple un codeur.
2. La décision, prise par les éléments du régulateur.
3. L'action sur le dispositif de sortie, par exemple: un moteur.

Quand le régulateur lit le capteur, il soustrait la valeur lue à la valeur de la consigne et ainsi, obtient l'«erreur de mesure». Il peut alors utiliser cette erreur pour calculer une correction à appliquer sur la variable d'entrée du procédé (l'*action*) de sorte que cette correction tende à supprimer l'erreur mesurée en sortie de procédé.

Dans une boucle PID, la correction à partir de l'erreur est calculée de trois façons: P) l'erreur de mesure courante est soustraite directement (effet proportionnel), I) l'erreur est intégrée pendant un laps de temps (effet intégral), D) l'erreur est dérivée pendant un laps de temps (effet dérivé).

Une régulation à PID peut être utilisée dans n'importe quel procédé pour contrôler une variable mesurable, en manipulant d'autres variables de ce procédé. Par exemple, elle peut être utilisée pour contrôler: température, pression, débit, composition chimique, vitesse et autres variables.

Dans certains systèmes de régulation, les régulateurs sont placés en série ou en parallèle. Dans ces cas, le régulateur *maître* produit les signaux utilisés par les régulateurs *esclaves*. Une situation courante dans le contrôle des moteurs, la régulation de vitesse, qui peut demander que la vitesse du moteur soit contrôlée par un régulateur *esclave* (souvent intégré dans le variateur de fréquence du moteur) recevant en entrée une valeur proportionnelle à la vitesse. Cette entrée de l'*esclave* est alors fournie par la sortie du régulateur *maître*, lequel reçoit la variable de consigne.

16.1.2 Théorie

Le *PID* représente les abréviations des trois actions qu'il utilise pour effectuer ses corrections, ce sont des ajouts d'un signal à un autre. Tous agissent sur la quantité régulée. Les actions aboutissent finalement à des *soustractions* de l'erreur de mesure, parce que le signal proportionnel est habituellement négatif.

16.1.2.1 Action Proportionnelle

Pour cette action, l'erreur est multipliée par la constante P (pour Proportionnel) qui est négative, puis ajoutée (soustraction de l'erreur de mesure) à la quantité régulée. P est valide uniquement sur la bande dans laquelle le signal de sortie du régulateur est proportionnel à l'erreur du système. Noter que si l'erreur de mesure est égale à zéro, la partie proportionnelle de la sortie du régulateur est également à zéro.

16.1.2.2 Action Intégrale

L'action intégrale fait intervenir la notion de temps. Elle tire profit du signal d'erreur passé qui est intégré (additionné) pendant un laps de temps, puis multiplié par la constante I (négative) ce qui en fait une moyenne, elle est enfin additionnée (soustraction de l'erreur de mesure) à la quantité régulée. La moyenne de l'erreur de mesure permet de trouver l'erreur moyenne entre la sortie du régulateur et la valeur de la consigne. Un système seulement proportionnel oscille en plus et en moins autour de la consigne du fait qu'en arrivant vers la consigne, l'erreur est à zéro, il n'enlève alors plus rien est dépasse la consigne, ou oscille et/ou se stabilise à une valeur trop basse ou trop élevée. L'addition sur l'entrée d'une proportion négative (soustraction) de l'erreur de mesure moyennée, permet toujours de réduire l'écart moyen entre la mesure en sortie et la consigne. Donc finalement, une boucle PI bien réglée verra sa sortie redescendre lentement à la valeur de la consigne.

16.1.2.3 Action Dérivée

L'action dérivée utilise aussi la notion de temps. Elle cherche à anticiper l'erreur future. La dérivée première (la pente de l'erreur) est calculée pour un laps de temps et multipliée par la constante (négative) D, puis elle est additionnée (soustraction de l'erreur de mesure) à la quantité régulée. L'action dérivée de la régulation fourni une réponse aux perturbations agissant sur le système. Plus important est le terme dérivé, plus rapide sera la réponse en sortie à une perturbation sur l'entrée.

Plus techniquement, une boucle PID peut être caractérisée comme un filtre appliqué sur un système complexe d'un domaine fréquentiel. C'est utilisé pour calculer si le système atteindra une valeur stable. Si les valeurs sont choisies incorrectement, le procédé entrera en oscillation et sa sortie n'atteindra jamais la consigne.

16.1.3 Réglage d'une boucle

Régler une boucle de régulation consiste à agir sur les paramètres des différentes actions (gain du proportionnel, gain de l'intégral, gain de la dérivée) sur des valeurs optimales pour obtenir la réponse désirée sur la sortie du procédé. Le comportement des procédés varient selon les applications lors d'un changement de consigne. Certains procédés ne permettent aucun dépassement de la consigne. D'autres doivent minimiser l'énergie nécessaire pour atteindre un nouveau point de consigne. Généralement la stabilité de la réponse est requise, le procédé ne doit pas osciller quels que soient les conditions du procédé et le point de consigne.

Régler une boucle est rendu plus compliqué si le temps de réponse du procédé est long; il peut prendre plusieurs minutes, voir plusieurs heures pour qu'une modification de consigne produise un effet stable. Certains procédés ne sont pas linéaires et les paramètres qui fonctionnent bien à pleine charge ne marchent plus lors du démarrage hors charge du procédé. Cette section décrit quelques méthodes manuelles traditionnelles pour régler ces boucles.

Il existe plusieurs méthodes pour régler une boucle PID. Le choix de la méthode dépendra en grande partie de la possibilité ou non de mettre la boucle «hors production» pour la mise au point ainsi que de la vitesse de réponse du système. Si le système peut être mis hors production, la meilleure méthode de réglage consiste souvent à soumettre le système à un changement de consigne, à mesurer la réponse en fonction du temps et à l'aide de cette réponse à déterminer les paramètres de la régulation.

16.1.3.1 Méthode simple

Si le système doit rester en production, une méthode de réglage consiste à mettre les valeurs I et D à zéro. Augmenter ensuite le gain P jusqu'à ce que la sortie de la boucle oscille. Puis, augmenter le gain I jusqu'à ce que cesse l'oscillation. Enfin, augmenter le gain D jusqu'à ce que la boucle soit suffisamment rapide pour atteindre rapidement sa consigne. Le réglage d'une boucle PID rapide provoque habituellement un léger dépassement de consigne pour avoir une montée plus rapide, mais certains systèmes ne le permettent pas.

Paramètre	**Temps de montée**	**Dépassement**	**Temps de réglage**	**S.S. Error**
P	Augmente	Augmente	Chang. faible	Diminue
I	Diminue	Augmente	Augmente	Eliminate
D	Chang. faible	Diminue	Diminue	Chang. faible

Effets de l'augmentation des paramètres

16.1.3.2 Méthode de Ziegler-Nichols

Une autre méthode de réglage est la méthode dite de "Ziegler-Nichols", introduite par John G. Ziegler et Nathaniel B. Nichols. Elle commence comme la méthode précédente: réglage des gains I et D à zéro et accroissement du gain P jusqu'à ce que la sortie du procédé commence à osciller. Noter alors le gain critique (K_c) et la période d'oscillation de la sortie (P_c). Ajuster alors les termes P, I et D de la boucle comme sur la table ci-dessous:

Type de régulation	**P**	**I**	**D**
P	$.5K_c$		
PI	$.45K_c$	$P_c/1.2$	
PID	$.6K_c$	$P_c/2$	$P_c/8$

Deuxième partie

La logique Ladder

Chapitre 17

La programmation en Ladder

17.1 Introduction

La logique Ladder ou langage de programmation Ladder est une méthode pour tracer les schémas en logique électrique. Il s'agit maintenant d'un langage graphique vraiment populaire pour la programmation des automates programmables industriels (API). Il a été à l'origine inventé pour décrire la logique à relais. Son nom est fondé sur la constatation que les programmes dans cette langue ressemblent à une échelle (ladder), avec deux «rails» verticaux et, entre eux, une série «d'échelons». En Allemagne et ailleurs en Europe, le style consiste à placer les rails horizontaux, un en haut de la page et l'autre en bas avec les échelons verticaux dessinés séquentiellement de la gauche vers la droite.

Un programme en logique Ladder, également appelé schéma Ladder, est ressemblant au schéma d'un ensemble de circuits électriques à relais. C'est l'intérêt majeur du schéma Ladder de permettre à une large variété de personnels techniques, ingénieurs, techniciens électriciens, etc de le comprendre et de l'utiliser sans formation complémentaire grâce à cette ressemblance.

La logique Ladder est largement utilisée pour programmer les API, avec lesquels le contrôle séquentiel des processus de fabrication est requis. Le Ladder est utile pour les systèmes de contrôle simples mais critiques, ou pour reprendre d'anciens circuits à relais câblés. Comme les contrôleurs à logique programmable sont devenus plus sophistiqués, ils ont aussi été utilisés avec succès dans des systèmes d'automatisation très complexes.

Le langage Ladder peut être considéré comme un langage basé sur les règles, plutôt que comme un langage procédural. Un «échelon» en Ladder représente une règle. Quand elles sont mises en application avec des éléments électromécaniques, les diverses règles «s'exécutent» toutes simultanément et immédiatement. Quand elle sont mises en application dans la logique d'un automate programmable, les règles sont exécutées séquentiellement par le logiciel, dans une boucle. En exécutant la boucle assez rapidement, typiquement plusieurs fois par seconde, l'effet d'une exécution simultanée et immédiate est obtenu.

17.2 Exemple

Les composants les plus communs du Ladder sont les contacts (entrées), ceux-ci sont habituellement NC (normalement clos) ou NO (normalement ouvert) et les bobines (sorties).

– Le contact NO

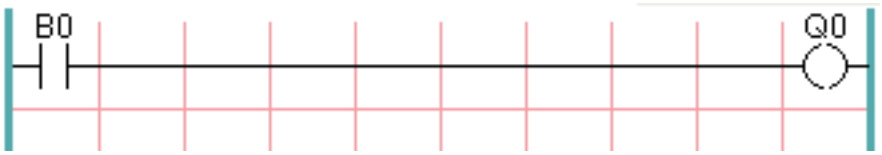
– Le contact NC

– La bobine (sortie)

Bien sûr, il y a beaucoup plus de composants dans le langage Ladder complet, mais la compréhension de ceux-ci aidera à appréhender le concept global du langage.

L’échelle se compose d’un ou plusieurs échelons. Ces échelons sont tracés horizontalement, avec les composants placés sur eux (entrées, sorties et autres), les composants sont évalués de la gauche vers la droite.

Cet exemple est un simple échelon:



L’entrée B0 sur la gauche et un contact normalement ouvert, il est connecté sur la sortie Q0 sur la droite. Imaginez maintenant qu’une tension soit appliquée à l’extrême gauche, dès que B0 devient vraie (par exemple: l’entrée est activée, ou l’utilisateur a pressé le contact NO), la tension atteint l’extrême droite en traversant la bobine Q0. Avec comme conséquence que la sortie Q0 passe de 0 à 1.

Chapitre 18

Classicladder Programming

18.1 Ladder Concepts

Classic Ladder 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.

18.2 Languages

The most common language used when working with Classic Ladder is *ladder*. Classic Ladder also supports Sequential Function Chart (Grafcet).

18.3 Components

There are 2 components to Classic Ladder.
– The real time module classicladder_rt
– The user space module (including a GUI) classicladder

18.3.1 Files

Typically classic ladder 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.

Ladder files (.clp) must not contain any blank spaces in the name.

18.3.2 Realtime Module

Loading the Classic Ladder real time module (classicladder_rt) is possible from a HAL file, or directly using a halcmd instruction. The first line loads real time the Classic Ladder module. The second line adds the function classicladder.0.refresh to the servo thread. This line makes Classic Ladder update at the servo thread rate.

```
loadrt classicladder_rt
addf classicladder.0.refresh servo-thread
```

The speed of the thread that Classic Ladder is running in directly affects the responsiveness to inputs and outputs. If you can turn a switch on and off faster than Classic Ladder can notice it then you may need to speed up the thread. The fastest that Classic Ladder 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 Classic Ladder 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.

18.3.3 Variables

It is possible to configure the number of each type of ladder object while loading the Classic Ladder real time module. If you do not configure the number of ladder objects Classic Ladder will use the default values.

TABLE 18.1: Default Variable Count

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
```

18.4 Loading the Classic Ladder user module

Classic Ladder 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 Classic Ladder HAL commands in the custom.hal file.

To load the user module:

```
loadusr classicladder
```

To load a ladder file:

```
loadusr classicladder myladder.clp
```

Classic Ladder 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 Classic Ladder with HAL without EMC:

