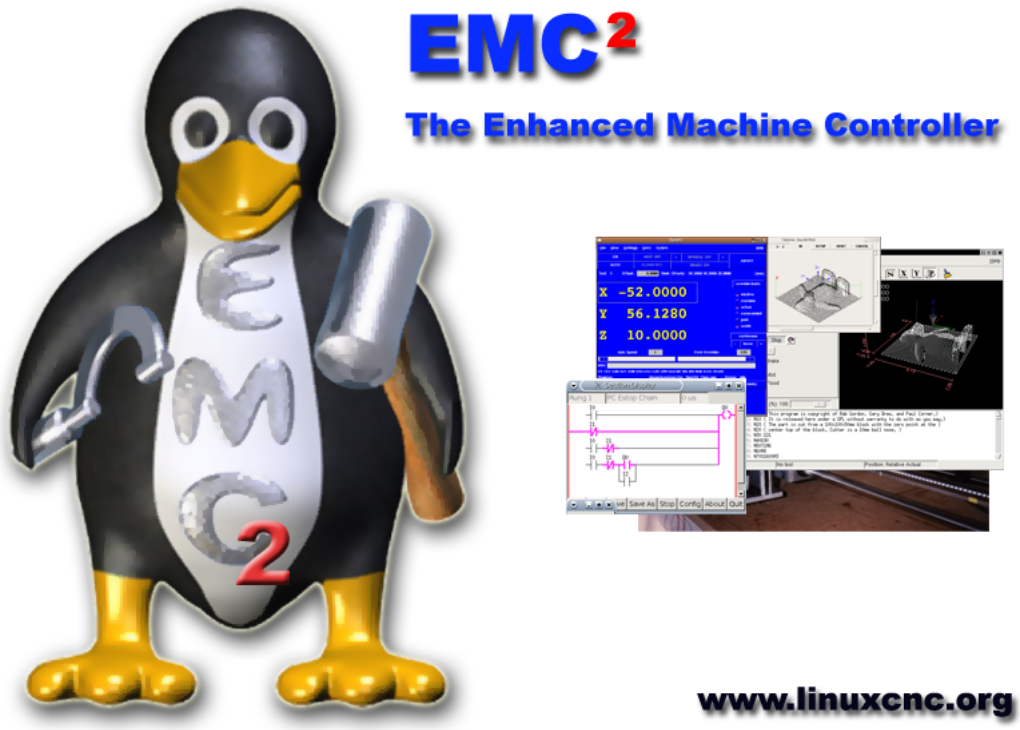
Manuel de l'intégrateur V2.3

The EMC Team

15 octobre 2011



NOTICE :

As of 2011-01-16, the French version of the EMC2 documentation is 2 years out of date due to not having a translator available.

It's recommended to use the English documentation whenever possible.

If you wish to provide updated French translation of EMC2, please contact us.

AVIS :

Au 2011-01-16, la version française de la documentation EMC2 est de 2 ans à la date d'échéance pour ne pas avoir un traducteur disponible.

Il est recommandé d'utiliser la documentation en anglais chaque fois que possible.

Si vous souhaitez fournir des mises à jour traduction française d'EMC2, s'il vous plaît contactez-nous.

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

Table des matières

Première partie

Installation d'EMC2

Chapitre 1

Installer le logiciel EMC2

1.1 Introduction

Un des problèmes les plus souvent évoqués par les utilisateurs à propos d'EMC2 a été qu'il ne s'installait pas de lui-même. Il fallait qu'ils récupèrent les sources, les compilent eux-mêmes, essaient d'appliquer un noyau Linux patché RT, etc. Les développeurs d'EMC2 ont donc choisi une distribution standard appelée Ubuntu[1].

Ubuntu a été choisi parce-qu'il est parfaitement dans l'esprit Open Source d'EMC2 :

- Ubuntu restera toujours gratuit, il n'y a pas de frais pour l'édition "enterprise edition", nous faisons de notre mieux pour rendre notre travail disponible à tous dans les mêmes termes de gratuité.
- Ubuntu fournit un support professionnel commercial à des centaines de sociétés dans le monde, vous aurez peut être besoin de ces services. Chaque nouvelle version d'Ubuntu reçoit des mises à jour de sécurité gratuites pendant 18 mois après sa publication, certaines versions sont supportées plus longtemps.
- Ubuntu utilise le meilleur en termes de traduction et d'accessibilité à ses infrastructures parmi ce que la communauté du logiciel libre peut offrir et pour faire qu'Ubuntu soit aprécié par autant d'utilisateurs que possible.
- Ubuntu est publié régulièrement selon un calendrier précis ; Une nouvelle version est publiée tous les six mois. Vous pouvez utiliser la dernière version stable ou aider à stabiliser la version en cours de développement.
- La communauté Ubuntu est entièrement dévouée aux principes de développement du logiciel libre ; elle encourage tout le monde à utiliser des logiciels libres et open source, les améliorer et les distribuer.

1.2 La page de téléchargement d'EMC

Vous pouvez trouver l'annonce des plus récentes versions publiées d'EMC2 sur le site www.linuxcnc.org. Les versions d'EMC2 sont fournies de deux manières (sources et paquets binaires). Les sources (décrites dans le manuel du développeur) consistent en un tarball (emc2-<version>.tar.gz), que vous devez charger et décompacter dans votre répertoire home.

Le présent document (plus orienté utilisateur final) expliquera seulement comment installer les paquets binaires sur une distribution Ubuntu[2].

[1] Le mot "Ubuntu" est un ancien mot Africain, signifiant "humanité aux autres". Ubuntu signifie aussi "Je suis ce que je suis à cause de ce que nous sommes tous". La distribution Ubuntu Linux amène l'esprit d'Ubuntu au monde du logiciel. Vous pouvez en lire plus à ce propos ici : http://www.ubuntu-fr.org/

[2] Pour plus d'informations sur les autres variantes Linux, lisez le Manuel du développeur ou demandez de l'aide sur la Liste de diffusion http://sourceforge.net/mail/?group_id=6744.

1.3 Le live CD d'EMC2

Les développeurs d'EMC2 ont créé un Live-CD basé sur Ubuntu 6.06 qui vous permet d'essayer EMC2 avant de l'installer, c'est également une manière facile d'installer ensemble Ubuntu et EMC2.

Téléchargez l'image image ISO http://linuxcnc.org/iso/emc2-ubuntu6.06-desktop-i386.iso (Miroir EU http://dsplabs.utt.ro/~juve/emc/) et gravez la sur un CD. (la somme MD5 du CD est vérifiable)

Quand vous bootez avec ce CD dans le lecteur de votre machine, vous pouvez voir et expérimenter un environnement identique à celui d'EMC2 qui sera le vôtre si vous choisissez de l'installer.

Si cette démonstation vous a convaicu, cliquez sur l'icône Install du bureau, répondez à quelques questions (votre nom, fuseau horaire, mot de passe) et l'installation terminera en quelques minutes.

Cette installation vous apportera tout les avantages du support de la communauté Ubuntu avec la configuration automatique d'EMC2. Quand une mise à jour d'EMC2 sera publiée, le gestionnaire de paquets vous le fera savoir et vous permettra une mise à jour aisée.

1.4 Script d'installation d'EMC2

Il est également possible d'utiliser un simple script d'installation d'emc2 sur Ubuntu pour les utilisateurs ayant déjà une installation existante d'Ubuntu. Il lance la commande expliquée dans 1.5.

Pour l'utiliser vous devez :

- Charger le script depuis Script pour Dapper http://linuxcnc.org/dapper/emc2-install.sh (Pour Ubuntu 6.06) ou Script pour Hardy Heron http://linuxcnc.org/hardy/emc2-install.sh (Pour Ubuntu 8.04)
- Le sauver sur votre bureau. Faire un clic droit sur son icône, sélectionner Propriétés. Choisir l'onglet Permissions et cocher Propriétaire : Exécuter. Fermer la fenêtre des propriétés.
- Maintenant double-cliquez sur l'icône emc2-install.sh, et choisissez "Run in Terminal". Une console va apparaître et vous demander votre mot de passe.
- Quand l'installation vous demande si vous voulez installer les paquets d'EMC2, pressez Entrée pour accepter. Laissez ensuite l'installation se poursuivre jusqu'à la fin.
- Quand elle est terminée, éjectez le CD puis vous devrez redémarrer votre machine (Système > Quitter > Redémarrer l'ordinateur). Quand vous aurez redémarré vous pourrez alors lancer EMC2 via le menu Applications > CNC.
- Si vous n'êtes pas prêt pour configurer votre machine, essayez la configuration sim-AXIS ; elle démarre en mode "machine simulée" qui ne requiert le raccordement d'aucun matériel.
- Maintenant que l'installation est terminée, Ubuntu vous avertira quand des mises à jour d'EMC2 seront disponibles. Quand ça arrivera, vous pourrez mettre à jour facilement et automatiquement avec le gestionnaire de mises à jour.

1.5 Installation manuelle par apt-get.

Cette petite section décrira comment installer EMC2 sur Ubuntu 6.06 “Dapper Dreake” en utilisant les commandes apt dans une console. Si vous connaissez un peu Linux et Debian, ça va être facile. Sinon, vous devriez peut être lire 1.4.

Premièrement, ajoutez le dépôt à /etc/apt/sources.list :

```
$ sudo sh -c 'echo "deb http ://www.linuxcnc.org/emc2/ dapper emc2.2" >>/etc/apt/sources.list;'
$ sudo sh -c 'echo "deb-src http ://www.linuxcnc.org/emc2/ dapper emc2.2" >>/etc/apt/sources.list'
```

Puis faites les mises à jour et l'installation d'emc2 avec :

```
$ sudo apt-get update
$ sudo apt-get install emc2
```

Ces commandes vont installer correctement les paquets emc2 avec toutes leurs dépendances[3].

Vous pourriez avoir une alarme indiquant que les paquets proviennent d'une source non vérifiée (ce qui voudrait dire que votre ordinateur ne reconnaît pas la signature GPG des paquets). Pour corriger cette situation, appliquez les commandes suivantes :

```
$ gpg --keyserver pgpkeys.mit.edu --recv-key BC92B87F
$ gpg -a --export BC92B87F | sudo apt-key add -
```

[3]Les dépendances sont un des atouts majeurs des distributions basées sur Debian. Elles assurent que vous avez la totalité de ce qui doit être installé. Même dans un cas comme emc2 qui nécessite un noyau de Linux patché pour travailler en temps réel, ainsi que toutes les librairies indispensables.

Chapitre 2

Compiler EMC2 depuis les sources

2.1 Introduction

Quelques difficultés sont à surmonter quand vous commencez à installer EMC2, son téléchargement et l'installation du software proprement dit. L'ensemble des fichiers d'EMC2 sont placés dans le dépôt git.linuxcnc.org. EMC2 est également disponible en paquets pré-compilés (pour différentes plateformes) pour téléchargement depuis ce site.

L'installation peut être une tâche compliquée pour quelqu'un de nouveau sous Linux. La partie la plus dure étant d'appliquer le patch temps réel (Real Time Linux) au noyau. Après ça, installer EMC2 est assez facile. Celà dit, il est dorénavant possible aux utilisateurs de profiter d'une possibilité totalement nouvelle, il leur suffit d'installer Ubuntu (une distribution Linux vraiment conviviale), puis d'exécuter un simple script d'installation, et ils auront alors un EMC2 directement en état de marche sur un noyau temps réel. Les informations pour accéder à cette solution sont diponibles sur www.linuxcnc.org à la page Download.

2.2 Page de téléchargement EMC

Vous pouvez trouver l'annonce des versions les plus récentes d'EMC2 sur www.linuxcnc.org. Les versions d'EMC2 sont fournies de deux manières, sources et paquets binaires. Les sources (described furtheron) sont sous forme de fichiers tarball (emc2-version.tar.gz), que vous devez télécharger et décompacter dans votre répertoire home.

2.3 Gestion des versions d'EMC2

EMC2 utilise un modèle de versions similaire (bien que simplifié) à celui utilisé par Debian. Il y a tout le temps trois versions d'EMC2. Debian utilise "stable", "testing" et "unstable". Nous utilisons "Released", "Testing" et "Head". Pour les dernières informations, cliquez sur la version qui vous intéresse.

Released est exactement ça, une version publiée d'EMC2 avec un numéro de version. Elle a été testée par beaucoup de développeurs et de béta testeurs avant d'être publiée, elle est utilisable par la moyenne des utilisateurs. Les développeurs et réguliers des IRC/mailing list sont prêts à aider ceux qui démarrent avec une version "released". **"Released"** est disponible sous plusieurs formes, incluant .debs pour Ubuntu et tarballs de sources pour une compilation locale. Il y a un dépôt Debian qui a toujours la dernière version "released" (elle permet donc de faciliter les mises à jour d'une version stable).

Testing est une version d'EMC2 qui est prête pour le "beta testing" mais pas pour une publication générale. Avant qu'une version soit labellisée **testing** elle doit d'abord être compilée et doit démarrer sur différentes plateformes, mais il y aura probablement des limitations et divers problèmes. La page **Testing** du wiki est prévue pour lister les problèmes connus et leurs solutions, mais il reste probablement aussi des bugs non découverts. Puisque la version **Testing** est un software "beta", il ne doit pas être utilisé pour tout ce qui est critique. Les utilisateurs de la version **Testing** doivent comprendre qu'il s'agit d'un software en beta et qu'ils doivent être disposés à donner des rapports de bugs détaillés si quelque chose ne va pas. **Testing** est disponible principalement comme une balise en git, toutefois pour la commodité des testeurs, un dépôt "testing" debian et/ou des tarballs peuvent aussi être disponibles. C'est le conseil d'administration d'EMC qui décide quand une version "Testing" est digne de devenir "Released". C'est une décision formelle, présentée par voix de motion aux votes du conseil d'administration ou votes par la mailing liste de l'IRC.

master est un terme pour indiquer l'emplacement des versions en début de développement. Une version **TRUNK** peut souvent être non fonctionnelle. Lorsque la version **TRUNK** sera réputée digne par de nombreux testeurs soit un grand nombre de personnes, la balise **"Testing"** lui sera appliquée. C'est une décision informelle, prise par consensus à la tête des développeurs, habituellement sur l'IRC. Le développement continue immédiatement et un autre **TRUNK** diverge de cette nouvelle version **Testing**. **TRUNK** n'a pas de numéro de version, au cours d'un veek-end chargé il peut changer littéralement toutes les 10 minutes.

2.4 Téléchargement et compilation des sources.

Les quelques sections suivantes décriront comment se procurer les sources d'EMC2 et les compiler.

Pour les télécharger, allez simplement sur www.linuxcnc.org à la page “Download” et prenez les tarballs de la dernière version “release” ou “testing”.

Quand vous les avez dans votre répertoire home, il faut les extraire, ouvrez une console et faites :

```
$ cd ~/
$ tar xzvf emc2-version.tar.gz
```

Puis vous devez décider quel type d'installation vous voulez. Il y a deux possibilités pour essayer EMC2 :

Installed Comme la plupart des autres logiciels sous Linux, les fichiers sont placés dans des répertoires système, ils sont automatiquement disponibles à tous les utilisateurs de l'ordinateur.[1]

Run-in-place Tous les dossiers sont conservés à l'intérieur du répertoire EMC2. Cette option est utile pour essayer EMC2, surtout quand il existe déjà une autre version d'EMC2 installée sur le système..

2.4.1 Télécharger une version git

Si vous souhaitez utiliser la version 'git' d'EMC2, veuillez suivre les instructions de notre wiki pour obtenir le code source : : http://wiki.linuxcnc.org/cgi-bin/emcinfo.pl?Git

2.5 Installed

EMC2 suit la manière standard de la compilation de logiciel sous linux. Pour compiler il suffit de se rendre dans le répertoire des sources :

[1] Le paquet pré-installé pour Ubuntu Linux utilise la méthode “installé”

```
$ cd ~/emc2/src
```

et d'y lancer ces commandes :

```
$ ./configure
$ make && sudo make install
```

Pour le lancer, tapez 'emc'.

2.6 Run-in-place

Si vous voulez seulement tester le logiciel avant de l'installer, ou si vous avez peur d'écraser une version déjà existante, vous pouvez essayer le mode Run-In-Place (RIP). Dans ce mode, il n'y a aucune installation et aucun fichier ne sera placé en dehors du répertoire `~/emc2`.

Faites juste :

```
$ cd ~/emc2/src
```

puis tapez ces commandes :

```
$ ./configure --enable-run-in-place
$ make && sudo make setuid
```

Dans une console, où vous voulez utiliser EMC2, tapez :[2]

```
$ . ~/emc2/scripts/emc-environment
```

Jusqu'à ce que vous fermiez la console, il sera mis en place afin que les programmes et les pages de manuel soient disponibles sans avoir à se référer au chemin à chaque fois. Ensuite vous pouvez lancer EMC2 en faisant :

```
$ emc
```

2.7 Simulateur

Pour installer EMC2 sur un système sans noyau temps réel, ajoutez `--enable-simulator` à la ligne de commande `configure`. Dans ce mode, seule la partie purement programme d'EMC2 démarrera. Aucun matériel n'aura à être contrôlé, les timmings ne sont pas garantis, mais les autres fonctionnalités de HAL, EMC2 et ses diverses interfaces sont disponibles. Pour utiliser ce mode ajoutez `--enable-run-in-place` à la commande configure, l'étape du `sudo make setuid` n'est pas nécessaire.

2.8 Editer et recompiler

Vous pouvez avoir besoin de recompiler le code d'EMC2 pour diverses raisons. Vous pouvez avoir à modifier le code source, ou vous pouvez avoir seulement téléchargé quelques nouveaux fichiers. Pour recompiler, tapez les commandes suivantes :

[2] En tapant cette commande dans dans le script de démarrage de la console, comme `~/.bash_profile`, vous n'aurez plus à la taper manuellement dans la fenêtre de chaque console.

```
$ cd ~/emc2/src
$ make && sudo make install # pour le run-installed
$ make && sudo make setuid  # pour le run-in-place
$ make                      # pour le run-in-place en simulateur
```

Le processus de compilation est suffisamment performant pour ne recompiler que ce qui est affecté par vos changements.

Deuxième partie

Configuration d'EMC2

Chapitre 3

Test de latence

Ce test est le premier test à effectuer sur un PC pour savoir si celui-ci est capable de piloter une machine CNC.

Le temps de latence est le temps nécessaire au PC pour arrêter ce qu'il est en train de faire pour répondre à une requête externe. Dans notre cas, la requête est l'horloge qui sert de référence pour les impulsions de pas. Plus la latence est basse, plus l'horloge pourra être rapide et donc, plus rapides et plus douces seront les impulsions de pas.

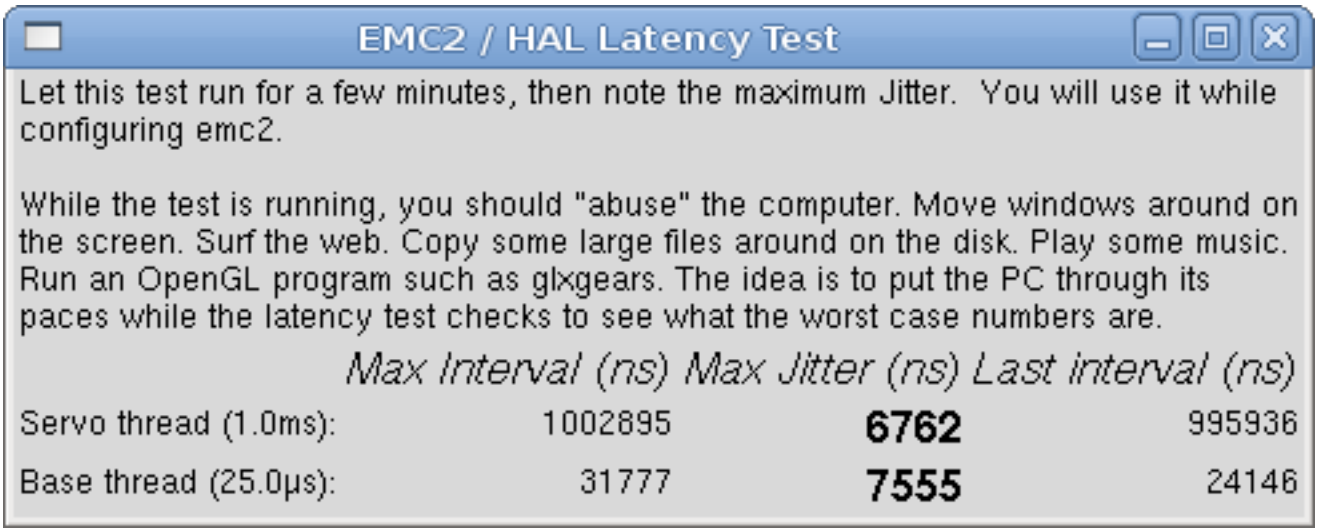
Le temps de latence est beaucoup plus important que la vitesse du µP. Un vieux Pentium III qui répond aux interruptions avec 10 microsecondes entre chacune peut donner de meilleurs résultats qu'un rapide P4 en Hyperthreading.