```
loadusr -w classicladder
```

The -w tells HAL not to close down the HAL environment until Classic Ladder is finished.

If you first load ladder program with the *--nogui* option then load Classic Ladder again with no options the GUI will display the last loaded ladder program.

In AXIS you can load the GUI from File/Ladder Editor. . .

18.5 Classic Ladder GUI

If you load Classic Ladder with the GUI it will display two windows: section display, and section manager.

18.5.1 Sections Manager

When you first start up Classic Ladder you get an empty Sections Manager window.

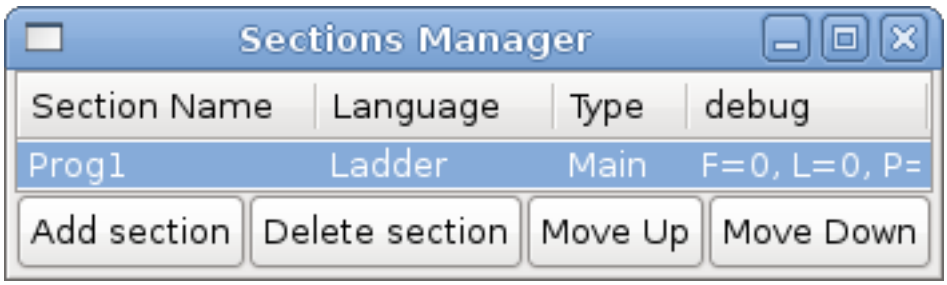


FIGURE 18.1 – Sections Manager Default Window

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.

18.5.2 Section Display

When you first start up Classic Ladder you get an empty Section Display window. Displayed is one empty rung.

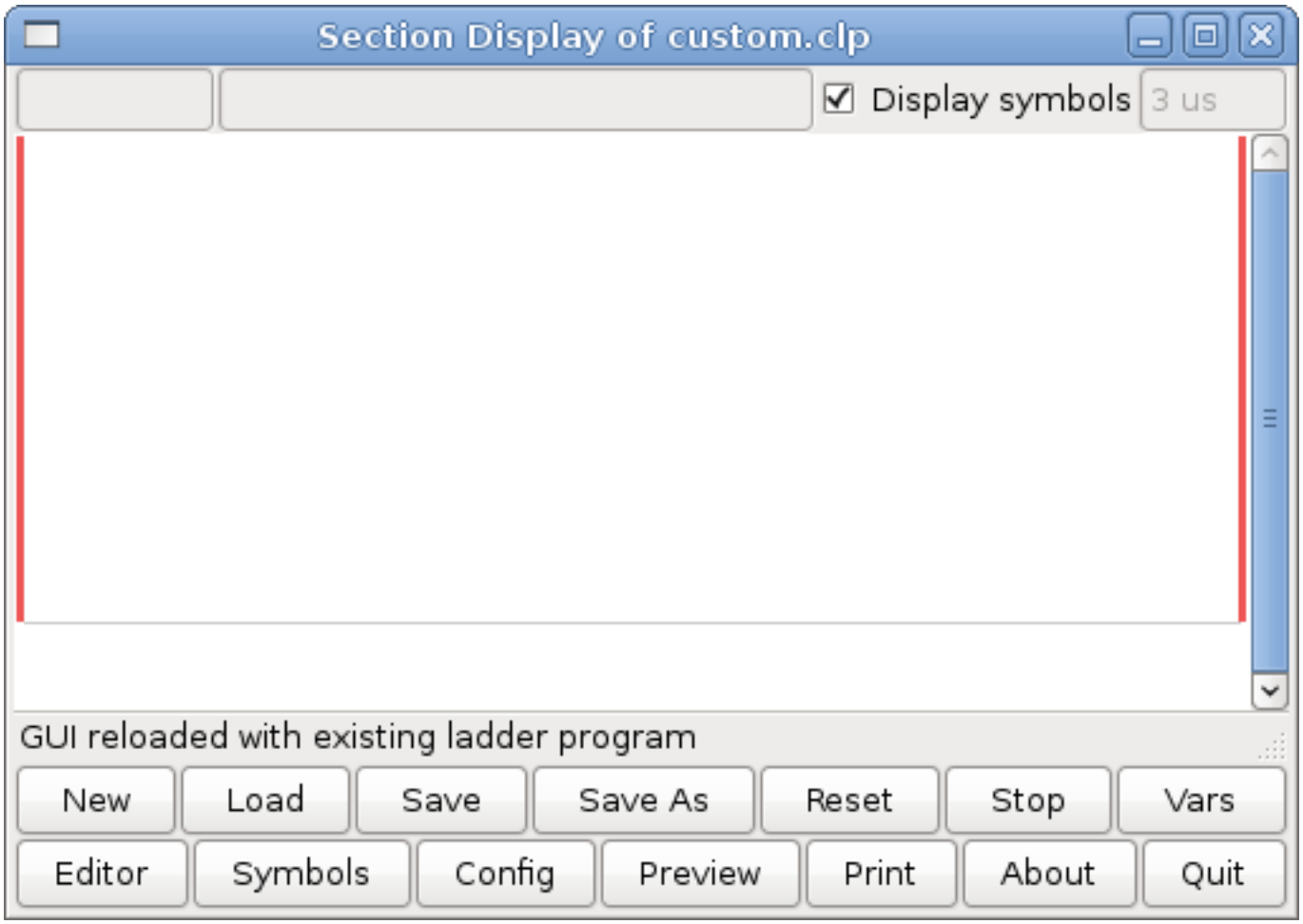


FIGURE 18.2 – Section Display Default Window

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 user program meaning Modbus and the display. The real time 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 didn't 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.

18.5.3 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.

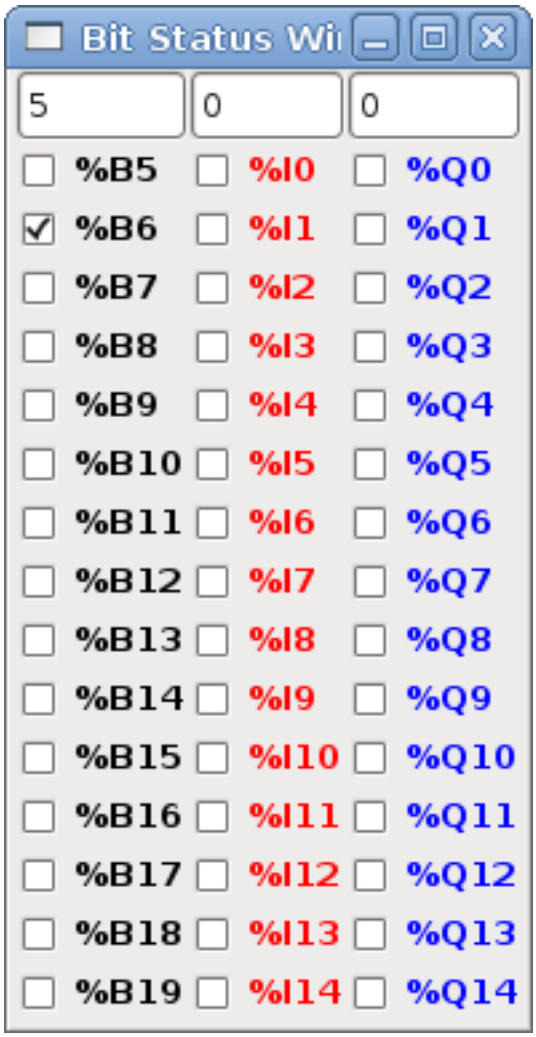


FIGURE 18.3 – Bit Status Window

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 Classic Ladder is running can not be changed and will be displayed as checked if on and unchecked if off.

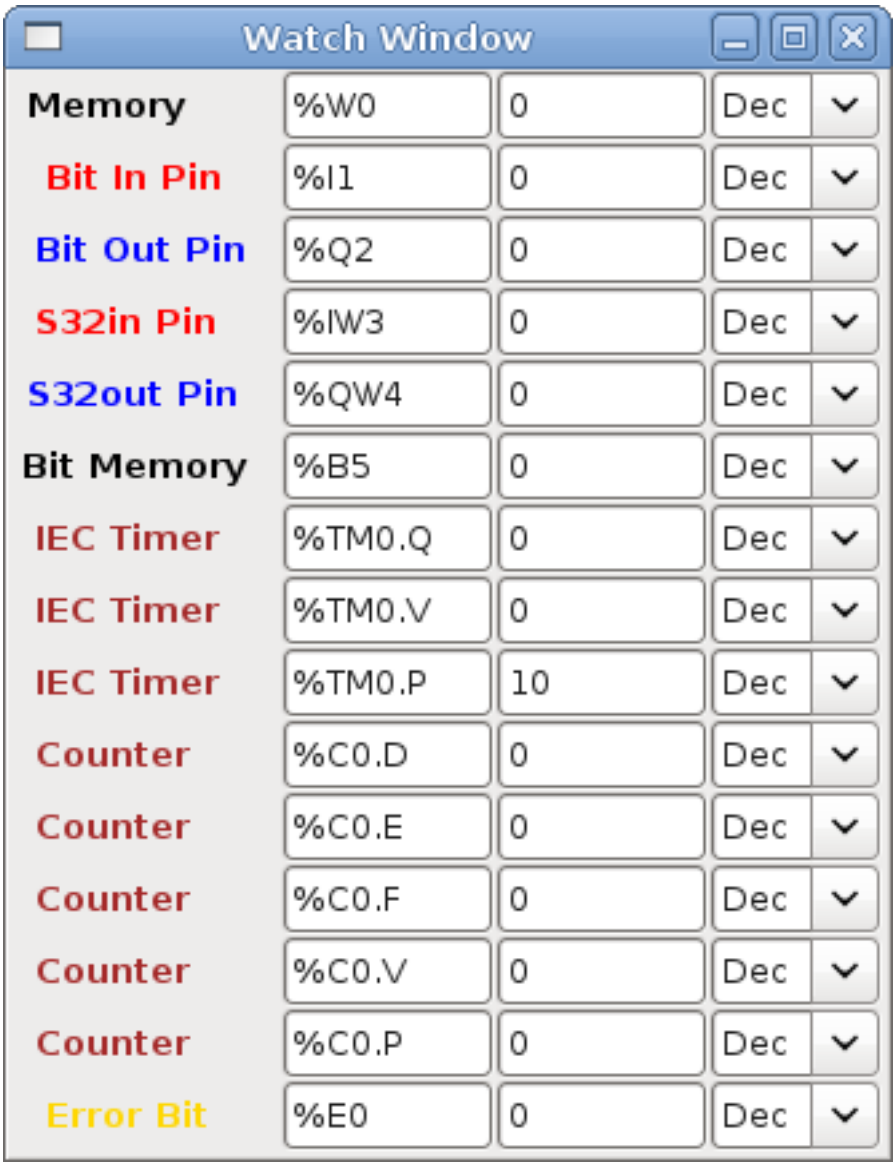


FIGURE 18.4 – Watch Window

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.

18.5.4 Symbol Window

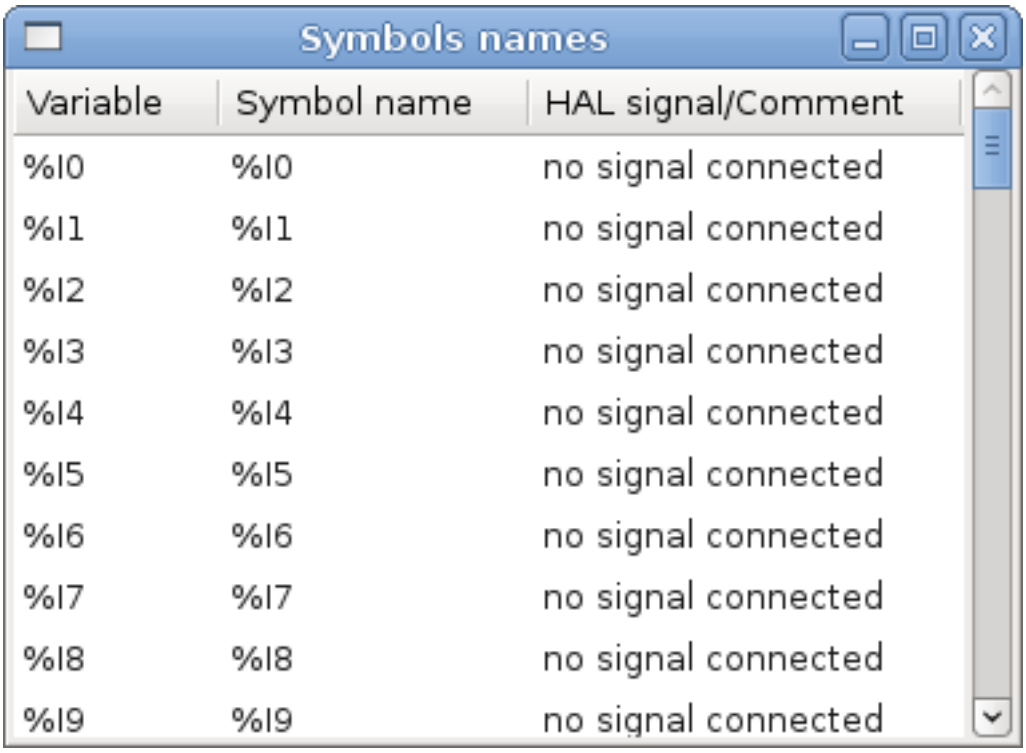


FIGURE 18.5 – Symbol Names window

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!

18.5.5 The Editor window

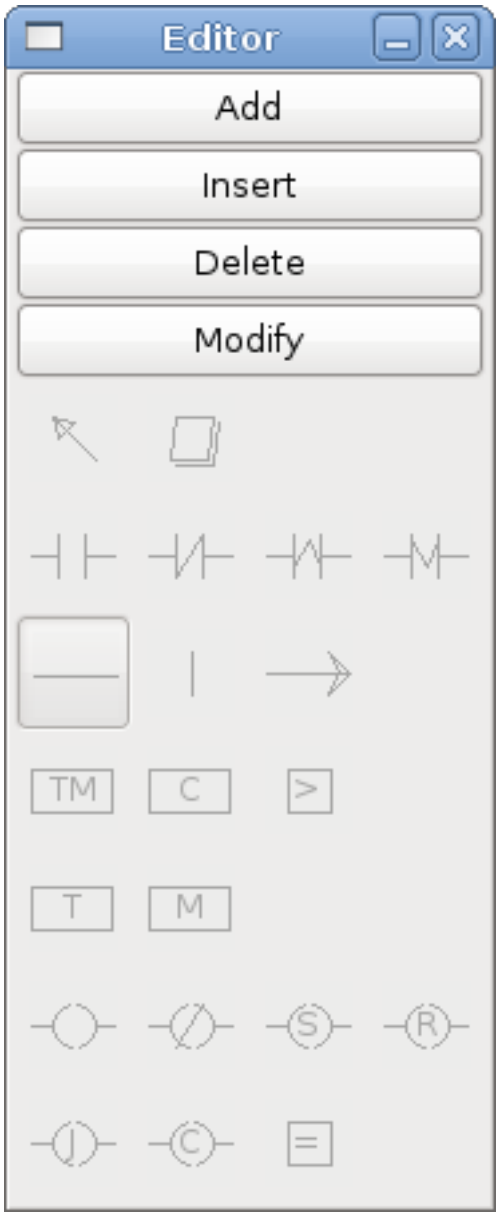


FIGURE 18.6 – Editor Window

– *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. (eg %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. (eg %W2=7 or %W1=%W2) ASSIGNMENT functions can only be placed at the right most side of the section display.

18.5.6 Config Window

The config window shows the current project status and has the Modbus setup tabs.

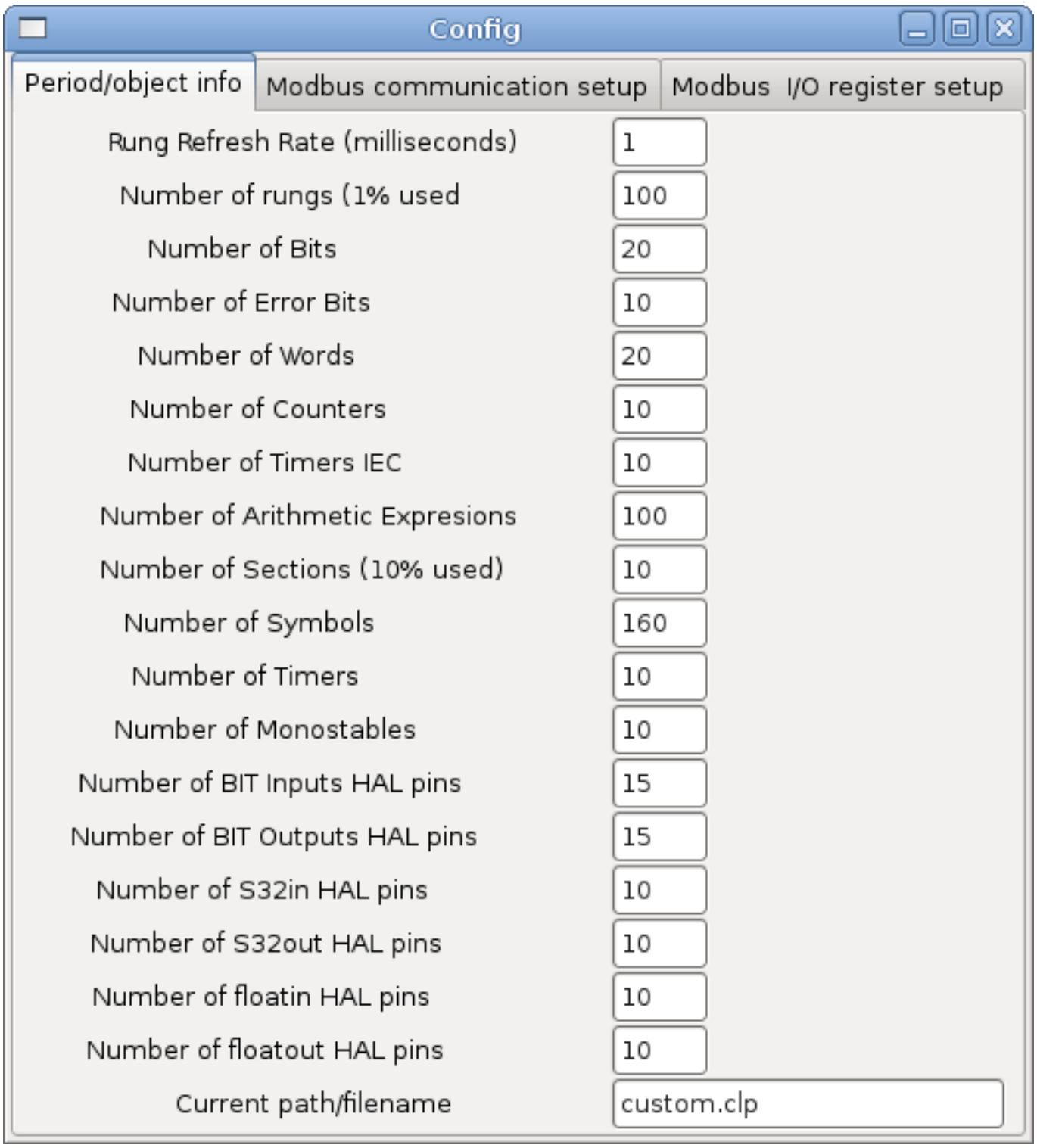


FIGURE 18.7 – Config Window

18.6 Ladder objects

18.6.1 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 eg. %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.

18.6.2 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 100ms, 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)

18.6.3 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)

18.6.4 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)

18.6.5 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.
– *%Cxx.D* - Counter xx done (Boolean, read only)
– *%Cxx.E* - Counter xx empty overflow (Boolean, read only)
– *%Cxx.F* - Counter xx full overflow (Boolean, read only)
– *%Cxx.V* - Counter xx current value (integer, read or write)
– *%Cxx.P* - Counter xx preset (integer, read or write)

18.6.6 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)
– (,) grouping
– ˆ (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.

18.6.7 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
```