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.

Faire générer les impulsions de pas au logiciel présente un grand 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 :

– Leur fréquence maximum est limitée
– Les trains d'impulsions générés sont irréguliers
– Ils chargent le processeur

La seule façon de savoir ce qu'il en est sur votre PC est de lancer un test de latence de HAL. Pour exécuter ce test, il suffit d'ouvrir une console et de taper : **`latency-test`** . Vous devriez voir quelque chose comme ceci :



Pendant que le test est en cours d'exécution, il faut « abuser » de l'ordinateur. Déplacez les fenêtres sur l'écran. Connectez vous à l'Internet. Copiez quelques gros fichiers sur le disque dur. Jouer de la musique. Lancez une démo OpenGL telle que **glxgears**. L'idée est de charger le PC au maximum pour que le temps de latence soit mesuré dans le pire des cas. **Ne pas exécuter EMC2 ou Stepconf pendant que latency-test est en cours d'exécution.**

Le chiffre **max jitter** dans cet exemple est de 17894 nanosecondes, soit 17.9 microsecondes. Enregistrer ce chiffre et entrez le dans Stepconf quand il le demande.

Dans cet exemple de test de latence il n'a fallu que quelques secondes pour afficher cette valeur. Vous devrez peut être lancer le test 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 (voir Fixing Dapper SMI Issues http://wiki.linuxcnc.org/cgi-bin/emcinfo.pl?FixingDapperSMIIssues)

Alors, que signifient les résultats ? Si le résultat de votre Max Jitter est en dessous d'environ 15-20 microsecondes (15000-20000 nanosecondes), l'ordinateur pourra donner d'excellents résultats avec la génération logicielle des pas. Si le temps de latence est à plus de 30-50 microsecondes, vous aurez de bons résultats, mais la vitesse maximum sera un peu décevante, spécialement si vous utilisez des micropas ou si le pas de votre vis est fin. Si les résultats sont de 100uS 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 EMC, en utilisant des micropas ou pas.

Notez que si vous obtenez une latence élevée, 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 Matrox d'occasion à $5US a résolu le problème. EMC n'exige pas de matériel de pointe.

3.1 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. EMC2 supporte un maximum de 8 ports parallèles.

Certaines parmis les bonnes cartes parallèles sont à base de chipset Netmos. Elles fournissent un signal +5V bien propre, elles sont fournies avec 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 fournie, 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 (racccordé par un ruban) 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 vous modifiez l'installation des cartes PCI de votre machine, n'oubliez pas de vérifier leurs adresses avant de lancer EMC.

Chapitre 4

Configuration, fichier ini

4.1 Fichiers utilisés pour la configuration

EMC 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 d'EMC. Ces fichiers sont lus à chaque fois que le logiciel démarre. Certains d'entre eux sont lus de nombreuses fois pendant l'exécution d'CNC.

Les fichiers de configuration inclus :

INI Le fichier ini écrase les valeurs par défaut compilées dans le code d'EMC. Il contient également des sections qui sont lues directement par le HAL (Hardware Abstraction Layer, couche d'abstraction matérielle).

HAL Les fichiers hal installent les modules de process, ils créent les liens entre les signaux d'EMC 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 d'une exécution à l'autre.

TBL Ce fichier contient les informations relatives aux outils.

NML Ce fichier configure les canaux de communication utilisés par EMC. Il est normalement réglé pour lancer toutes les communications avec un seul ordinateur, peut être modifié pour communiquer entre plusieurs ordinateurs.

.emcrc 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 d'EMC.[2]

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 EMC et doivent toujours avoir la section et le nom donné à l'item.

4.2 Organisation du fichier INI

Un fichier INI typique suit une organisation simple ;

[1] Ne confondez pas 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/user/)

– commentaires.
– sections,
– variables.

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

4.2.1 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 ignorée. 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.
; Je l'ai ajusté le 12 janvier 2006
```

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

```
# DISPLAY = tkemc
DISPLAY = axis
# DISPLAY = mini
# DISPLAY = keystick
```

Dans cette liste, la variable DISPLAY est positionnée sur axis puisque toutes les autres sont commentées. Si quelqu'un édite une liste comme celle-ci et par erreur, décommente deux lignes, c'est la première rencontrée qui sera utilisée.

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

```
INCORRECT = value     # and a comment
```

4.2.2 Sections

Les différentes parties d'un fichier .ini sont regroupées dans des 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 emc :

– [EMC] informations générales (4.3.1)
– [DISPLAY] sélection du type d'interface graphique (4.3.2)
– [RS274NGC] ajustements utilisés par l'interpréteur de g-code
– [EMCMOT] Réglages utilisés par le contrôleur de mouvements temps réel (4.3.3)
– [HAL] spécifications des fichiers .hal (4.3.5)
– [TASK] Réglages utilisés par le contrôleur de tâche (4.3.4)
– [TRAJ] Réglages additionnels utilisés par le contrôleur de mouvements temps réel (4.3.6)
– [AXIS_0] ... [AXIS_n] Groupes de variables pour AXIS (4.3.7)
– [EMCIO] Réglages utilisés par le contrôleur d'entrées/sorties (4.3.8)
– [HALUI] Commandes MDI utilisées par HALUI. Voir le chapitre sur HALUI pour plus d'informations (13.2.12)

4.2.3 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 = jusqu'à 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 EMC. 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 celles qui sont utilisées dans les exemples de fichiers de configuration.

4.3 Définition des variables du fichier INI

4.3.1 Section [EMC]

VERSION = $Revision : 1.3 $ 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.

RS274NGC_STARTUP_CODE = G21 G90 Une chaine de codes NC qui sera utilisée pour initialiser l'interpréteur. Elle ne se substitue pas à la spécification des gcodes modaux du début de chaque fichier ngc. Les codes modaux des machines diffèrent, ils pourraient être modifiés par les gcodes interprétés plus tôt dans la session.

4.3.2 Section [DISPLAY]

Les différentes interfaces du programme utilisent différentes options. Toutes les options ne sont pas supportées par toutes les interfaces.

DISPLAY = tkemc Le nom de l'interface utilisateur à utiliser. Les options disponibles sont les suivantes :

- axis
- keystick
- mini
- tkemc
- xemc

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 emc. 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_LINEAR_VELOCITY = .25 Vitesse minimum par défaut pour les jogs linéaires, en unités machine par seconde. Seulement utilisé dans l'interface AXIS.

MAX_LINEAR_VELOCITY = 1.0 Vitesse maximum par défaut pour les jogs linéaires, en unités machine par seconde. Seulement utilisé dans l'interface AXIS.

DEFAULT_ANGULAR_VELOCITY = .25 Vitesse minimum par défaut pour les jogs angulaires, en unités machine par seconde. Seulement utilisé dans l'interface AXIS.

MAX_ANGULAR_VELOCITY = 1.0 Vitesse maximum par défaut pour les jogs angulaires, en unités machine par seconde. Seulement utilisé dans l'interface AXIS.

PROGRAM_PREFIX = ~/emc2/nc_files Répertoire par défaut des fichiers de g-codes et emplacement des M-codes définis par l'utilisateur.

INCREMENTS = 1 mm, .5 mm, . . . Définit les incréments disponibles pour le jog incremental. Voir la section 4.3.2 pour plus d'informations. Seulement utilisé dans l'interface AXIS.

INTRO_GRAPHIC = emc2.gif L'image affichée sur l'écran d'accueil.

INTRO_TIME = 5 Durée d'affichage de l'écran d'accueil.

OPEN_FILE = /full/path/to/file.ngc Le fichier NC à utiliser au démarrage d'AXIS.

4.3.3 Section [EMCMOT]

BASE_PERIOD = 50000 (HAL) "Période de base" des tâches, exprimée en nanosecondes. C'est la plus rapide des horloges de la machine.
Avec un système à servomoteurs, il n'y a généralement pas de raison pour que **BASE_PERIOD** soit plus petite que **SERVO_PERIOD**.
Sur une machine de type "step&direction" avec génération logicielle des impulsions de pas, c'est **BASE_PERIOD** qui détermine le nombre maximum de pas par seconde. Si de longues impulsions de pas ou de longs espaces entre les impulsions ne sont pas requis par l'électronique, la fréquence maximum absolue est de un pas par **BASE_PERIOD**. Ainsi, la **BASE_PERIOD** utilisée ici donnera une fréquence de pas maximum absolue de 20000 pas par seconde. 50000ns est une valeur assez large. La plus petite valeur utilisable est liée au résultat du test de latence (**??**), à la longueur des impulsions de pas nécessaire et à la vitesse du µP.
Choisir une BASE_PERIOD trop basse peut amener à des messages "Unexpected realtime delay", des bloquages ou des reboots spontanés.

SERVO_PERIOD = 1000000 (HAL) Période de la tâche "Servo", exprimée également en nanosecondes. Cette valeur sera arrondie à un multiple entier de **BASE_PERIOD**. Elle est utilisée aussi sur des systèmes basés sur des moteurs pas à pas
C'est la vitesse avec laquelle la nouvelle position des moteurs est traitée, les erreurs de suivi vérifiées, les valeurs des sorties PID sont rafraichies etc.
Sur la plupart des systèmes **cette** valeur n'est pas à modifier. Il s'agit du taux de mise à jour du planificateur de mouvement de bas niveau.

TRAJ_PERIOD = 1000000 (HAL) Période du planificateur de trajectoire, exprimée en nanosecondes. Cette valeur sera arrondie à un multiple entier de **SERVO_PERIOD**.
Excepté pour les machines avec une cinématique particulière (ex : hexapodes) Il n'y a aucune raison de rendre cette valeur supérieure à **SERVO_PERIOD**.

4.3.4 Section [TASK]

CYCLE_TIME = 0.001 Période exprimée en secondes, à laquelle EMCTASK 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.

4.3.5 Section [HAL]

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 emc s'arrête. Selon les pilotes de matériel utilisés, il est ainsi possible de positionner les sorties sur des valeurs définies quand emc 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 logiciels de protection des défauts de fonctionnement.

POSTGUI_HALFILE = example2.hal *(Seulement avec l'interface AXIS)* Exécute 'example2.hal' après que l'interface graphique ait créé ses HAL pins.

4.3.6 Section [TRAJ]

La section [TRAJ] contient les paramètres généraux du module planificateur de trajectoires d' EMC-MOT. Vous n'aurez pas à modifier ces valeurs si vous utilisez EMC 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 4, U est toujours l'articulation 7 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.

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 qu'EMC démarrera.

NO_FORCE_HOMING = 1 EMC 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'utilisateur averti de s'affranchir de cette restriction de sécurité lors de la phase de mise au point de la machine.

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

4.3.7 Section [AXIS_<num>]

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

TYPE = LINEAR Type des axes, soit LINEAR, soit ANGULAR.

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 utilisées, 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. Les valeurs internes sont des triplets représentant les positions suivantes :

1. Positions nominales
2. Positions en marche positive
3. Positions en marche négative.

La position nominale est celle où devrait être le mobile. La position en marche positive signifie, où se trouve le mobile pendant le déplacement dans le sens positif. La position en marche négative signifie, où se trouve le mobile pendant le déplacement dans le sens négatif. Un triplet par ligne. Actuellement la limite d'EMC2 est de 256 triplets par axe. Si COMP_FILE est spécifié, BACKLASH est ignoré. Les valeurs sont en unités machine.

COMP_FILE_TYPE = 1 En spécifiant une valeur non nulle, le format des triplets du fichier COMP_FILE sera différent. Pour COMP_FILE_TYPE = 0, les valeurs des triplets seront : position nominale, position en marche positive, position en marche négative. Pour COMP_FILE_TYPE différent de 0, les valeurs dans COMP_FILE seront : position nominale, écart sens positif, écart sens négatif. Comparées au valeurs définies au dessus elles correspondent à, nominale, nominale-position en marche positive, nominal-position en marche négative.

Exemple de triplet avec COMP_FILE_TYPE = 0 : 1.00 1.01 0.99.

Le même exemple de triplet avec COMP_FILE_TYPE = 1 : 1.00 -0.01 0.01

MIN_LIMIT = -1000 Limite minimum des mouvements de cet axe (limite soft), en unités machine. Quand cette limite tend à être dépassée, le contrôleur arrête le mouvement.

MAX_LIMIT = 1000 Limite maximum des mouvements de cet axe (limite soft), 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'erreurs 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 intéractivement 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 TkEMC depuis le menu « Réglages » puis « Calibration ».

4.3.7.1 Variables relatives aux prises d'origines

Les paramètres suivants sont relatifs aux prises d'origine, pour plus d'informations, lire le chapitre sur la POM.5

HOME_OFFSET = 0.0 Position du contact d'origine machine de l'axe ou impulsion d'index, en unités machine.

HOME_SEARCH_VEL = 0.0 Vitesse du mouvement inital de prise d'origine, en unités machine par seconde. Une valeur de zéro suppose que la position courante est l'origine machine. Si votre machine n'a pas de contact d'origine, laissez 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, vous pouvez mettre sur Yes. Quand il est sur yes, il aura une incidence sur le type de séquence de prise d'origine utilisé.

HOME_IGNORE_LIMITS = NO Certaines machines utilisent un seul et même contact comme limite d'axe et origine machine de l'axe. Cette variable devra être positionnée sur yes si c'est le cas de votre machine.

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.

4.3.7.2 Variables relatives aux servomoteurs

Les entrées suivantes concernent les systèmes à servomoteurs, comme la carte du système univstep de Pico Systems.[3] Cette description suppose que les unités en sortie du composant PID sont des Volts.

P = 50 (HAL) La composante **p**roportionnelle 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, exemple : $\frac{volt}{mm}$ si l'unité machine est le millimètre.

I = 0 (HAL) La composante **i**ntégrale 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 : $\frac{volt}{mm\,s}$ si l'unité machine est le millimètre.

D = 0 (HAL) La composante **d**érivée 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 : $\frac{volt}{mm/s}$ si l'unité machine est le millimètre.

FF0 = 0 (HAL) Gain à priori (feedforward) d'orde 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 : $\frac{volt}{mm}$ si l'unité machine est le millimètre.

FF1 = 0 (HAL) Gain à priori (feedforward) de premier orde. 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 : $\frac{volt}{mm\,s}$ si l'unité machine est le millimètre.

FF2 = 0 (HAL) Gain à priori (feedforward) de second orde. 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 : $\frac{volt}{mm\,s^2}$ si l'unité machine est le millimètre.

OUTPUT_SCALE = 1.000

OUTPUT_OFFSET = 0.000 (HAL) Ces deux valeurs sont, l'échelle et le facteur d'offset de l'ampli moteur de cet axe. 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, EMC 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 :

$$raw = \frac{output - offset}{scale}$$

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/sec :

$$amplifier[volts] = (output[\frac{mm}{sec}] - offset[\frac{mm}{sec}])/250\frac{mm}{sec\,volt}$$

Notez que les unités d'offset sont en unités machine, exemple : mm/sec 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 normallement 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 :

[3] Référez vous au "Manuel de l'intégrateur d'EMC2" pour des informations complémentaires sur les systèmes à servomoteurs et leur contrôle en PID.

1. Construire un tableau de calibrage pour la sortie, piloter le DAC avec la tension souhaitée et mesurer le résultat. Voir le tableau 4.3.7.2 pour un exemple de mesures de tension.
2. Effectuer un "least squares" linéaire pour obtenir les coefficients a, b tels que :

$$meas = a * raw + b$$

3. Notez que nous voulons des sorties brutes de sorte que nos résultats mesurés soient identiques à la sortie commandée. Ce qui signifie :

   (a)
   $$cmd = a * raw + b$$

   (b)
   $$raw = (cmd - b)/a$$

4. En conséquence, les coefficients a et b d'ajustement linéaire peuvent être directement utilisés comme valeurs d'échelle et d'offset pour le contrôleur.

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 mis à l'échelle de la sortie en unités brutes.

MIN_OUTPUT = -10 (HAL) La valeur minimale 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 mis à l'échelle de la sortie en unités brutes.

Mesure des tensions de sortie

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

INPUT_SCALE = 20000 (HAL) Spécifie le nombre d'impulsions qui correspond à un mouvement d'une unité machine. 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 mm, nous avons :

$$\begin{aligned} input\_scale &= 2000\frac{impulsions}{tour} * 10\frac{tours}{pouce} \\ &= 20000\frac{impulsions}{pouce} \end{aligned}$$

4.3.7.3 Variables relatives aux moteurs pas à pas

SCALE = 4000 (HAL) Spécifie le nombre d'impulsions qui correspond à un mouvement d'une unité machine. Pour les sytèmes à moteurs pas à pas, c'est le nombre d'impulsions de pas nécessaires pour avancer d'une unité machine. Pour les systèmes à servo, c'est le nombre d'impulsions de retour signifiant que le mobile a avancé d'une unité machine. Un second chiffre, si spécifié, sera ignoré.

Par exemple, un pas moteur de 1.8 degré, en mode demipas, avec une réduction de 10 tours/pouce et des unités souhaitées en mm, nous avons :

$$\begin{aligned} input\_scale &= \frac{2\,pas}{1.8\,degre} * 360\frac{degres}{tour} * 10\frac{tours}{pouce} \\ &= 4000\frac{pas}{pouce} \end{aligned}$$

D'anciens fichiers de configuration .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 backlash 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 auss 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.

4.3.8 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 qu'EMCIO tournera en permanence. Il est préférable de ne pas modifier cette valeur.

TOOL_TABLE = tool.tbl Ce fichier contient les informations des outils.

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.

Chapitre 5

Prise d'origine

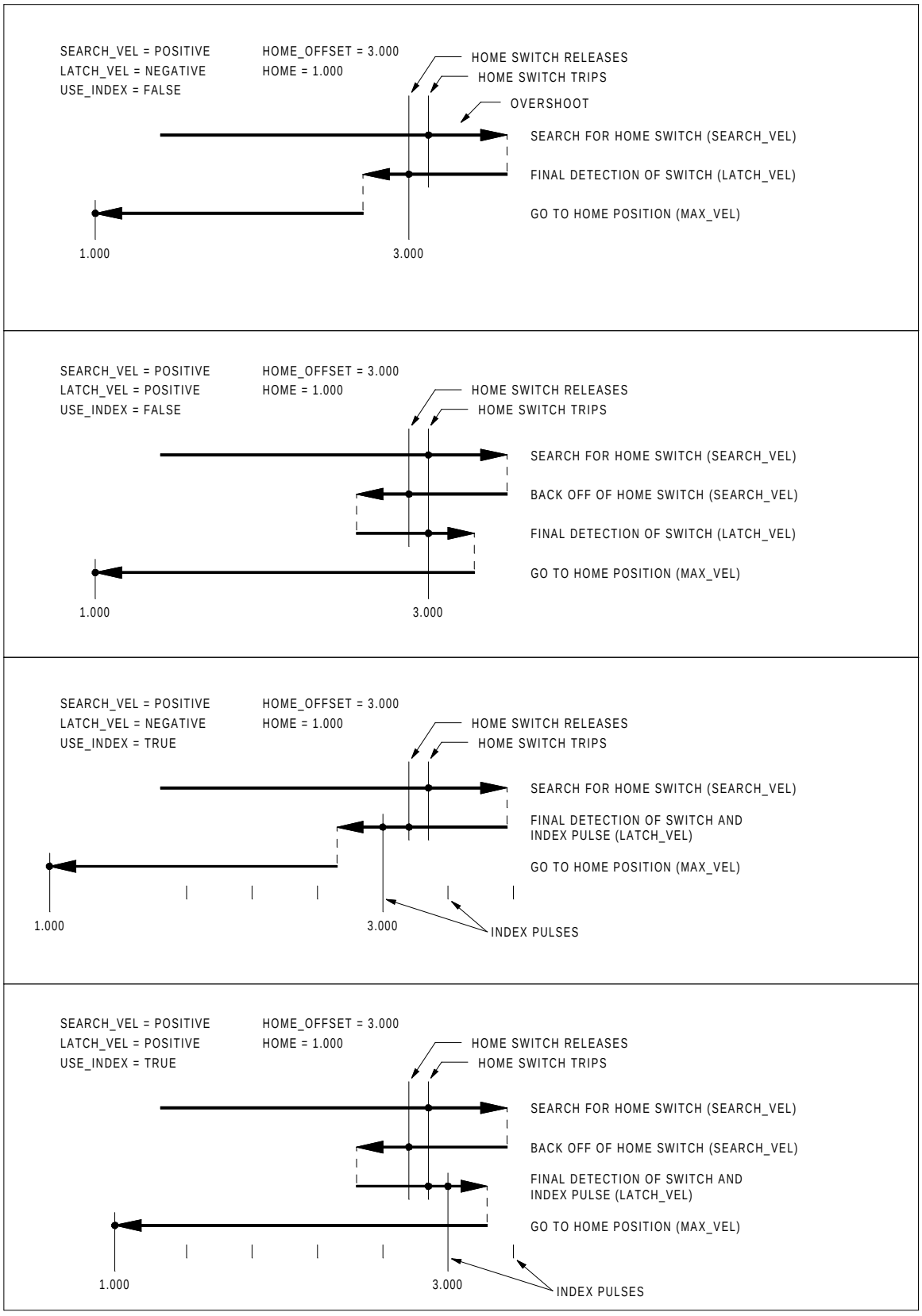
5.1 Vue d'ensemble

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 d'EMC en conséquence. Toutefois, les machines sont différentes les unes des autres et la prise d'origine est maintenant devenue assez complexe.

5.1.1 Séquence de prise d'origine

La figure 5.1.1 montre les quatre séquences de prise d'origine possibles, avec les variables de configuration associées 5.1. Une description détaillée de ces paramètres sera faite au chapitre suivant.

5.1.1



Les séquences de POM possibles

5.1.2 Configuration

Il y a six combinaisons possibles des variables qui définissent le déroulement de la séquence de prise d'origine. EIles sont définies dans la section [AXIS] du fichier ini.

SEARCH_VEL	LATCH_VEL	USE_INDEX	Type de séquence
nonzero	nonzero	NO	Switch-seulement
nonzero	nonzero	YES	Switch + Index
0	nonzero	YES	Index-seulement
0	0	NO	Aucun
Autres combinaisons			Erreur

TAB. 5.1 – Combinaisons des variables de la POM

5.1.2.1 HOME_SEARCH_VEL = 0

Vitesse de la phase initiale de prise d'origine, c'est la recherche du contact d'origine machine. Une valeur différente de zéro indique à EMC la présence d'un contact d'origine machine. EMC va alors commencer par vérifier si ce contact est déjà pressé. 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 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.

5.1.2.2 HOME_LATCH_VEL = 0

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 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 la position d'origine 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, EMC dégagera le contact à la vitesse définie ici. L'acquisition de la position d'origine 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. Si "HOME_SEARCH_VEL" est différent de zéro et que "HOME_LATCH_VEL" est égal à zéro, c'est une cause d'erreur, l'opération de prise d'origine échouera. La valeur par défaut est zéro.

5.1.2.3 HOME_IGNORE_LIMITS = YES/NO

Peut contenir les valeurs YES ou NO. Cette variable détermine si EMC 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, EMC 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.

5.1.2.4 HOME_USE_INDEX = YES/NO

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), EMC fera l'acquisition de l'origine machine sur le premier front de l'impulsion d'index. Si elle est fausse (=NO), EMC 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.

5.1.2.5 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 de l'impulsion d'origine, EMC ajuste les coordonnées de l'axe à la valeur de "HOME_OFFSET". La valeur par défaut est zéro.

5.1.2.6 HOME

C'est la position sur laquelle ira le mobile à la fin de la séquence de prise d'origine. 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.[1]

5.1.2.7 HOME_IS_SHARED

Si cet axe n'a pas un contact d'origine séparé des autres, mais plusieurs contacts câblés sur la même broche, mettez 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.

5.1.2.8 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", leur POM s'effectueront simultanément. Si "HOME_SEQUENCE" est égale à -1 ou non 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.

[1] La distinction entre l'origine machine et le décalage d'origine n'est pas aussi claire que je le voudrais. J'envisage de faire un petit dessin et un exemple pour la clarifier.

5.2 Tours

5.2.1 Plan par défaut

Quand l'interpréteur d'EMC 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 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
```