Note
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.

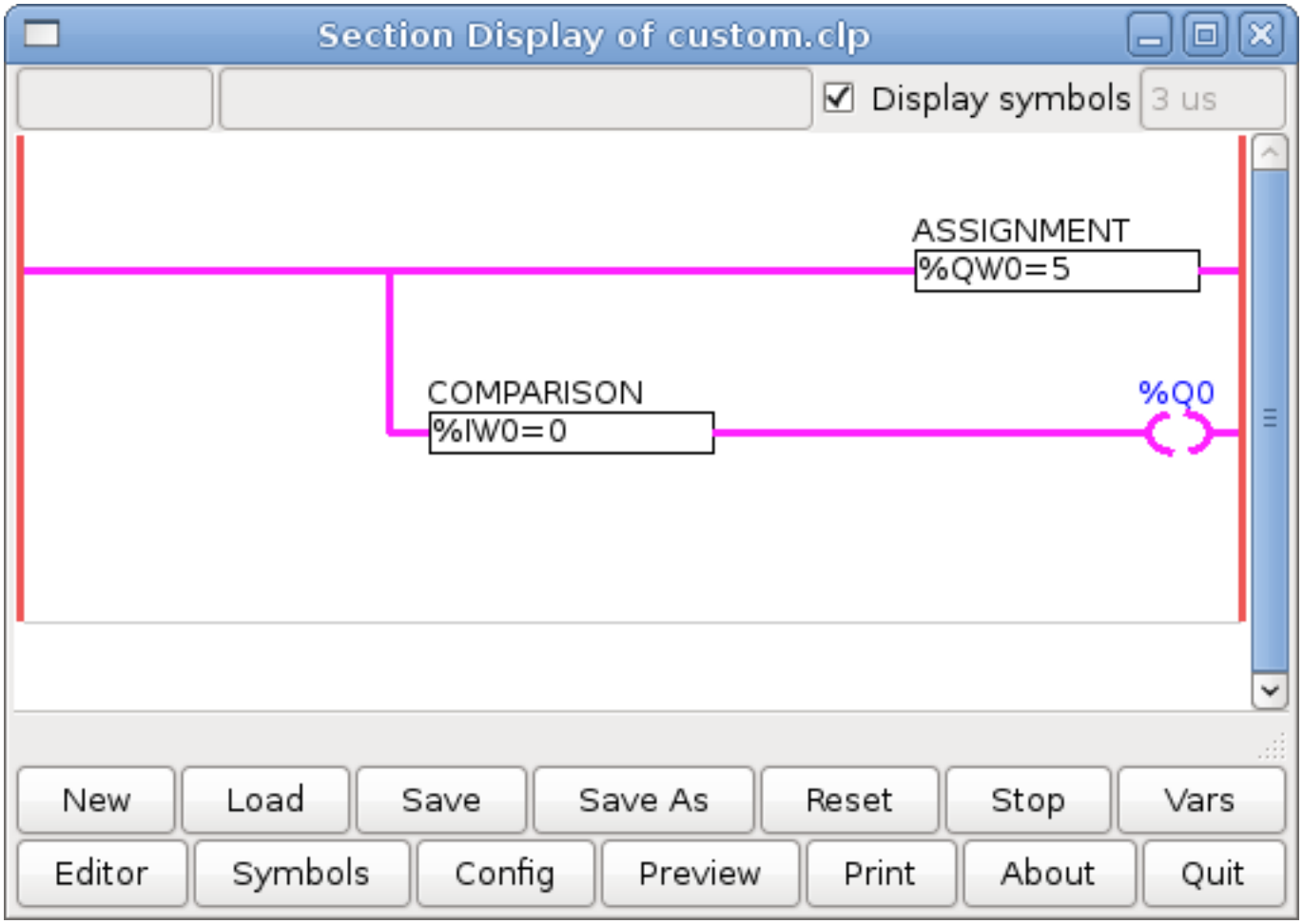


FIGURE 18.8 – Assign/Compare Example

18.6.8 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 eg. %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 it's N.O. contact will be closed (on, true, etc)
– *N.C. COIL* - (a relay coil that inverses its contacts.) When coil is energized it"s N.O. contact will be open (off, false, etc)
– *SET COIL* - (a relay coil with latching contacts) When coil is energized it's N.O. contact will be latched closed.
– *RESET COIL* - (a relay coil with latching contacts) When coil is energized It's N.0. contact will be latched open.
– *JUMP COIL* - (a *goto* coil) when coil is energized 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 program jumps to a subroutine section designated by a subroutine number -subroutines are designated SR0 to SR9 (designate them in the section manager)

AVERTISSEMENT
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!

18.6.8.1 JUMP COIL

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.

18.6.8.2 CALL COIL

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.

18.7 Classic Ladder 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.

List of variables :
- *%Bxxx* - Bit memory xxx (Boolean)
- *%Wxxx* - Word memory xxx (32 bits signed integer)
- *%IWxxx* - Word memory xxx (S32 in pin)
- *%QWxxx* - Word memory xxx (S32 out pin)
- *%IFxx* - Word memory xx (Float in pin) (**converted to S32 in Classic Ladder**)
- *%QFxx* - Word memory xx (Float out pin) (**converted to S32 in Classic Ladder**)
- *%Txx.R* - Timer xx running (Boolean, user read only)
- *%Txx.D* - Timer xx done (Boolean, user read only)
- *%Txx.V* - Timer xx current value (integer, user read only)
- *%Txx.P* - Timer xx preset (integer)
- *%TMxxx.Q* - Timer xxx done (Boolean, read write)
- *%TMxxx.P* - Timer xxx preset (integer, read write)
- *%TMxxx.V* - Timer xxx value (integer, read write)
- *%Mxx.R* - Monostable xx running (Boolean)
- *%Mxx.V* - Monostable xx current value (integer, user read only)
- *%Mxx.P* - Monostable xx preset (integer)
- *%Cxx.D* - Counter xx done (Boolean, user read only)
- *%Cxx.E* - Counter xx empty overflow (Boolean, user read only)
- *%Cxx.F* - Counter xx full overflow (Boolean, user read only)
- *%Cxx.V* - Counter xx current value (integer)
- *%Cxx.P* - Counter xx preset (integer)
- *%Ixxx* - Physical input xxx (Boolean) (HAL input bit)
- *%Qxxx* - Physical output xxx (Boolean) (HAL output bit)
- *%Xxxx* - Activity of step xxx (sequential language)
- *%Xxxx.V* - Time of activity in seconds of step xxx (sequential language)
- *%Exx* - 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

18.8 GRAFCET Programming

AVERTISSEMENT
This is probably the least used and most poorly understood feature of Classic Ladder. 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 image: 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 [show sequential program]
– *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* - This 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 %Xxxx (eg. %X5) is used to see if a step is active. The variable %Xxxx.V (eg. %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 (eg. %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.

18.9 Modbus

Things to consider:
– Modbus is a userspace 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 Classic Ladder 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 Classic Ladder userspace program.

Loading Modbus

```
loadusr -w classicladder --modmaster myprogram.clp
```

The -w makes HAL wait until you close Classic Ladder before closing realtime session. Classic Ladder 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 Classic Ladder user program this page will not be displayed.