Chapitre 6

EMC2 et HAL

Voir également les man pages **motion(9)** et **iocontrol(1)**

6.1 motion (realtime)

Ces pins, paramètres et fonctions sont créés par le module temps réel "`motmod`".

6.1.1 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-nn (Float, In) Ces pins sont contrôlées par M66. Les valeurs pour nn valides sont : 00, 01, 02, 03.

motion.current-vel (Float, out) The current tool velocity

motion.digital-in-nn (bit, In) Ces pins sont contrôlées par M62 à M65. Les valeurs valides pour nn sont : 00, 01, 02, 03.

motion.digital-out-nn (bit, out) Ces pins sont controlées par les mots `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-enabled (bit, out) TRUE quand l'état de la machine est "machine on".

motion.motion-inpos (bit, In) TRUE si la machine est en position.

motion.probe-input (bit, In) `G38.x` utilise la valeur de cette pin pour déterminer quand la sonde de mesure entre en contact. TRUE le contact de la sonde est fermé (touche), FALSE le contact de la sonde est ouvert.

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 broche "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.

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-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.tooloffset.w (Float, out) montre l'effet de l'offset w. Il peut provenir de la table d'outils (G43 actif), ou du g-code (G43.1 actif)

motion.tooloffset.x (float, out) montre l'effet de l'offset x. Il peut provenir de la table d'outils (G43 actif), ou du g-code (G43.1 actif)

motion.tooloffset.z (float, out) montre l'effet de l'offset z. Il peut provenir de la table d'outils (G43 actif), ou du g-code (G43.1 actif)

6.1.2 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.coord-error (bit, RO) TRUE quand le mouvement est en erreur, ex : dépasser une limite soft.

motion.coord-mode (bit, RO) TRUE quand le mouvement est en "mode coordonnées" par opposition au "mode téléopération".

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.in-position (bit, RO) Identique à la pin **motion.motion-inpos**

motion.on-soft-limit (bit, RO)

motion.program-line (s32, RO)

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)

motion.servo.overruns En voyant de grandes différences entre les valeurs successives de **motion.servo.last-period**, le contrôleur de mouvement peut déterminer qu'il a eu un échec pour respecter ses contraintes de timing. Chaque fois qu'une erreur est détectée, cette valeur est incrémentée.

6.1.3 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 d'emc.

6.2 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**.

6.2.1 Pins

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.home-sw-in (bit, In) Doit être mis TRUE si le contact d'origine de cet axe est pressé.

axis.N.homing (bit, out) TRUE si la prise d'origine de cette axe a été faite.

axis.N.pos-lim-sw-in (bit, In) Doit être mis TRUE si le fin de course de limite positive de cet axe est activé.

axis.N.neg-lim-sw-in (bit, In) Doit être mis TRUE si le fin de course de limite négative de cet axe est activé.

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.

[1] Dans une machine à "cinématique triviale", il y a correspondance une pour une, entre les articulations et les axes.
NDT : 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-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".

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.motor-pos-cmd (Float, out) La position commandée pour cet axe.

axis.N.motor-pos-fb (Float, In) La position actuelle de cet axe.

axis.N.joint-pos-cmd Position commandée de l'axe (par opposition à celle du moteur). Il peut y avoir un décalage entre la position de l'axe et celle du moteur, par exemple, le processus de prise d'origine peut ajuster cet écart.

axis.N.joint-pos-fb Le retour de position (par opposition à celui du moteur).

6.2.2 Paramètres

Beaucoup de ces paramètres servent d'aide au déboguage et sont sujets aux changements ou au retrait à tout moment.

axis.N.active TRUE quand cet axe est actif.

axis.N.backlash-corr Valeur brute de ratrappage de jeu.

axis.N.backlash-filt Valeur filtrée de ratrappage de jeu (respect des limites de mouvement).

axis.N.backlash-vel Vitesse de ratrappage de jeu.

axis.N.coarse-pos-cmd

axis.N.error TRUE quand une erreur ce produit sur cet axe, ex : une limite de course est atteinte.

axis.N.f-error Erreur de suivi actuelle.

axis.N.f-error-lim Limite d'erreurs de suivi.

axis.N.f-errored TRUE quand cet axe à dépassé la limite d'erreurs de suivi.

axis.N.free-pos-cmd Position commandée en "free planner" pour cet axe.

axis.N.free-tp-enable TRUE quand le "free planner" est activé pour cet axe.

axis.N.free-vel-lim Vitesse limite en "free planner".

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

axis.N.homed TRUE si la prise d'origine de cet axe a bien été réalisée.

axis.N.in-position TRUE si cet axe, utilisant le "free planner", a atteint un arrêt.

axis.N.joint-vel-cmd Vitesse commandée des axes.

axis.N.neg-hard-limit Fin de course de limite d'axe négative.

axis.N.neg-soft-limit Limite soft négative de cet axe.

axis.N.pos-hard-limit Fin de course de limite d'axe positive.

axis.N.pos-soft-limit Limite soft positive de cet axe.

6.3 iocontrol (espace utilisateur)

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

6.3.1 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 enfoncé.

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 assez haut.

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 dans 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 enfoncé.

iocontrol.0.user-request-enable (bit, out) TRUE quand l'utilisateur relâche l'arrêt d'urgence.

Troisième partie

Spécificités de HAL

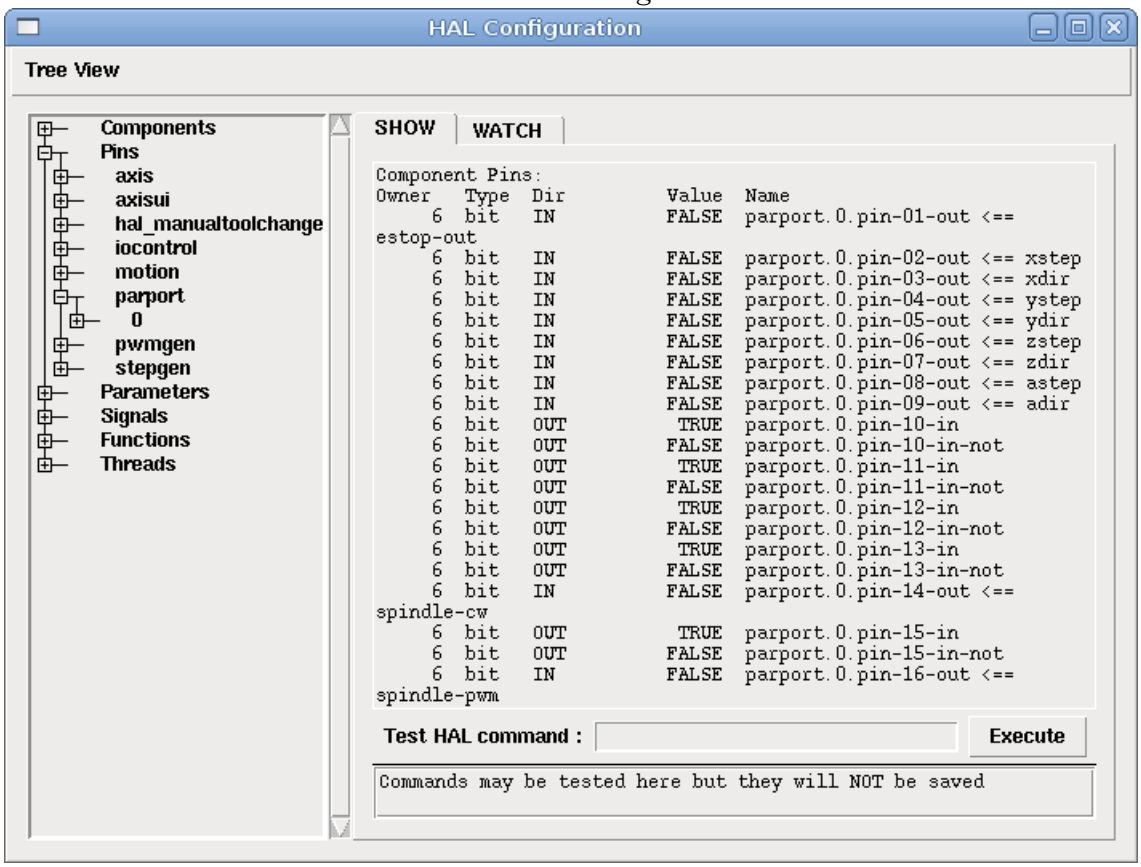
Chapitre 7

Les bases de HAL

7.1 Commandes de Hal

Des informations plus détaillées peuvent être trouvées dans la man page en tapant "man halcmd" dans une console. Pour voir la configuration de HAL ainsi que le status de ses pins et paramétres utiliser la fenêtre HAL Configuration dans le menu « Machine » d'AXIS. Pour visualiser le status des pins, ouvrir l'onglet « Watch » puis cliquer dans l'arborescence sur les pins qui doivent être visualisées dans la fenêtre watch.

FIG. 7.1 – Fenêtre de configuration de HAL



7.1.1 loadrt

La commande "loadrt" charge un composant temps réel de HAL. Les composants temps réel doivent être ajoutés au thread temps réel pour être fonctionnels. Il n'est pas possible de charger un composant de l'espace utilisateur dans l'espace temps réel.

Syntaxe et exemple :

loadrt <composant> <options>
loadrt mux4 count=1

7.1.2 addf

La commande "addf" ajoute une fonction à un thread temps réel. Si l'assistant StepConf a été utilisé pour créer la configuration, deux threads ont été créés.

– base-thread (le thread haute vitesse) ce thread prends en main les items nécessitant une réponse très rapide comme la génération d'impulsions, la lecture et l'écriture sur le port parallèle.
– servo-thread (le thread basse vitesse) ce thread prends en main les items n'étant pas influencés par la vitesse comme le contrôleur de mouvement, l'API ClassicLadder et les commandes manuelles.

Syntaxe et exemple :

addf <composant> <thread>
addf mux4 servo-thread

7.1.3 loadusr

La commande "loadusr" charge un composant de HAL de l'espace utilisateur. Les programmes de l'espace utilisateur ont leur propre process séparé qui optionellement communique avec les autres composants de HAL via leurs pins et paramètres. Il n'est pas possible de charger un composant temps réel dans l'espace utilisateur.

Les drapeaux peuvent être un ou plusieurs parmis les suivants :

-W pour attendre que le composant soit prêt. Le composant est supposé avoir le même nom que le premier argument de la commande.

-Wn <nom> pour attendre un composant, qui porte le nom donné sous la forme <nom>.

-w pour attendre la fin du programme

-i pour ignorer la valeur retournée par le programme (avec -w)

Syntaxe et exemple :

loadusr <composant> <options>
loadusr halui
loadusr -Wn spindle gs2_vfd -n spindle
En anglais ça donne "loadusr wait for name spindle component gs2_vfd name spindle."
Le -n spindle est une partie du composant gs2_vfd et non de la commande loadusr.

7.1.4 net

La commande "net" crée une "connection" entre un signal et une ou plusieurs pins. Les indicateurs de direction "<= et =>" sont seulement des aides à la lecture, ils n'ont pas d'autre utilité.

Syntaxe et exemple :

net <signal-name> <pin-name> <opt-direction> <opt-pin-name>
net both-home-y <= parport.0.pin-11-in

Chaque signal ne peut avoir qu'une seule source (une seule pin de HAL "out") et autant de « lecteurs » (des pins de HAL "in") que souhaité. Dans la colonne Dir de la fenêtre de configuration de HAL il est possible de voir quelles pins sont "in" et quelles pins sont "out".

Pour faire celà en une ligne :

net xStep stepgen.0.out => parport.0.pin-02-out parport.0.pin-08-out

Ou pour le faire en plusieurs lignes, utiliser simplement le signal avec les lecteurs des lignes suivantes :

net xStep stepgen.0.out => parport.0.pin-02-out
net xStep => parport.0.pin-02-out

Les pins appelées I/O pins comme « index-enable », ne suivent pas cette règle.

7.1.5 setp

La commande "setp" ajuste la valeur d'une pin ou d'un paramètre. Les valeurs valides dépendront du type de la pin ou du paramètre.

- bit = true ou 1 et false ou 0 (True, TRUE, true sont toutes valides)
- float = un flottant sur 32 bits, avec approximativement 24 bits de résolution et au plus 200 bits d'étendue dynamique.
- s32 = un nombre entier compris entre -2147483648 et 2147483647
- u32 = un nombre entier compris entre 0 et 4294967295

Pour des informations sur les flottants voir ici (en anglais) :

http://en.wikipedia.org/wiki/Floating_point

Les paramètres peuvent être positionnés avant utilisation ou pendant l'utilisation, toutefois certains composants ont des paramètres qui doivent être positionnés avant utilisation. Il n'est pas possible d'utiliser « setp » sur une pin connectée à un signal.

Syntaxe et exemple :

setp <pin/parameter-name> <value>
setp paraport.0.pin-08-out TRUE

7.1.6 Quatre commandes obsolètes

7.1.6.1 linksp

The command "linksp" creates a "connection" between a signal and one pin.

Syntaxe et exemple :

linksp <signal-name> <pin-name>
linksp X-step parport.0.pin-02-out

La commande "linksp" a été incluse dans la commande "net".

7.1.6.2 linkps

The command "linkps" creates a "connection" between one pin and one signal. It is the same as linksp but the arguments are reversed.

Syntaxe et exemple :

linkps <pin-name> <signal-name>
linkps parport.0.pin-02-out X-Step

La commande "linkps" a été incluse dans la commande "net".

7.1.6.3 unlinkp

The command "unlinkp" unlinks a pin from the connected signal. If no signal was connected to the pin prior running the command, nothing happens.

Syntaxe et exemple :

unlinkp <pin-name>
unlinkp parport.0.pin-02-out

7.1.6.4 newsig

the command "newsig" creates a new HAL signal by the name <signame> and the data type of <type>. Type must be "bit", "s32", "u32" or "float". Error if <signame> all ready exists.

Syntaxe et exemple :

newsig <signame> <type>
newsig Xstep bit

D'autres informations peuvent être trouvées dans le manuel de HAL ou la man page de « halrun ».

7.2 Fichiers Hal

Si l'assistant StepConf a été utilisé pour générer la configuration trois fichiers HAL ont dû être créés dans le répertoire de la configuration.