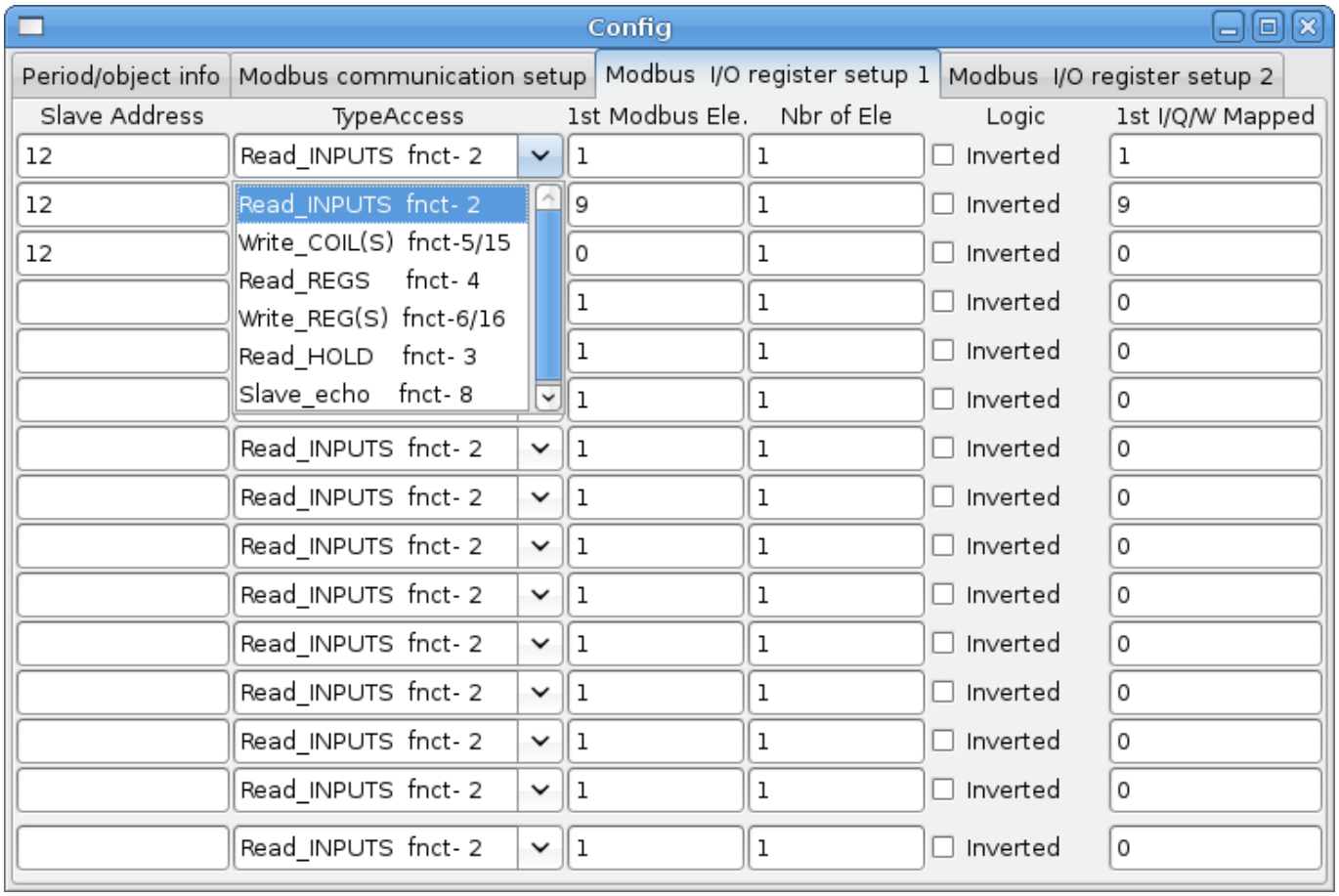


FIGURE 18.9 – Config I/O

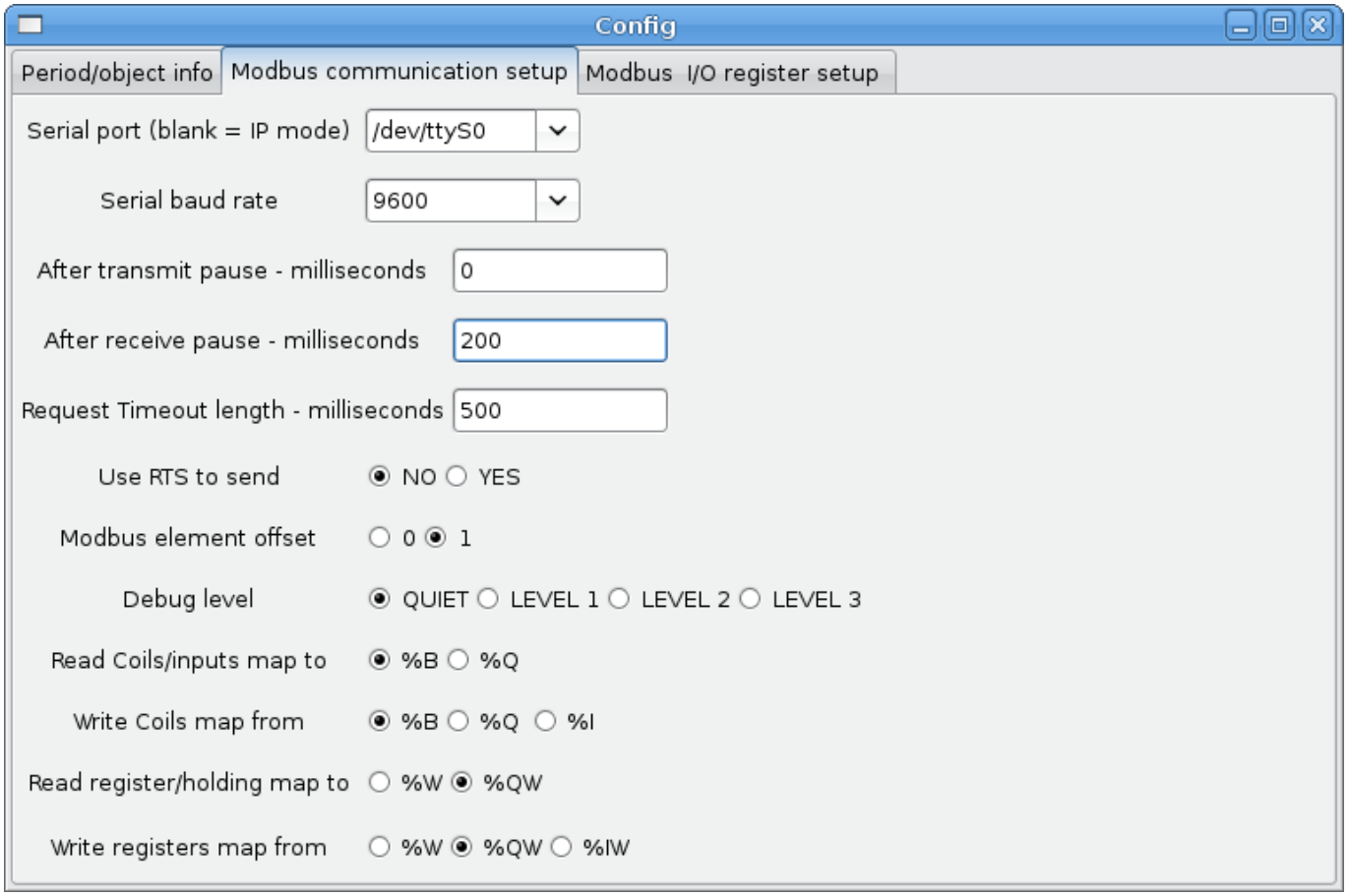


FIGURE 18.10 – Config Coms

– *SERIAL PORT* - For IP blank. For serial the location/name of serial driver eg. /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 Classic Ladder'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 Classic Ladder's %Q variables starting at %Q9 ending at %Q10.

The third line is set to write 2 registers (16 bits each) starting at the 0th register number (register 0) so register numbers 0-1 are mapped onto Classic Ladder'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 Classic Ladder running in a terminal to see the message.

18.9.1 MODBUS Settings

Serial:
– Classic Ladder 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. Classic Ladder 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).

18.9.2 MODBUS Info

– Classic Ladder 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 parity is used.
– 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, http://www.advantech.com) and a serial Modbus/RTU one (http://www.ipac.ws).
– See examples: adam-6051 and modbus_rtu_serial.
– Web links: http://www.modbus.org and this interesting one: http://www.iatips.com/modbus.html
– MODBUS TCP SERVER INCLUDED
– Classic Ladder 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.

More information on modbus protocol is available on the internet.

http://www.modbus.org/

18.9.3 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.

18.9.4 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 Classic Ladder 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 user 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.

18.10 Setting up Classic Ladder

In this section we will cover the steps needed to add Classic Ladder to a Stepconf Wizard generated config. On the advanced Configuration Options page of Stepconf Wizard check off "Include Classic Ladder PLC".

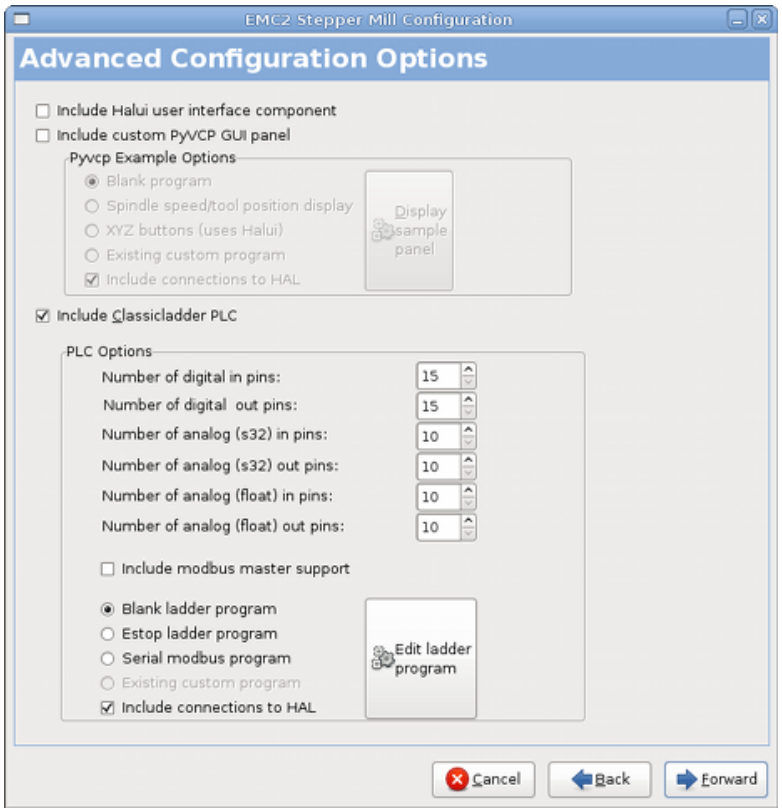


FIGURE 18.11 – Stepconf Classic Ladder

18.10.1 Add the Modules

If you used the Stepconf Wizard to add Classic Ladder you can skip this step.

To manually add Classic Ladder you must first add the modules. This is done by adding a couple of lines to the custom.hal file.

This line loads the real time module:

```
loadrt classicladder_rt
```

This line adds the Classic Ladder function to the servo thread:

```
addf classicladder.0.refresh servo-thread
```

18.10.2 Adding Ladder Logic

Now start up your config and select "File/Ladder Editor" to open up the Classic Ladder 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.

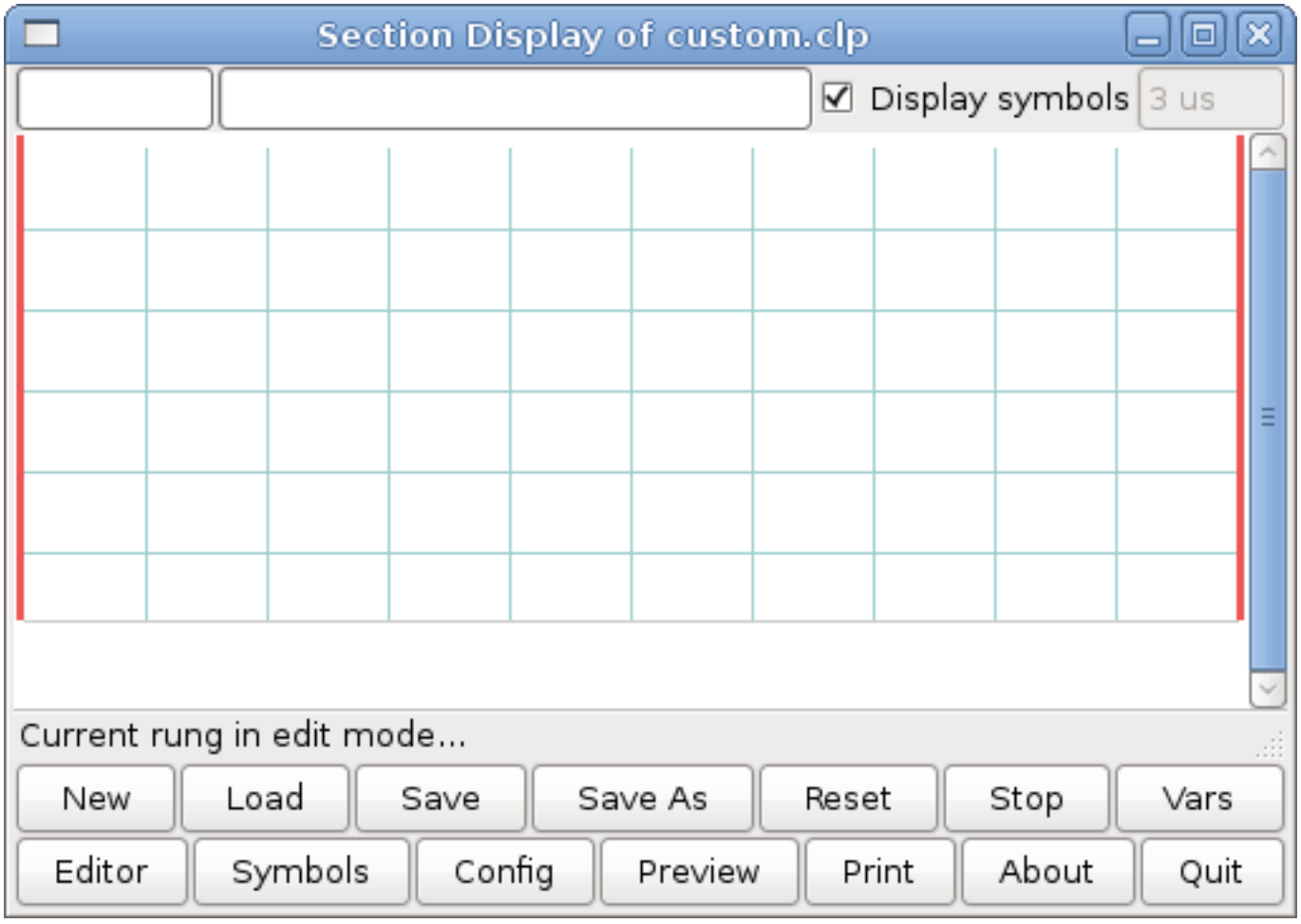


FIGURE 18.12 – Section Display with Grid

Now click on the N.O. Input in the Editor Window.

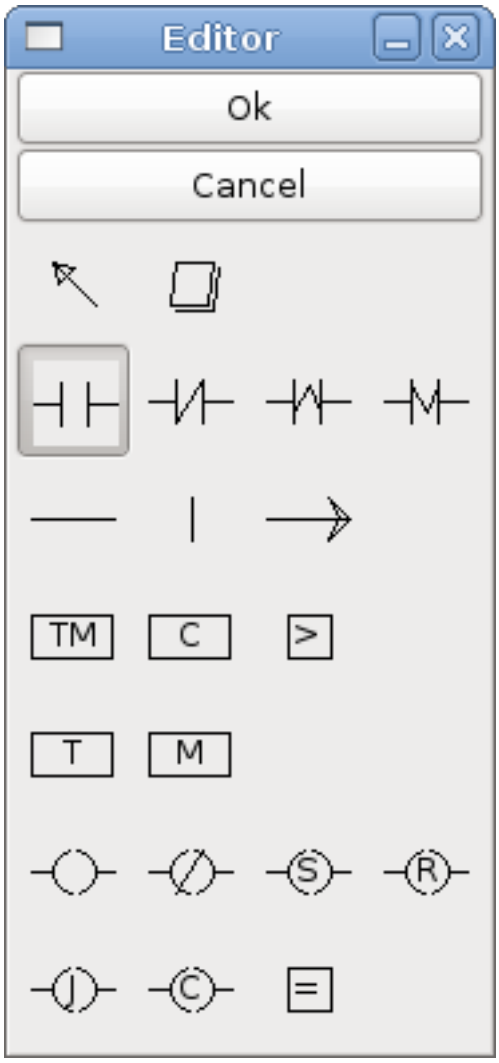


FIGURE 18.13 – Editor Window

Now click in the upper left grid to place the N.O. Input into the ladder.

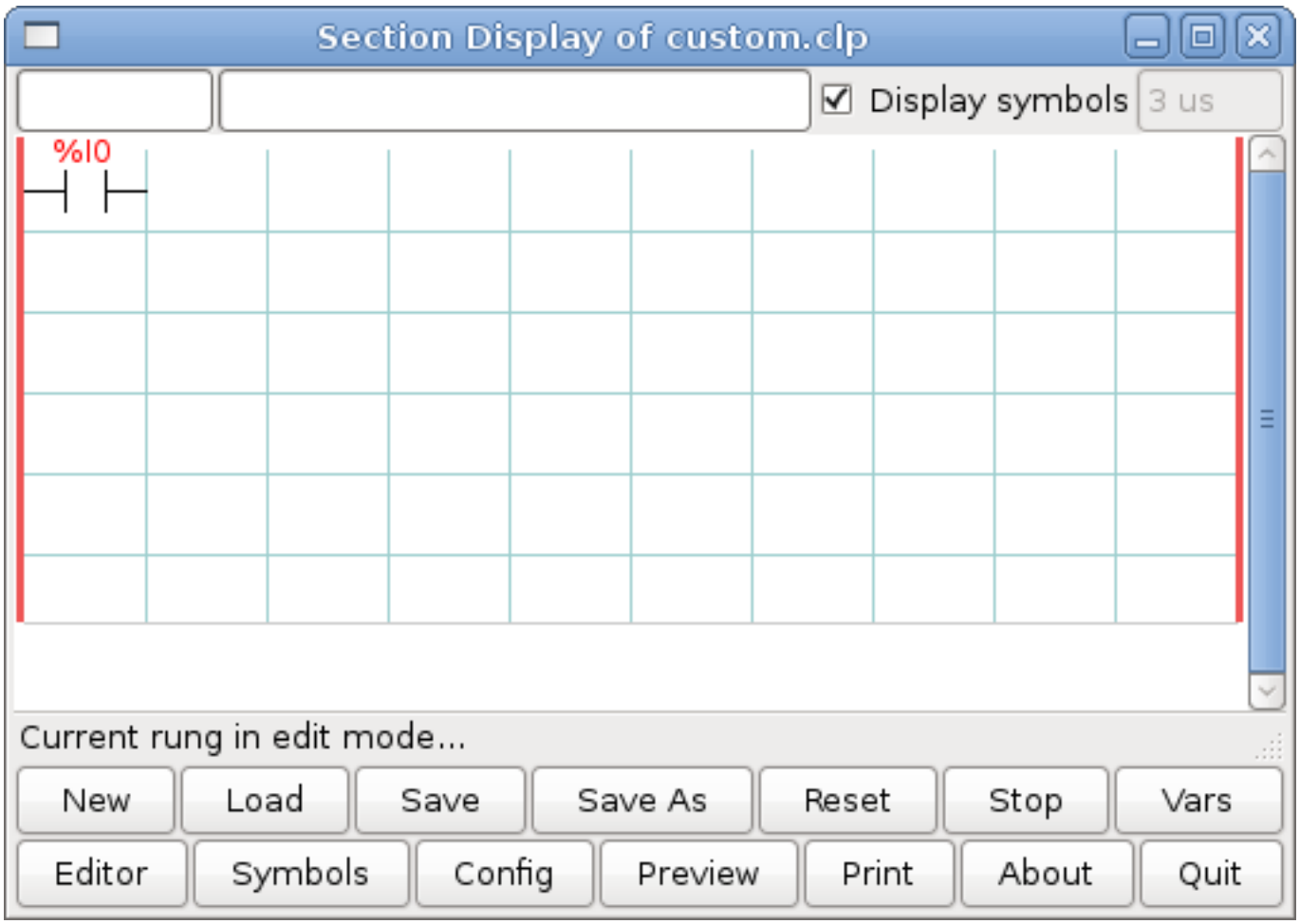


FIGURE 18.14 – Section Display with Input

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.