- ma-fraiseuse.hal (si ne nom de la config est nomée "ma-fraiseuse") Ce fichier est chargé en premier, il ne doit pas être modifié sous peine de ne plus pouvoir l'utiliser avec l'assistant StepConf.
- custom.hal Ce fichier est le deuxième à être chargé et il l'est avant l'interface utilisateur graphique (GUI). C'est dans ce fichier que ce trouvent les commandes personnalisées de l'utilisateur devant être chargées avant la GUI.
- custom_postgui.hal Ce fichier est chargé après la GUI. C'est dans ce fichier que se trouvent les commandes personnalisées de l'utilisateur devant être chargées après la GUI. Toutes les commandes relatives aux widgets de pyVCP doivent être placées ici.

7.3 Composants de logiques combinatoire

Hal contient plusieurs composants logiques temps réel. Les composants logiques suivent une tables de vérité montrant les états logiques des sorties en fonction de l'état des entrées. Typiquement, la manipulation des bits d'entrée détermine l'état électrique des sorties selon la table de vérité des portes.

7.3.1 and2

Le composant "and2" est une porte "and" à deux entrées. Sa table de vérité montre la sortie pour chaque combinaison des entrées.

Syntaxe

and2 [count=N|names=name1[,name2...]]

Fonctions

and2.n

Pins

and2.N.in0 (bit, in)
and2.N.in1 (bit, in)
and2.N.out (bit, out)

Table de vérité

in0	in1	out
False	False	False
True	False	False
False	True	False
True	True	True

7.3.2 not

Le composant "not" est un simple inverseur d'état.

Syntaxe

not [count=n|names=name1[,name2...]]

Fonctions

not.all
not.n

Pins

not.n.in (bit, in)
not.n.out (bit, out)

Table de vérité

in	out
True	False
False	True

7.3.3 or2

Le composant "or2" est une porte OR à deux entrées.

Syntaxe

or2[count=,|names=name1[,name2...]]

Fonctions

or2.n

Pins

or2.n.in0 (bit, in)
or2.n.in1 (bit, in)
or2.n.out (bit, out)

Table de vérité

in0	in1	out
True	False	True
True	True	True
False	True	True
False	False	False

7.3.4 xor2

Le composant "xor2" est une porte XOR à deux entrées (OU exclusif).

Syntaxe

xor2[count=,|names=name1[,name2...]]

Fonctions

xor2.n

Pins

xor2.n.in0 (bit, in)
xor2.n.in1 (bit, in)
xor2.n.out (bit, out)

Table de vérité

in0	in1	out
True	False	True
True	True	False
False	True	True
False	False	False

7.3.5 Exemples de logique combinatoire

Un exemple de connection avec un "and2", deux entrées vers une sortie.

```
loadrt and2 count=1
addf and2.0 servo-thread
net my-sigin1 and2.0.in0 <= parport.0.pin-11-in
net my-sigin2 and2.0.in.1 <= parport.0.pin-12-in
net both-on parport.0.pin-14-out <= and2.0.out
```

Dans cet exemple un and2 est chargé dans l'espace temps réel, puis ajouté à servo thread. Ensuite la broche d'entrée 11 du port parallèle est connectée à l'entrée in0 de la porte. Puis la broche d'entrée 12 du port est connectée à l'entrée in1 de la porte. Enfin la sortie and2.0.out de la porte est connectée à la broche de sortie 14 du port parallèle. Ainsi en suivant la table de vérité du and2, si les broches 11 et 12 du port sont à 1, alors sa sortie 14 est à 1 aussi.

7.4 Halshow

Le script halshow peut vous aider à retrouver votre chemin dans un HAL en fonctionnement. Il s'agit d'un système très spécialisé qui doit se connecter à un HAL en marche. Il ne peut pas fonctionner seul car il repose sur la capacité de HAL de rapporter ce qu'il connaît de lui même par la librairie d'interface de halcmd. Chaque fois que halshow fonctionne avec une configuration d'EMC différente, il sera différent.

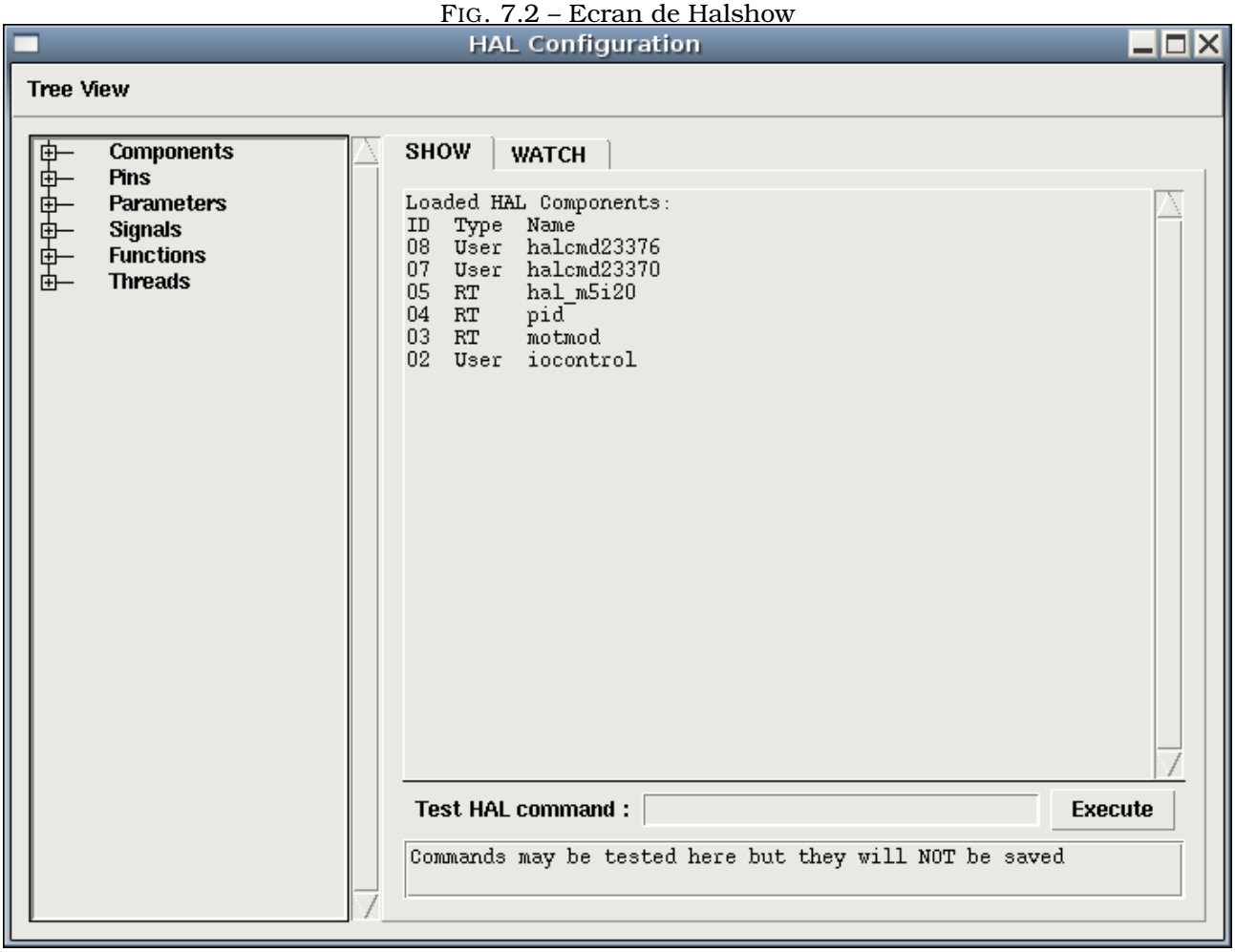
Comme nous le verrons bientôt, cette capacité de HAL de se documenter lui même est un des facteurs clés pour arriver à un système CNC optimum.

On peut accéder à Halshow depuis Axis, pour cela, aller dans le menu "Machine" puis choisir "Afficher la configuration de HAL".

7.4.1 Zone de l'arborescence de Hal

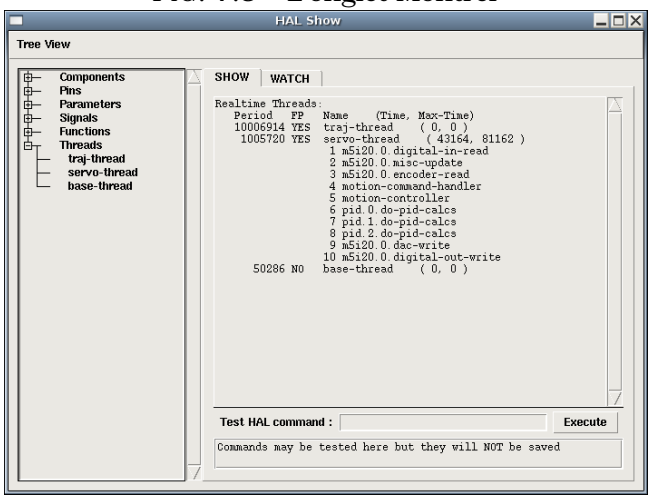
La gauche de l'écran que montre la figure 7.2 est une arborescence, un peu comme vous pouvez le voir avec certains navigateurs de fichiers. Sur la droite, une zone avec deux onglets : MONTRER et WATCH.

FIG. 7.2 – Ecran de Halshow



L'arborescence montre toutes les parties principales de HAL. En face de chacune d'entre elles, se trouve un petit signe + ou - dans une case. Cliquer sur le signe plus pour déployer cette partie de

FIG. 7.3 – L'onglet Montrer



l'arborescence et affichera son contenu. Si cette case affiche un signe moins, cliquer dessus repliera cette section de l'arborescence.

Il est également possible de déployer et de replier l'arborescence complète depuis le menu "Arborescence".

7.4.2 Zone de l'onglet MONTRER

En cliquant sur un nom dans l'arborescence plutôt que sur son signe plus ou moins, par exemple le mot "Components", HAL affichera tout ce qu'il connait du contenu de celui-ci. La figure 7.2 montre une liste comme celle que vous verrez si vous cliquez sur "Components" avec une carte servo standard m5i20 en fonctionnement. L'affichage des informations est exactement le même que celui des traditionnels outils d'analyse de HAL en mode texte. L'avantage ici, c'est que nous y avons accès d'un clic de souris. Accès qui peut être aussi large ou aussi focalisé que vous le voulez.