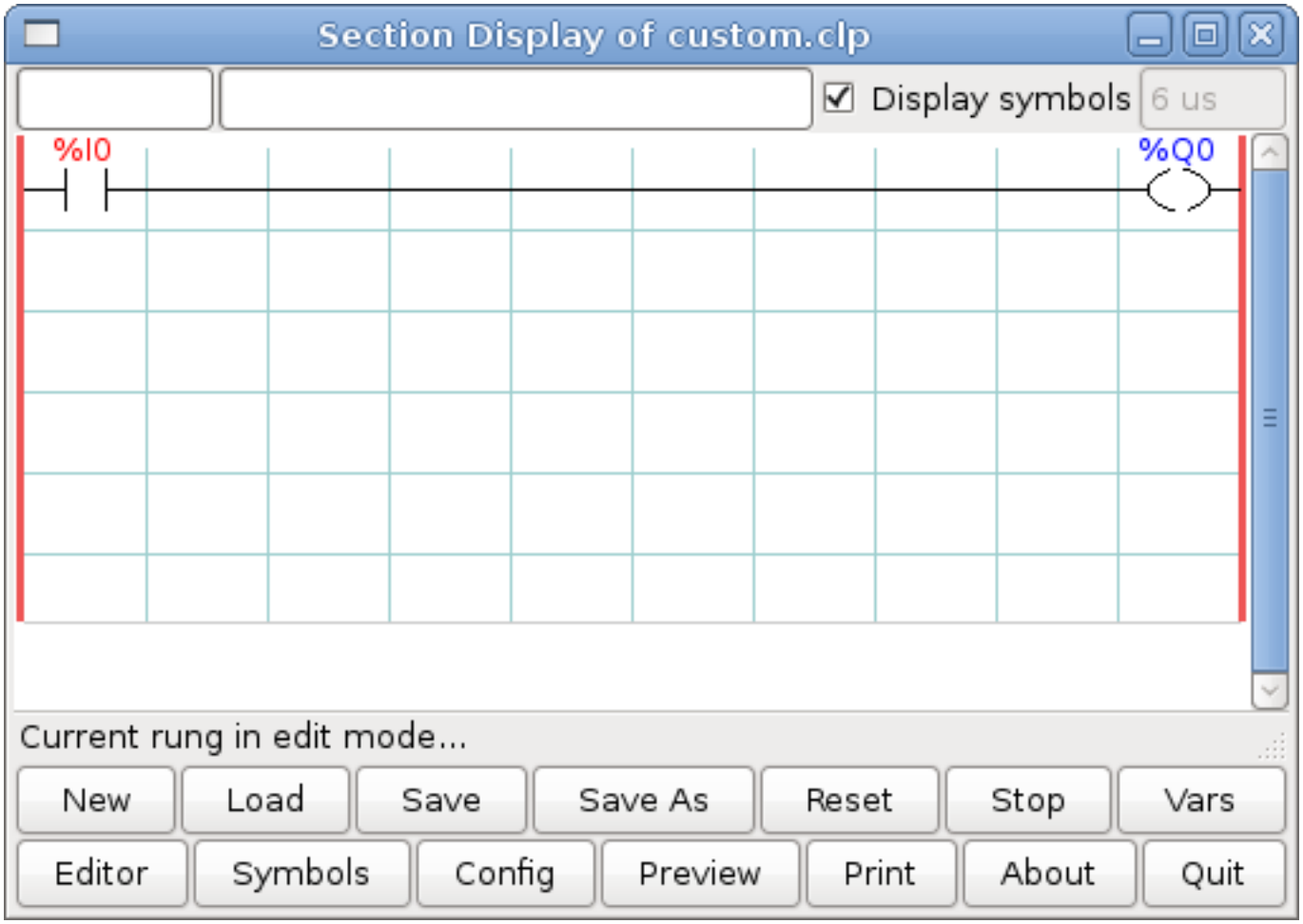


FIGURE 18.15 – Section Display with Rung

Now click on the OK button in the Editor window. Now your Section Display should look like this.

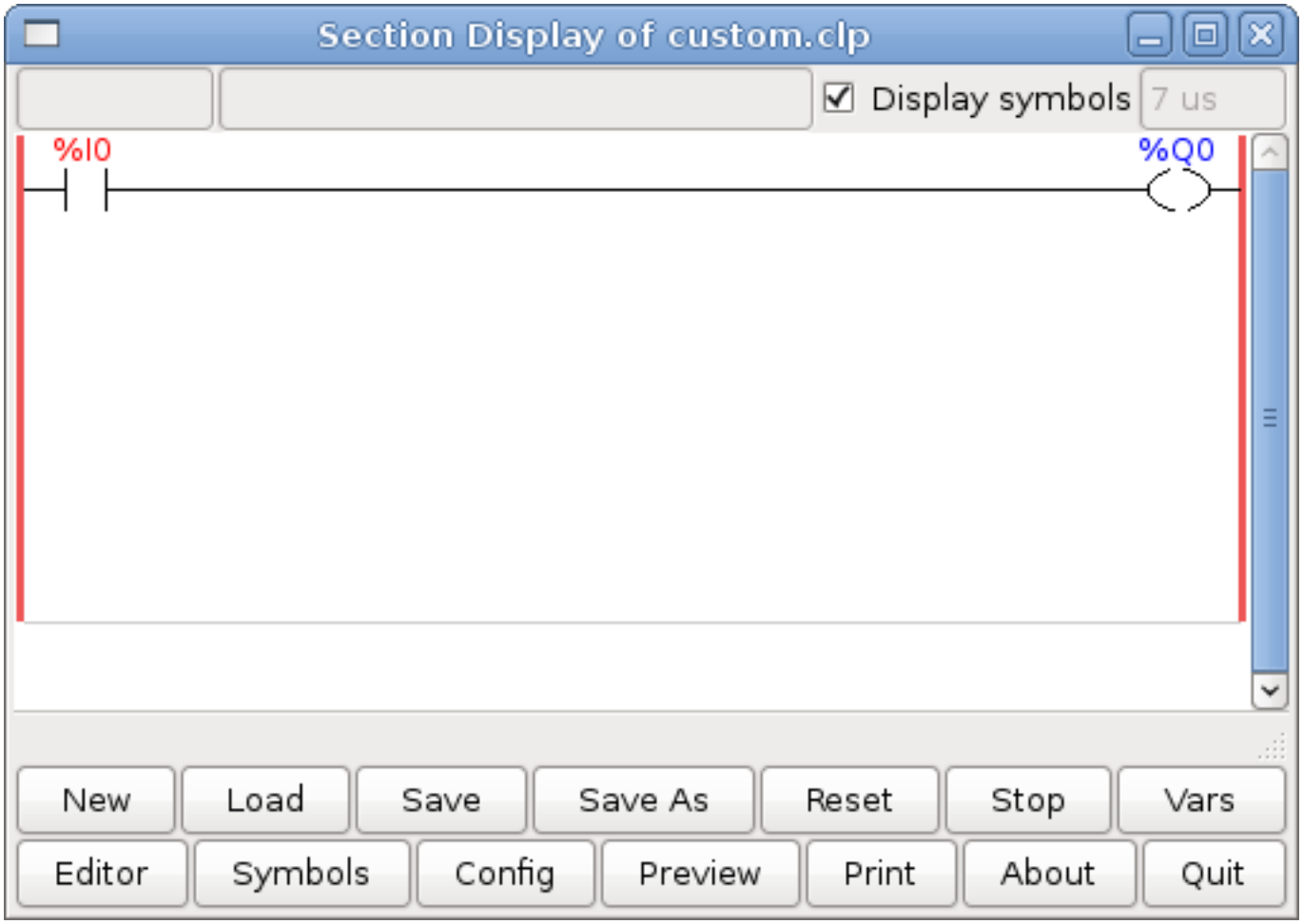


FIGURE 18.16 – Section Display Finished

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.

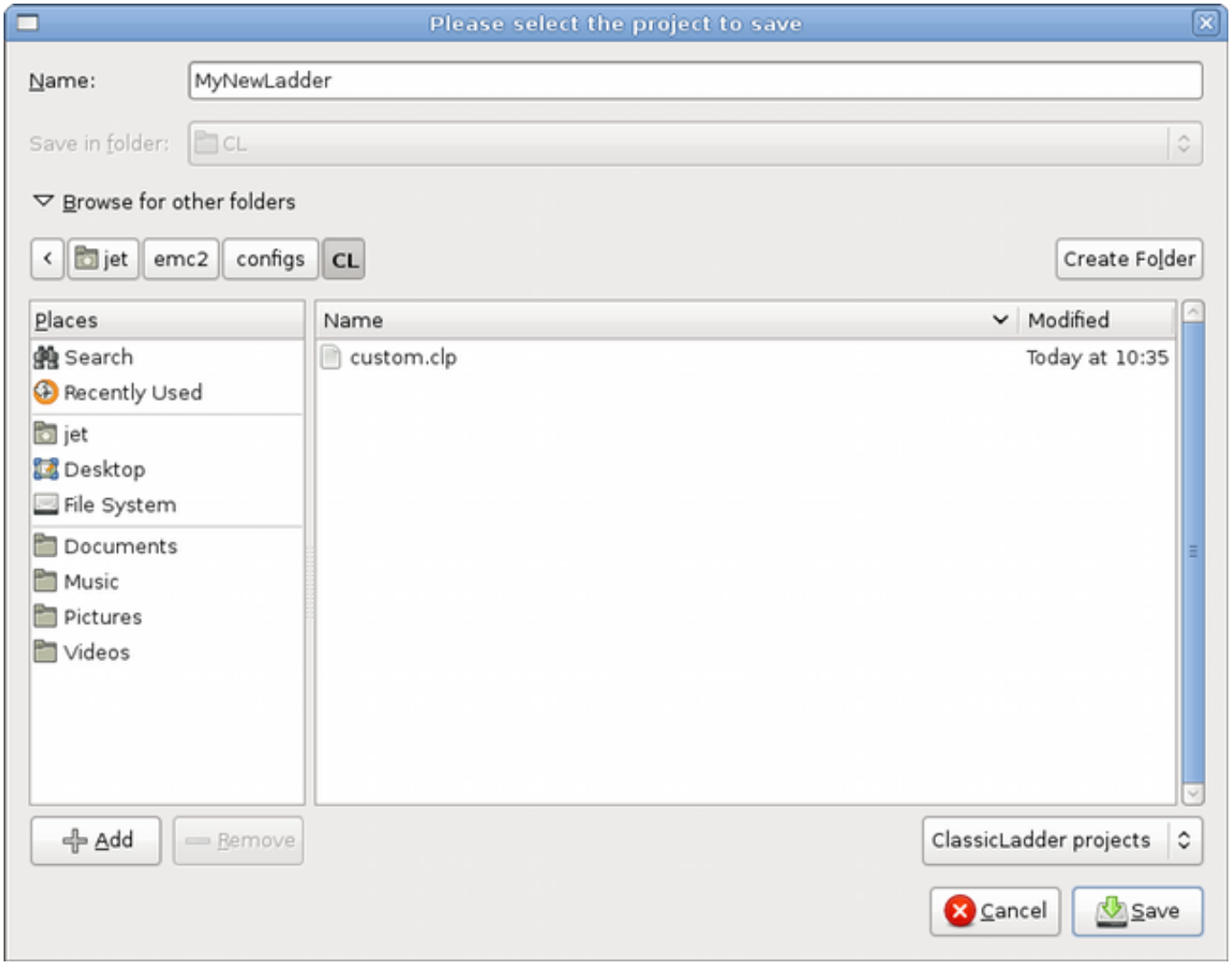


FIGURE 18.17 – Save As Dialog

Again if you used the Stepconf Wizard to add Classic Ladder 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.

18.11 Ladder Examples

18.11.1 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.

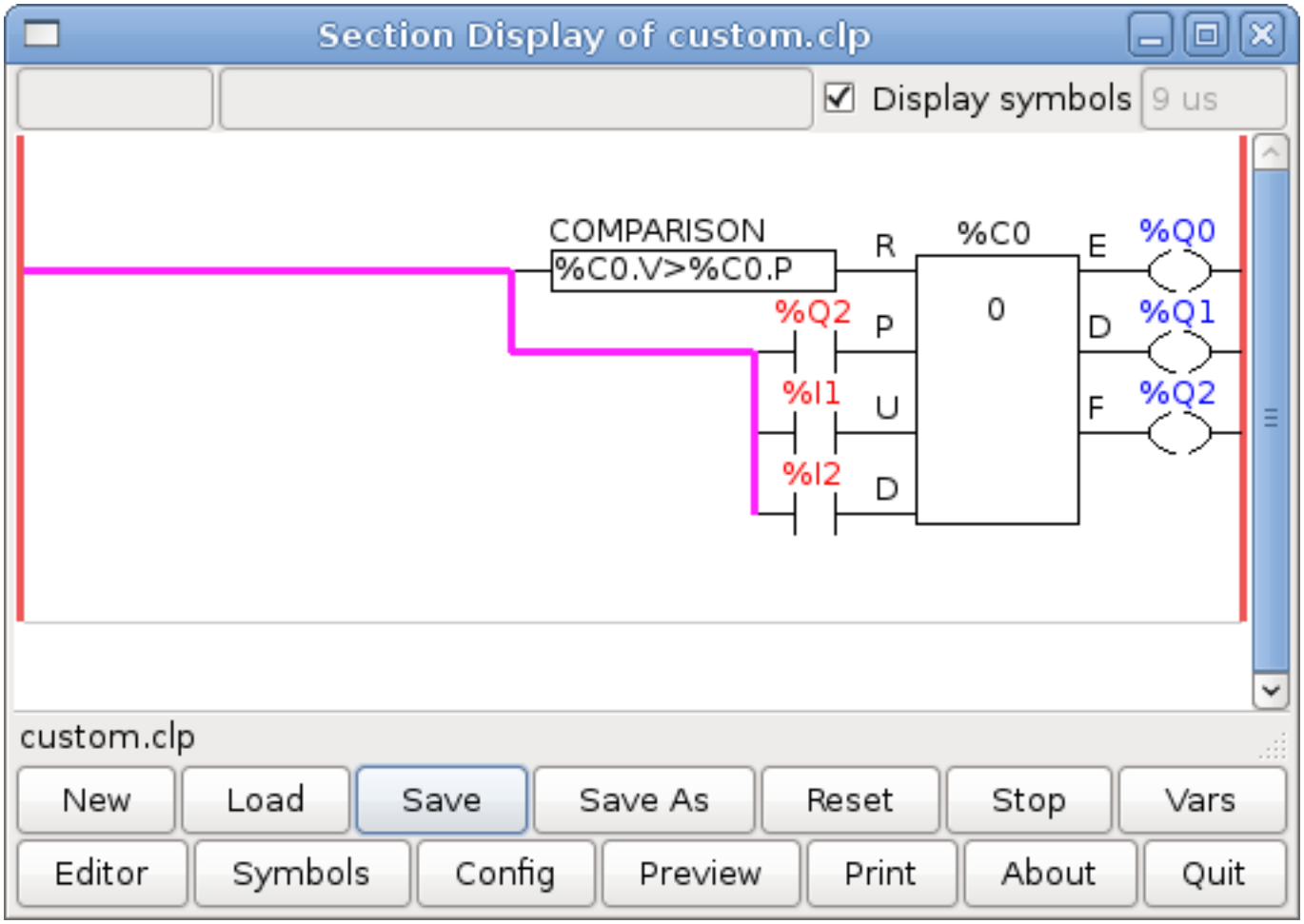


FIGURE 18.18 – Wrapping Counter

18.11.2 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.

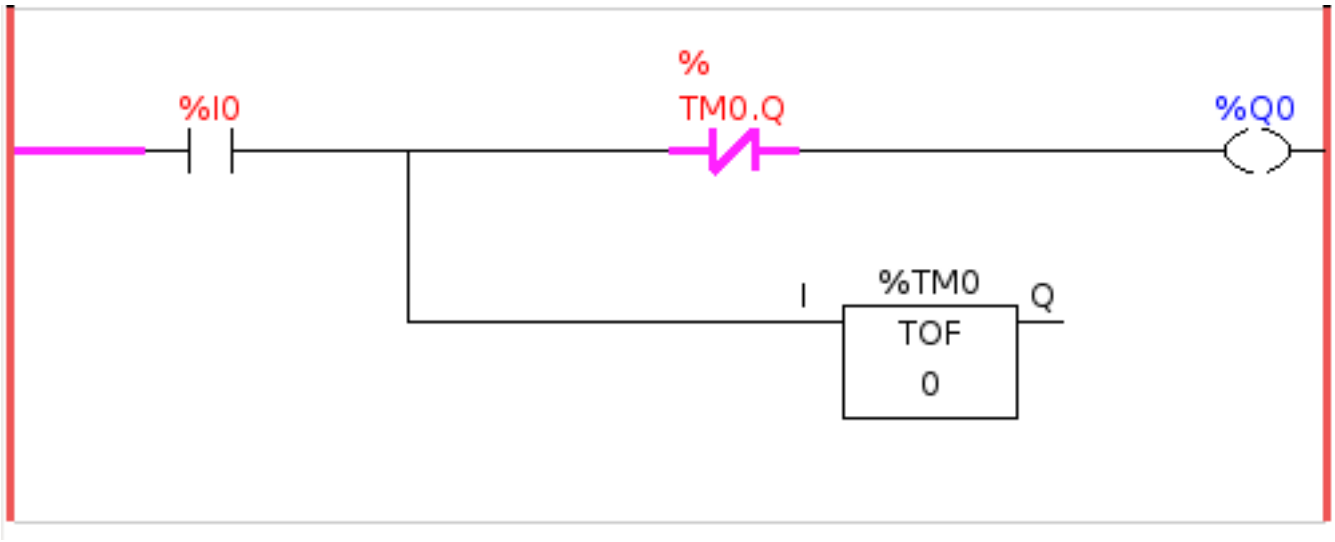


FIGURE 18.19 – Reject Extra Pulse

18.11.3 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 Classic Ladder.

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 Classic Ladder 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.

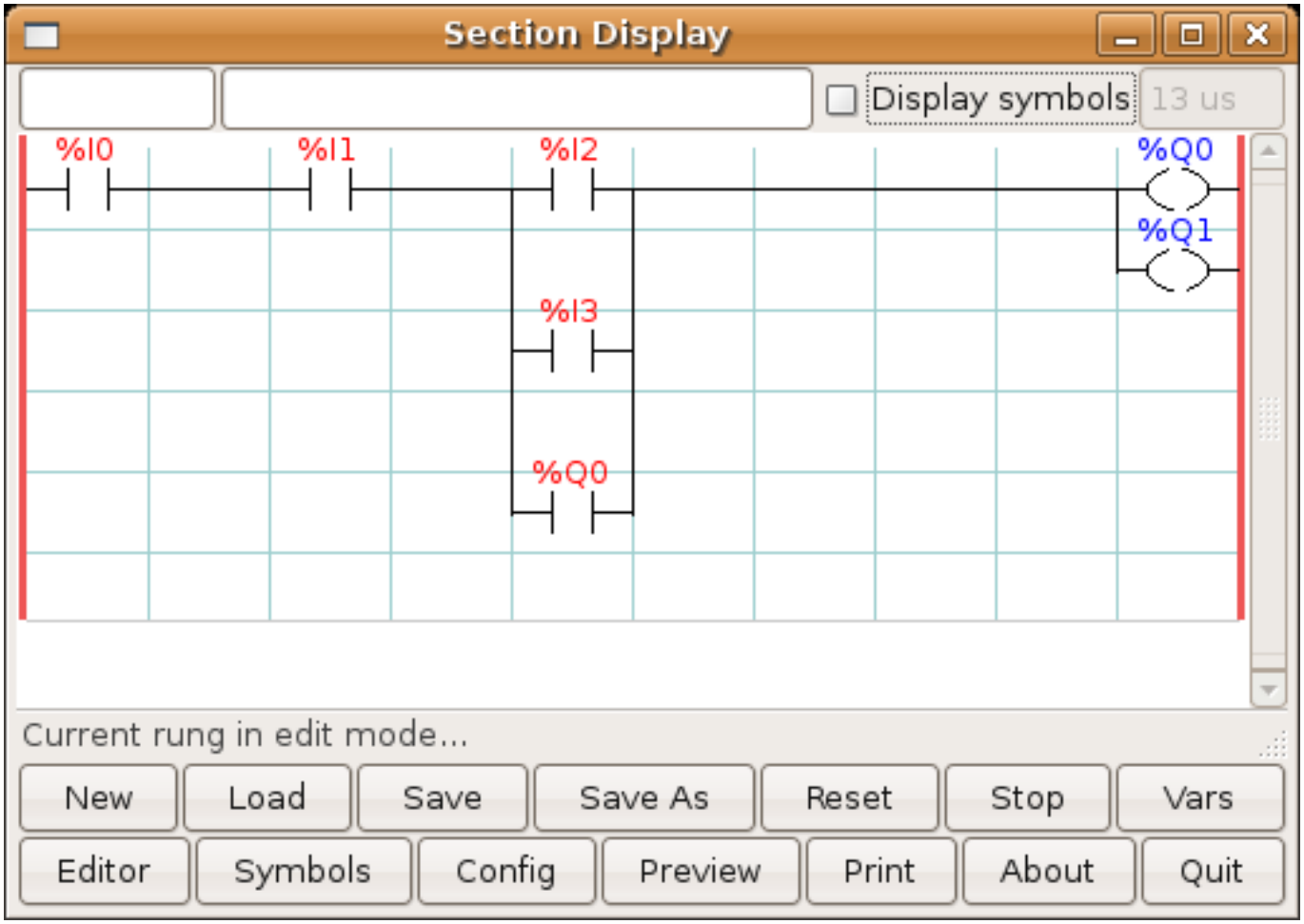


FIGURE 18.20 – E-Stop Section Display

After building the ladder select Save As and save the ladder as estop.clp

Now add the following line to your custom.hal file.

```
# 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 = Ouput 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 Classic Ladder
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.

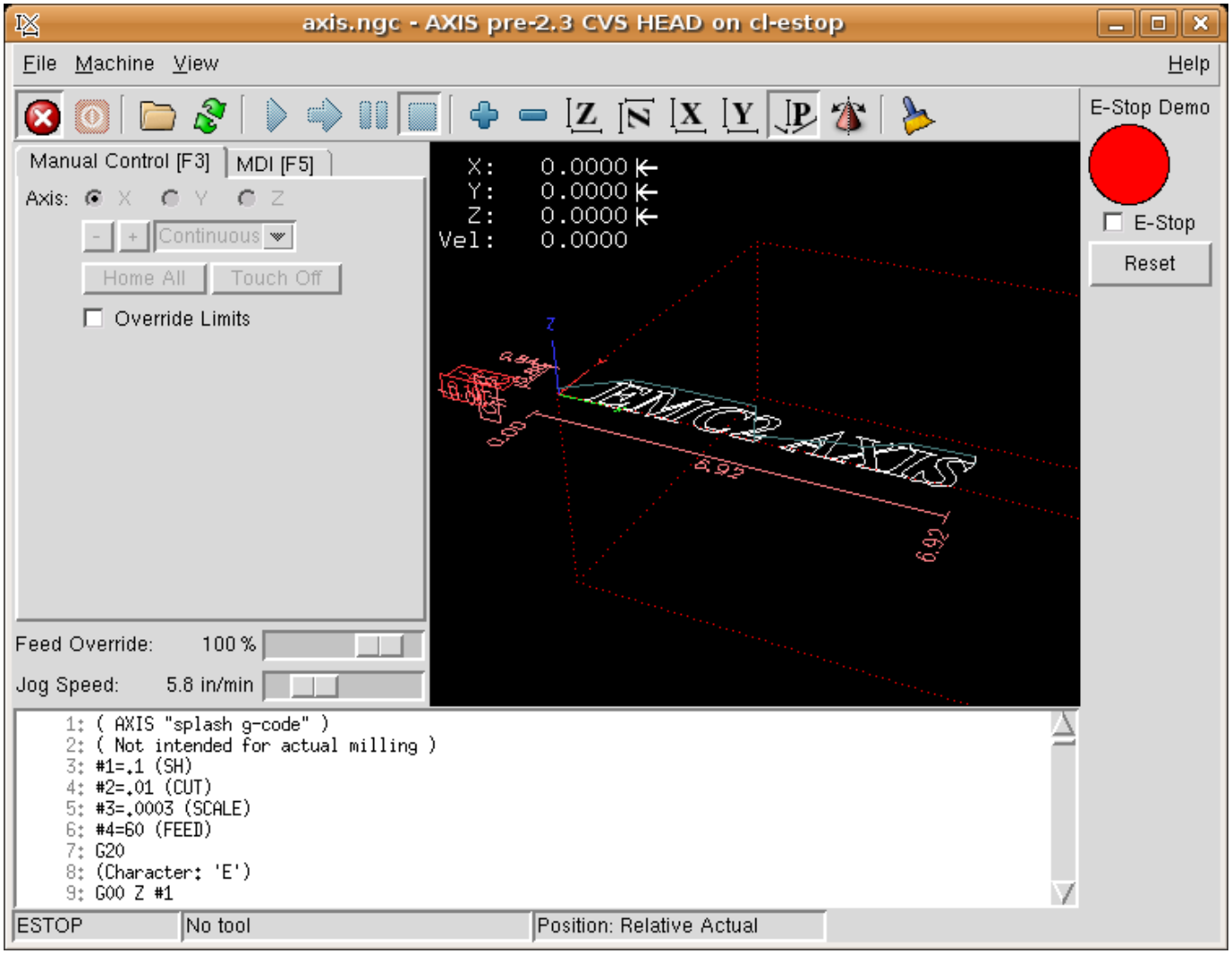


FIGURE 18.21 – AXIS E-Stop

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.

18.11.4 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.

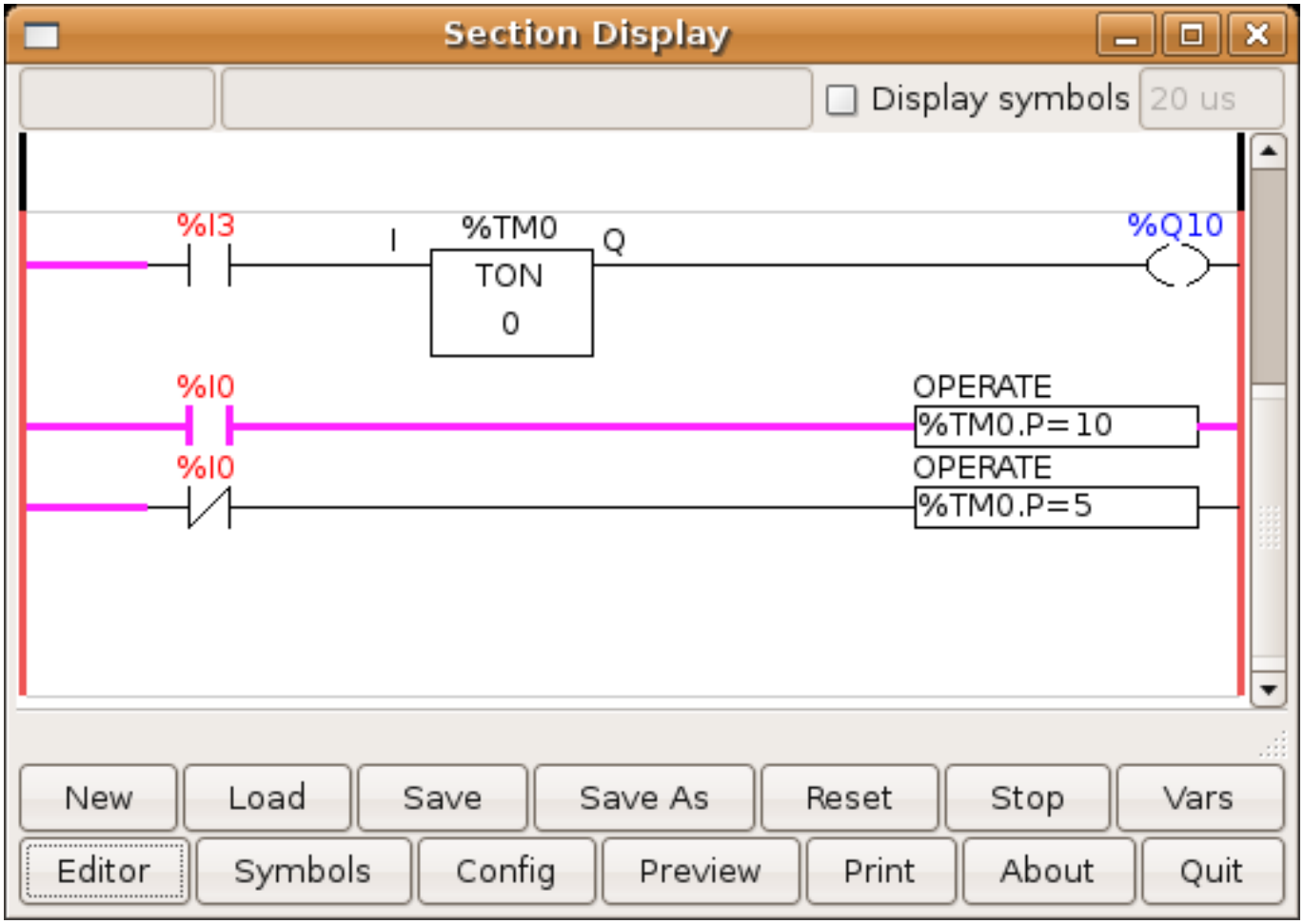


FIGURE 18.22 – Timer/Operate Example

In this case %I0 is true so the timer preset value is 10. If %I0 was false the timer preset would be 5.

18.11.5 Tool Turret

– This Example is not complete yet.

This is a program for one type of tool turret. The turret has a home switch at tool position 1 and another switch to tell you when the turret is in a lockable position. To keep track of the actual tool number one must count how many positions past home you are. We will use Classic Ladder's counter block *$CO*.The counter is preset to 1 when RESET is true. The counter is increased by one on the rising edge of INDEX. We then *COMPARE* the counter value (%C0.V) to the tool number we want (in the example only checks for tool 1 and 2 are shown). We also *OPERATE* the counter value to a word variable (%W0) that (you can assume) is mapped on to a s32 out HAL pin so you can let some other HAL component know what the current tool number is. In the real world another s32 (in) pin would be used to get the requested tool number from LinuxCNC.You would have to load Classic Ladder's real time module specifying that you want s32 in and out pins. See *loading options* above. [display turret sample]

18.11.6 Sequential Example

– This Example is not complete yet.

This is a sequential program.
When the program is first started step one is active.
Then when %B0 is true, steps 2 and 3 are then active and step one is inactive.

Then when %B1 and/or %B2 are true, step 4 and/or 5 are active and step 2 and/or 3 are inactive.
Then when either %B3 OR %B4 are true, step 6 is true and steps 4 and 5 are inactive.
Then when %B5 is true step 1 is active and step 6 is inactive and it all starts again.

As shown, the sequence has been:
%B0 was true making step 2 and 3 active, then %B1 became true
(and still is-see the pink line through %B1)
making step 4 active and step 2 inactive.
Step 3 is active and waiting for %B2 to be true.
Step 4 is active and is waiting for %B3 to be true.
WOW, that was quite a mouthful!!

Troisième partie

Exemples d'utilisation

Chapitre 19

Deuxième port parallèle sur port PCI

Lors de l'ajout d'un deuxième port parallèle placé dans un slot PCI il est indispensable de connaître son adresse avant de pouvoir l'utiliser avec LinuxCNC. Pour trouver l'adresse de ce port, ouvrez un terminal et tapez:

```
lspci -v
```

Vous devriez voir quelques choses comme ci-dessous parmi la liste du matériel installé en PCI:

```
Communication controller: NetMos Technology PCI 1 port parallel adapter (rev 01)
LSI Logic / Symbios Logic: Unknown device 0010
    medium devsel, IRQ 11
    ports at a800 [size=8]
    ports at ac00 [size=8]
    ports at b000 [size=8]
    ports at b400 [size=8]
    ports at b800 [size=8]
    ports at bc00 [size=16]
```

Dans notre cas, l'adresse était la première, nous avons donc modifié le fichier .hal pour passer de

```
loadrt hal_parport cfg=0x378
```

à

```
loadrt hal_parport cfg="0x378 0xa800 in"
```

Noter les guillemets obligatoires encadrant les adresses.

Nous avons également ajouté:

```
addf parport.1.read base-thread
addf parport.1.write base-thread
```

pour que le second port parallèle soit lu et écrit.

Par défaut les 8 premières broches des ports parallèles sont des sorties. Le drapeau *in* situé derrière l'adresse d'un port permet de les positionner comme étant 8 entrées sur ce port.

Chapitre 20

Contrôle de la broche

20.1 Vitesse broche en 0-10V

Si la vitesse de la broche est contrôlée par un variateur de fréquence avec une consigne vitesse en 0 à 10V et qu'une carte de conversion (DAC) comme la m5i20 est utilisée pour sortir le signal.

Premièrement il faut calculer le facteur d'échelle entre la vitesse broche et la tension de commande. Dans cet exemple, la vitesse maximale de la broche sera de 5000 tr/mn pour une tension de commande de 10V. 10 Volts/5000 tr/mn = 0.002 Volts par tr/mn notre facteur d'échelle sera donc de 0.002.

Si la carte DAC ne dispose pas d'une fonction échelle, il est nécessaire d'ajouter un composant *scale* (echelle) au fichier hal pour calibrer *motion.spindle-speed-out* entre 0 et 10 comme demandé par le variateur de fréquence.

```
loadrt scale count=1
addf scale.0 servo-thread
setp scale.0.gain 0.002
net spindle-speed-scale motion.spindle-speed-out => scale.0.in
net spindle-speed-DAC scale.0.out => «le nom de la sortie de votre »DAC
```

20.2 Vitesse de broche en PWM

Si la vitesse de la broche peut être contrôlée par un signal de PWM, utiliser le composant *pwmgen* pour créer le signal:

```
loadrt pwmgen output_type=0
addf pwmgen.update servo-thread
addf pwmgen.make-pulses base-thread
net spindle-speed-cmd motion.spindle-speed-out => pwmgen.0.value
net spindle-on motion.spindle-on => pwmgen.0.enable
net spindle-pwm pwmgen.0.pwm => parport.0.pin-09-out
# Adapter selon la vitesse maximale de la broche en tr/mn
setp pwmgen.0.scale 1800
```

La réponse du contrôleur de PWM est simple: 0% donne 0tr/mn, 10% donnent 180 tr/mn… 100% donnent 1800 tr/mn. Si un minimum est nécessaire pour faire tourner la broche, suivre l'exemple *nist-lathe* fourni dans les exemples de configuration pour ajouter un composant d'échelle.

20.3 Marche broche

Si un signal de marche broche reliant *motion.spindle-on* à une broche de sortie physique est envisagé. Pour relier ces pins à une broche du port parallèle, ajouter une ligne comme la suivante dans le fichier .hal, il faut bien sûr qu'elle soit câblée à l'interface de contrôle.

```
net spindle-enable motion.spindle-on => parport.0.pin-14-out
```

20.4 Sens de rotation de la broche

Pour contrôler le sens de rotation de la broche, les pins de HAL *motion.spindle-forward* et *motion.spindle-reverse* étant contrôlées par M3 et M4, peuvent être mise à une valeur positive différente de zéro pour que M3/4 inverse le sens de la broche.

Pour relier ces pins à des broches du port parallèle utiliser, par exemple, les lignes suivantes dans le fichier .hal, bien sûr ces broches doivent être câblées à l'interface de contrôle.

```
net spindle-fwd motion.spindle-forward -> parport.0.pin-16-out
net spindle-rev motion.spindle-reverse => parport.0.pin-17-out
```

20.5 Démarrage en rampe

Si la broche doit démarrer en rampe et que le contrôleur n'a pas cette possibilité, HAL pourra le faire. Il faut premièrement détourner la sortie de *motion.spindle-speed-out* et la faire transiter par un composant *limit2* dont l'échelle est ajustée de sorte que la consigne suive une rampe entre *motion.spindle-speed-out* et le périphérique recevant la consigne de vitesse. Ensuite, il faut faire connaitre à LinuxCNC le moment ou la broche a atteint sa vitesse pour que les mouvements puissent commencer.

Reprenant dans l'exemple en 0-10 V, la ligne:

```
net spindle-speed-scale motion.spindle-speed-out => scale.0.in
```

sera modifiée, comme indiqué dans l'exemple ci-dessous:

Introduction aux composants de HAL *limit2* et *near*:

Au cas ou vous ne les auriez jamais rencontré auparavant, voici une rapide introduction à ces deux composants de HAL utilisés dans l'exemple suivant.

- Le composant de HAL *limit2* est un composant qui reçoit une valeur sur une entrée et fourni une valeur en sortie, limitée entre les seuils min et max et *également*, limitée pour ne pas dépasser l'amortissement spécifié. En d'autres termes, les fluctuations de la valeur de sortie seront toujours lentes et cette lenteur est ajustable.
- Le composant de HAL *near* est un composant à deux entrées et une sortie binaire qui indique quand les deux entrées sont approximativement égales.

Voir le manuel de HAL ou les man pages, taper juste *man limit2* ou *man near*.

```
# charge un composant temps érel limit2 et un near avec des noms éaiss à suivre
loadrt limit2 names=spindle-ramp
loadrt near names=spindle-at-speed

# ajoute les fonctions au thread
addf spindle-ramp servo-thread
addf spindle-at-speed servo-thread

# fixe le èparamtre max pour l'amortissement
# (ééacclration/éééédclration de la broche en éunits par seconde)
setp spindle-ramp.maxv 60

# édtourne la sortie vitesse broche et l'envoie à l'éentre de la rampe
net spindle-cmd <= motion.spindle-speed-out => spindle-ramp.in

# la sortie de la rampe est éenvoye à l'éentre de lé'chelle
net spindle-ramped <= spindle-ramp.out => scale.0.in
```

```
# pour connaitre quand commencer le mouvement on envoie la vitesse de broche
# écommande à une éentre du composant spindle-at-speed (qui est un composant near).
# on envoie également le signal de fin de rampe (vitesse actuelle)
# sur l'autre éentre de spindle-at-speed
net spindle-cmd => spindle-at-speed.in1
net spindle-ramped => spindle-at-speed.in2

# la sortie de spindle-at-speed est éenvoye à motion.spindle-at-speed
# et quand elle devient TRUE, les mouvements peuvent commencer
net spindle-ready <= spindle-at-speed.out => motion.spindle-at-speed
```

20.6 Vitesse de broche avec signal de retour

Une information de retour est nécessaire pour que LinuxCNC puisse réaliser des mouvements synchronisés avec la broche comme le filetage ou la vitesse de coupe constante. L'assistant de configuration StepConf peut réaliser les connections lui même si les signaux *Canal A codeur broche* et *Index codeur broche* sont choisis parmi les entrées.

Matériel supposé présent:
– Un codeur est monté sur la broche et délivre 100 impulsions par tour sur son canal A.
– Ce canal A est raccordé à la broche 10 du port parallèle.
– L'index de ce codeur est connecté à la broche 11 du port parallèle.
Configuration de base pour ajouter ces composants:

```
loadrt encoder num_chan=1
addf encoder.update-counters base-thread
addf encoder.capture-position servo-thread
setp encoder.0.position-scale 100
setp encoder.0.counter-mode 1
net spindle-position encoder.0.position => motion.spindle-revs
net spindle-velocity encoder.0.velocity => motion.spindle-speed-in
net spindle-index-enable encoder.0.index-enable <=> motion.spindle-index-enable
net spindle-phase-a encoder.0.phase-A
net spindle-phase-b encoder.0.phase-B
net spindle-index encoder.0.phase-Z
net spindle-phase-a <= parport.0.pin-10-in
net spindle-index <= parport.0.pin-11-in
```

20.7 Vitesse broche atteinte

Si le moteur de broche possède un retour d'information de vitesse provenant d'un codeur, il est alors possible d'utiliser la variable *motion.spindle-at-speed* pour permettre à LinuxCNC d'attendre que la broche ait atteint sa vitesse de consigne avant d'effectuer tout mouvement. Cette variable passe à TRUE quand la vitesse commandée est atteinte. Comme le retour vitesse est la vitesse de consigne ne sont jamais *exactement* identiques, il faut utiliser le composant *near* qui indique quand les deux composantes sont suffisamment proches l'une de l'autre.

Il est nécessaire de connecter la commande de vitesse broche sur near.n.in1 et le signal de retour vitesse du codeur sur near.n.in2. La sortie near.n.out est connectée à motion.spindle-at-speed. Le paramètre near.n.scale doit être ajusté pour indiquer dans quelle mesure les deux valeurs sont suffisamment proches pour passer activer la sortie. Selon le matériel utilisé, il pourra être utile d'ajuster l'échelle.

Les éléments suivants sont à ajouter au fichier HAL pour activer *Spindle At Speed*. Si near est déjà présent dans le fichier HAL, augmenter le numéro de composant et adapter le code suivant en conséquence. S'assurer que le nom du signal est bien le même dans le fichier HAL.

```
# charger un composant near et l'attacher à un thread
loadrt near
addf near.0 servo-thread
```

```
# connecter une éentre à la vitesse de broche écommande
net spindle-cmd => near.0.in1

# connecter une éentre à la mesure de vitesse broche du codeur
net spindle-velocity => near.0.in2

# connecter la sortie sur l'éentre spindle-at-speed
net spindle-at-speed motion.spindle-at-speed <= near.0.out

# Ajuster les éentres de vitesse de broche pour être dans une fourchette de 1%
setp near.0.scale 1.01
```

Chapitre 21

Utilisation d'une manivelle

Cet exemple explique comment relier une manivelle, facile à trouver aujourd'hui sur le marché. Cet exemple utilisera la manivelle MPG3 avec une carte d'interface C22 de chez CNC4PC et un second port parallèle placé sur un slot PCI. Cet exemple fournira trois axes avec chacun trois incréments de pas: 0.1, 0.01, 0.001.

Dans le fichier *custom.hal* ou dans un fichier jog.hal, ajouter ce qui suit en vérifiant bien que les composants *mux4* ou *encoder* ne soient pas déjà utilisés. Si c'était le cas il faudrait en augmenter le nombre en incrémentant la valeur du *count* de la commande *loadrt*. Ajuster également le numéro de référence. Les composants mux4 et encoder sont décrits dans le manuel de HAL et dans les man pages.

Manivelle de jog

```
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

# Mode position
# Chaque cran de manivelle provoque un pas écalibr,
# la édure du mouvement total peut édpasser la édure de rotation de la manivelle.
# C'est le mode par édfaut.

setp axis.N.jog-vel-mode 0

# Mode vitesse
# L'axe s'êarrte quand la manivelle s'êarrte, êmme si la pas de jog est incomplet.
# éDcommenter la ligne suivante pour obtenir ce mode de fonctionnement,
# et commenter le mode position.

setp axis.N.jog-vel-mode 1

# Chaque axe est éajust éindpendemment des autres.

# Tailles des pas de jog

setp mux4.0.in0 0.1
setp mux4.0.in1 0.01
setp mux4.0.in2 0.001

# éSlecteur de taille des pas du jog

net scale1 mux4.0.sel0 <= parport.1.pin-09-in
net scale2 mux4.0.sel1 <= parport.1.pin-10-in

net pend-scale axis.0.jog-scale <= mux4.0.out
```

```
net pend-scale axis.1.jog-scale
net pend-scale axis.2.jog-scale

# Signaux du codeur

net mpg-a encoder.0.phase-A <= parport.1.pin-02-in
net mpg-b encoder.0.phase-B <= parport.1.pin-03-in

# éSlecteur d'axe

net mpg-x axis.0.jog-enable <= parport.1.pin-04-in
net mpg-y axis.1.jog-enable <= parport.1.pin-05-in
net mpg-z axis.2.jog-enable <= parport.1.pin-06-in

net pend-counts axis.0.jog-counts <= encoder.0.counts
net pend-counts axis.1.jog-counts
net pend-counts axis.2.jog-counts
```

Chapitre 22

Broche avec variateur GS2

22.1 Exemple

Cet exemple montre les connexions demandées pour utiliser un variateur de fréquence fourni par la société Automation Direct pour piloter une broche. [1] La direction de la broche et sa vitesse seront contrôlées par LinuxCNC.

L'utilisation du composant GS2 est très simple à régler. Une configuration complète peut être réalisée par l'assistant Stepconf. Bien vérifier que les pins *Spindle CW* et *Spindle PWM* sont inutilisées sur la page de réglage du port parallèle.

Placer les lignes suivantes dans le fichier custom.hal pour connecter LinuxCNC au GS2 et avoir le contrôle de celui-ci depuis l'interface utilisateur.

```
# Charger le composant utilisateur pour le variateur variateur
loadusr -Wn spindle-vfd gs2_vfd -n spindle-vfd

# connecter la pin de direction au variateur
net gs2-fwd spindle-vfd.spindle-fwd <= motion.spindle-forward

# connecter la pin de Marche/êArrt au variateur
net gs2-run spindle-vfd.spindle-on <= motion.spindle-on

# connecter la pin indiquant que la consigne vitesse est atteinte
net gs2-at-speed motion.spindle-at-speed <= spindle-vfd.at-speed

# connecter la commande de vitesse au variateur
net gs2-RPM spindle-vfd.speed-command <= motion.spindle-speed-out
```

Sur le variateur de fréquence lui même, il est nécessaire de fixer quelques paramètres avant de pouvoir communiquer avec lui par Modbus. D'autres seront ajustés selon les caractéristiques demandées par le système. Ces réglages sortent, pour la plupart, du cadre de cet exemple, ils sont tous expliqués dans le manuel du variateur qu'il est impératif de consulter.
– Les switches de communication doivent être positionnés sur RS-232C.
– Les paramètres moteur doivent être ajustés en fonction du moteur.
– P3.00 (Source des commandes de marche) doit être ajusté sur *Opérations déterminées par l'interface RS-485*, *03* ou *04*
– P4.00 (Source des commandes de fréquence) doit être ajusté sur *Fréquence déterminée par l'interface RS232C/RS485*, *05*
– P9.02 (Protocole de communication) doit être ajusté sur *Modbus RTU mode, 8 bits de donnée, pas de parity, 2 bits de stop*, *03*
Un panneau de commande virtuel PyVCP, basé sur cet exemple, est donné dans l'exemple avec un variateur.

1. En Europe on trouve une gamme de produits identiques sous la marque Omron.

Quatrième partie

Diagnostics

Chapitre 23

Moteurs pas à pas

Si ce que vous obtenez ne correspond pas à ce que vous espériez, la plupart du temps c'est juste un petit manque d'expérience. Accroître son expérience permet souvent une meilleure compréhension globale. Porter un diagnostic sur plusieurs problèmes est toujours plus facile en les prenant séparément, de même qu'une équation dont on a réduit le nombre de variables est toujours plus rapide à résoudre. Dans le monde réel ce n'est pas toujours le cas mais c'est une bonne voie à suivre.

23.1 Problèmes communs

23.1.1 Le moteur n'avance que d'un pas

La raison la plus fréquente dans une nouvelle installation pour que le moteur ne bouge pas est l'interversion entre le signal de pas et le signal de direction. Si, quand vous pressez le bouton de jog dans un sens puis dans l'autre, le moteur n'avance que d'un pas à chaque fois et toujours dans la même direction, vous êtes dans ce cas.

23.1.2 Le moteur ne bouge pas

Certaine interfaces de pilotage de moteurs ont une broche d'activation (enable) ou demandent un signal de pompe de charge pour activer leurs sorties.

23.1.3 Distance incorrecte

Si vous commandez une distance de déplacement précise sur un axe et que le déplacement réel ne correspond pas, alors l'échelle de l'axe n'est pas bonne.

23.2 Messages d'erreur

23.2.1 Erreur de suivi

Le concept d'erreur de suivi est étrange quand il s'agit de moteurs pas à pas. Etant un système en boucle ouverte, aucune contre réaction ne permet de savoir si le suivi est correct ou non. LinuxCNC calcule si il peut maintenir le suivi demandé par une commande, si ce n'est pas possible il stoppe le mouvement et affiche une erreur de suivi. Les erreurs de suivi sur les systèmes pas à pas sont habituellement les suivantes:
– FERROR to small - (FERROR trop petit)
– MIN_FERROR to small - (MIN_FERROR trop petit)
– MAX_VELOCITY to fast - (MAX_VELOCITY trop rapide)
– MAX_ACCELERATION to fast - (MAX_ACCELERATION trop rapide)

– BASE_PERIOD set to long - (BASE_PERIOD trop longue)
– Backlash ajouté à un axe (rattrapage de jeu)
Toutes ces erreurs se produisent lorsque l'horloge temps réel n'est pas capable de fournir le nombre de pas nécessaire pour maintenir la vitesse requise par le réglage de la variable BASE_PERIOD. Ce qui peut se produire, par exemple après un test de latence trop bref pour obtenir un valeur fiable, dans ce cas, revenir à une valeur plus proche de ce qu'elle était et réessayez. C'est également le cas quand les valeurs de vitesse maximum et d'accélération maximum sont trop élevées.

Si un backlash a été ajouté, il est nécessaire d'augmenter STEPGEN_MAXACCEL aux environs du double de MAX_ACCELERATION dans la section [AXIS] du fichier INI et ce, pour chacun des axes sur lesquels un backlash a été ajouté. LinuxCNC utilise une *extra accélération* au moment de l'inversion de sens pour reprendre le jeu. Sans correction du backlash, l'accélération pour le générateur de pas peut être juste un peu plus basse que celle du planificateur de mouvements.

23.2.2 Erreur de RTAPI

Quand vous rencontrez cette erreur:

```
RTAPI: ERROR: Unexpected realtime delay on task n
```

C'est généralement que la variable BASE_PERIOD dans la section [EMCMOT] du fichier ini a une valeur trop petite. Vous devez lancer un *Latency Test* pendant une durée plus longue pour voir si vous n'avez pas un délai excessif quelque part, responsable de ce problème. Si c'est le cas réajuster alors BASE_PERIOD avec la nouvelle valeur obtenue.

LinuxCNC vérifie le nombre de cycles du CPU entre les invocations du thread temps réel. Si certains éléments de votre matériel provoquent un délai excessif ou que les threads sont ajustés à des valeurs trop rapides, vous rencontrerez cette erreur.

Note
Cette erreur n'est affichée qu'une seule fois par session. En effet, si votre BASE_PERIOD était trop basse vous pourriez avoir des centaines de milliers de messages d'erreur par seconde si plus d'un était affiché.

Plus d'informations sur le test de latence.

23.3 Tester

23.3.1 Tester le timing des pas

Si un de vos axes vibre, grogne ou fait des petits mouvements dans toutes les directions, c'est révélateur d'un mauvais timing d'impulsions de pas de ce moteur. Les paramètres du pilote matériel sont a vérifier et a ajuster. Il peut aussi y avoir des pertes de pas aux changements de direction. Si le moteur cale complétement, il est aussi possible que les paramètres MAX_ACCELERATION ou MAX_VELOCITY aient des valeurs trop élevées.

Le programme suivant vérifie que la configuration de l'axe Z est correcte. Copiez le programme dans le répertoire de votre linuxcnc/nc_files nommez le *TestZ.ngc* ou similaire. Initialisez votre machine avec Z = 0.000 sur le dessus de la table. Chargez et lancez le programme. Il va effectuer 200 mouvements d'aller et retour entre 10.00 et 30.00mm. Si vous avez un problème de configuration, la position de l'axe Z affichée à la fin du programme, soit 10.00mm, ne correspondra pas à la position mesurée. Pour tester un autre axe remplacez simplement le Z des G0 par le nouvel axe.

```
( Faire Z=0 au dessus de la table avant de édmarrer! )
( Ce programme teste les pertes de position en Z )
( msg, test 1 de la configuration de l'axe Z )
G21 #1000=100 ( boucle 100 fois )
( cette boucle comporte un édlai èaprs chaque mouvement )
( test des érglages d'éeacclration et de vitesse )
o100 while [#1000]
   G0 Z30.000
   G4 P0.250
```

```
   G0 Z10.000
   G4 P0.250
   #1000 = [#1000 - 1]
o100 endwhile
( msg, test 2 de la configuration de l'axe Z, pressez S pour continuer)
M1 (un êarrt ici)
#1000=100 ( boucle 100 fois )
( Les boucles suivantes n'ont plus de édlai en fin de mouvements )
( test des érglages des temps de maintien des pilotes)
( et les érglages d'ééacclration max )
o101 while [#1000]
   G0 Z30.000 .
   G0 Z10.000
   #1000 = [#1000 - 1]
o101 endwhile
( msg, Fin Z doit être à 10mm au dessus de la table )
M2
```

Chapitre 24

Glossary

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

LinuxCNC
The Enhanced Machine Controller. Initially a NIST project. LinuxCNC is able to run a wide range of motion devices.

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

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

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

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

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

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

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

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

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

GUI
Graphical User Interface.

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

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

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

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

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

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

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

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

kernel-space
See real-time.

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

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

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

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

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

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

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

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

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

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

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

Real-time
Software that is intended to meet very strict timing deadlines. Under Linux, in order to meet these requirements it is necessary to install RTAI or RTLINUX and build the software to run in those special environments. For this reason real-time software runs in kernel-space.

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

RTLINUX
See http://www.rtlinux.org, one of two real-time extensions for Linux that LinuxCNC can use to achieve real-time performance.

RTAPI
A portable interface to real-time operating systems including RTAI and RTLINUX

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

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

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

Signed Integer
A whole number that can have a positive or negative sign. In HAL it is known as s32. (A signed 32-bit integer has a usable range of -2,147,483,647 to +2,147,483,647.)

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

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

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

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

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

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

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

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

Unsigned Integer
A whole number that has no sign. In HAL it is known as u32. (An unsigned 32-bit integer has a usable range of zero to 4,294,967,296.)

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

Chapitre 25

Legal Section

25.1 Copyright Terms

Copyright (c) 2000-2015 LinuxCNC.org

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

25.2 GNU Free Documentation License

GNU Free Documentation License Version 1.1, March 2000

Copyright © 2000 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

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

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

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

1. APPLICABILITY AND DEFINITIONS

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

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

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary

Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

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

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

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

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

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

2. VERBATIM COPYING

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

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

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

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

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

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

4. MODIFICATIONS

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

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

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

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

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

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

5. COMBINING DOCUMENTS

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

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

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

6. COLLECTIONS OF DOCUMENTS

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

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

7. AGGREGATION WITH INDEPENDENT WORKS

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

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

8. TRANSLATION

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

9. TERMINATION

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

10. FUTURE REVISIONS OF THIS LICENSE

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

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

ADDENDUM: How to use this License for your documents

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

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

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

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

Chapitre 26

Index

_
.linuxcncrc, 10

A
A/U, 42
acme screw, 174
ANGULAR UNITS, 18
axis, 14, 174
axis (hal pins), 37

B
backlash, 174
backlash compensation, 174
ball nut, 174
ball screw, 174
BASE PERIOD, 17
brochage, 39

C
cinematique, 114
Cinematique triviale), 114
CNC, 174
codeur, 22
commentaires, 12
comp, 174
coordinate measuring machine, 175

D
Démarrage en rampe, 164
display units, 175
Documentation des widgets, 48
DRO, 175

E
EDM, 175
encoder, 175

F
feed, 175
feed rate, 175
feedback, 175
feedrate override, 175
FERROR, 20
Fichier ini, 12
Frequence de pas maximale, 39

G
G-Code, 175
GUI, 174, 175

H
HAL, 10, 176
HAL), 40
HOME, 27
home, 176
HOME IGNORE LIMITS, 27
HOME IS SHARED, 27
HOME OFFSET, 27
HOME SEARCH VEL, 20
HOME SEQUENCE, 27
HOME USE INDEX, 27

I
INI, 10, 176
INPUT SCALE, 22
Instance, 176
iocontrol (HAL pins), 38

J
jog, 176
joint coordinates, 176

K
keystick, 14
kinematics, 176

L
lead screw, 176
LINEAR UNITS, 18
LinuxCNC, 175
LinuxCNCIO, 175
LinuxCNCMOT, 175
loop, 177

M
machine units, 176
Machines Cartesiennes, 114
Machines CNC, 114

Marche broche, 163
marche machine, 43
MAX ACCELERATION, 18
MAX LIMIT, 20
MAX VELOCITY, 18
MDI, 176
MIN FERROR, 20
MIN LIMIT, 20
mini, 14
Motion, 34
motion (hal pins), 35

N
NIST, 176
NML, 10

O
offsets, 176
ORIENT OFFSET, 16

P
PARAMETER FILE, 16
part Program, 176
program units, 176
PyVCP avec Axis, 46

R
rapid, 177
rapid rate, 177
real-time, 177
RS274NGC, 177
RS274NGC STARTUP CODE, 17
RTAI, 177
RTAPI, 177
RTLINUX, 177

S
Section [DISPLAY] du fichier ini, 14
Section [EMC] du fichier ini, 14
Section [EMCIO] du fichier ini, 22
Section [EMCMOT] du fichier ini, 17
Section [FILTER] du fichier ini, 15
Section [HAL] du fichier ini, 17
Section [HALUI] du fichier ini, 18
Section [RS274NGC] du fichier ini, 16
Section [TASK] du fichier ini, 17
Section [TRAJ] du fichier ini, 18
Sections, 12
Sections [AXIS_n] du fichier ini, 19
Sens de rotation de la broche, 164
servo motor, 177
SERVO PERIOD, 17
signal enable, 42
signal polarite, 42
Signed Integer, 177
spindle, 177
standard pinout, 40
stepper, 39
stepper motor, 177
SUBROUTINE PATH, 17

T
TASK, 177
TBL, 10
Tk, 177
tklinuxcnc, 14
touchy, 14
TRAJ PERIOD, 17
Traverse Move, 177

U
UNITS, 19
units, 178
Unsigned Integer, 178
USER M PATH, 17

V
VAR, 10
Verrouillage Indexeur, 28
Vitesse broche atteilte, 165
Vitesse broche en 0-10V, 163
Vitesse broche PWM, 42
Vitesse de broche en PWM, 163
Vitesse de détection du contact d'origine, 26
Vitesse de recherche du contact d'origine, 26
VOLATILE HOME, 27

W
world coordinates, 178

X
xlinuxcnc, 14