Si nous examinons de plus près l'affichage de l'arborescence, nous pouvons voir que les six éléments principaux peuvent tous être déployés d'au moins un niveau. Quand ces niveaux sont à leur tour déployés vous obtenez une information de plus en plus focalisée en cliquant sur le nom des éléments dans l'arborescence. Vous trouverez que certaines hal pins et certains paramètres affichent plusieurs réponses. C'est dû à la nature des routines de recherche dans halcmd lui même. Si vous cherchez une pin, vous pouvez en trouver deux comme cela :

```
Component Pins :
Owner Type Dir Value Name
06 bit -W TRUE parport.0.pin-10-in
06 bit -W FALSE parport.0.pin-10-in-not
```

Le deuxième nom de pin contient le nom complémenté du premier.

Dans le bas de l'onglet Montrer, un champ de saisie permet de jouer sur le fonctionnement de HAL. Les commandes que vous entrez ici et leur effet sur HAL, ne sont pas enregistrés. Elles persisteront tant qu'EMC tournera, mais disparaîtront dès son arrêt.

Le champ de saisie marqué "Tester une commande HAL :" acceptera n'importe quelle commande valide pour halcmd. Elles incluent :

– loadrt, unloadrt
– addf, delf
– newsig, delsig
– linkpp, linksp, linkps, unlinkp

– setp, sets

Ce petit éditeur entrera une commande à chaque fois que vous presserez <Entrée> ou que vous cliquerez sur le bouton “Exécuter”. Si une commande y est mal formée, un dialogue d’erreur s’affichera. Si vous n’êtes pas sûr de savoir comment formuler une commande, vous trouverez la réponse dans la documentation de halcmd et des modules spécifiques avec lesquels vous travaillez.

Nous allons utiliser cet éditeur pour ajouter un module différentiel à HAL et le connecter à la position d’un axe pour voir le ratio de changement de position, par exemple, l’accélération. Il faut d’abord charger un module de HAL nommé blocks, l’ajouter au thread servo et le connectet à la pin position d’un axe. Une fois cela fait, nous pourrons retrouver la sortie du différentiateur dans halscope. Alors allons-y. (oui j’ai vérifié).

```
loadrt blocks ddt=1
```

Maintenant, regardez dans components, vous devriez y voir blocks.

```
Loaded HAL Components :
ID Type Name
10 User halcmd29800
09 User halcmd29374
08 RT blocks
06 RT hal_parport
05 RT scope_rt
04 RT stepgen
03 RT motmod
02 User iocontrol
```

Effectivement, il est là. Dans notre cas l’id est 08. Ensuite nous devons savoir quelles fonctions sont disponibles avec lui, nous regardons dans Functions.

```
Exported Functions :
Owner CodeAddr Arg FP Users Name
08 E0B97630 E0DC7674 YES 0 ddt.0
03 E0DEF83C 00000000 YES 1 motion-command-handler
03 E0DF0BF3 00000000 YES 1 motion-controller
06 E0B541FE E0DC75B8 NO 1 parport.0.read
06 E0B54270 E0DC75B8 NO 1 parport.0.write
06 E0B54309 E0DC75B8 NO 0 parport.read-all
06 E0B5433A E0DC75B8 NO 0 parport.write-all
05 E0AD712D 00000000 NO 0 scope.sample
04 E0B618C1 E0DC7448 YES 1 stepgen.capture-position
04 E0B612F5 E0DC7448 NO 1 stepgen.make-pulses
04 E0B614AD E0DC7448 YES 1 stepgen.update-freq
```

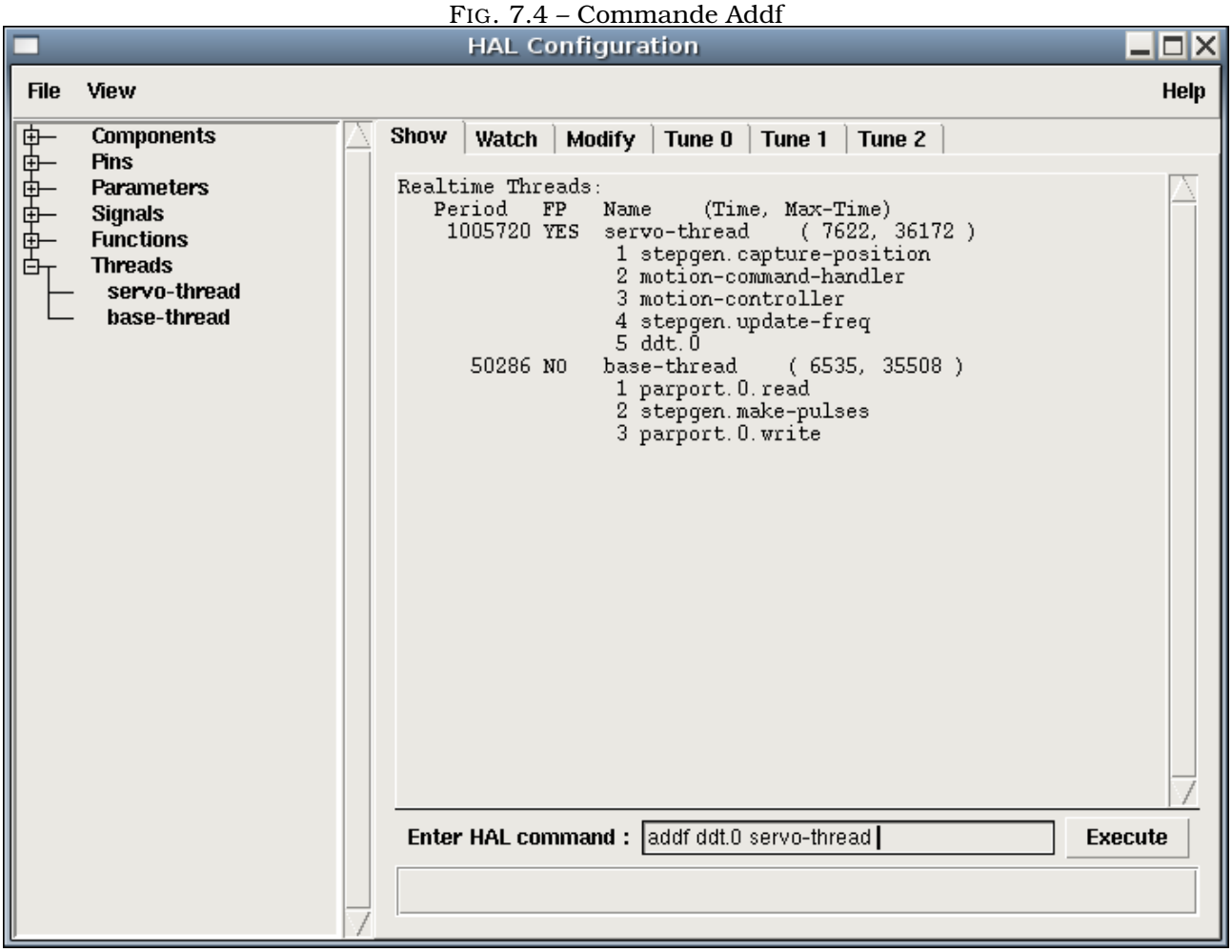
Ici, nous cherchons owner #08 et voyons que blocks a exporté une fonction nommée ddt.0. Nous devrions être en mesure d’ajouter ddt.0 au thread servo et il fera ses calculs chaque fois que le thread sera mis à jour. Encore une fois recherchons la commande addf et on voit qu’elle utilise trois arguments comme cela :

```
addf <functname> <threadname> [<position>]
```

Nous connaissons déjà functname=ddt.0, pour trouver le nom du thread, déployons l’arborescence des Threads. Nous y trouvons deux threads, servo-thread et base-thread. La position de ddt.0 dans le thread n’est pas critique. Passons la commande :

```
addf ddt.0 servo-thread
```

FIG. 7.4 – Commande Addf



Comme c'est juste pour visualiser, nous laissons la position en blanc pour obtenir la dernière position dans le thread. La figure 7.4 montre l'état de halshow après que cette commande a été exécutée.

Ensuite, nous devons connecter ce block à quelque chose. Mais comment savoir quelles pins sont disponibles ? La réponse se trouve dans l'arbre, en regardant sous Pins. On y trouve ddt et on voit :

```
Component Pins :
Owner Type Dir Value Name
08 float R- 0.00000e+00 ddt.0.in
08 float -W 0.00000e+00 ddt.0.out
```

Cela semble assez facile à comprendre, mais à quel signal ou pin voulons-nous nous connecter, ça pourrait être une pin d'axe, une pin de stepgen, ou un signal. On vois cela en regardant dans axis.0.

```
Component Pins :
Owner Type Dir Value Name
03 float -W 0.00000e+00 axis.0.motor-pos-cmd ==> Xpos-cmd
```

Donc, il semble que Xpos-cmd devrait être un bon signal à utiliser. Retour à l'éditeur et entrons la commande suivante :

```
linksp Xpos-cmd ddt.0.in
```

Maintenant si on regarde le signal Xpos-cmd dans l'arbre, on voit ce qu'on a fait.

```
Signals :
Type Value Name
float 0.00000e+00 Xpos-cmd
<== axis.0.motor-pos-cmd
==> ddt.0.in
==> stepgen.0.position-cmd
```

Nous voyons que ce signal provient de axis.0.motor-pos-cmd et va, à la fois, sur ddt.0.in et sur stepgen.0.position-cmd. En connectant notre block au signal nous avons évité les complications avec le flux normal de cette commande de mouvement.

La zone de l'onglet "Montrer" utilise halcmd pour découvrir ce qui se passe à l'intérieur de HAL pendant son fonctionnement. Il vous donne une information complète de ce qu'il découvre. Il met aussi à jour dès qu'une commande est envoyée depuis le petit éditeur pour modifier ce HAL. Il arrive un temps ou vous voulez autre chose d'affiché, sans la totalité des informations disponibles dans cette zone. C'est la grande valeur de l'onglet WATCH d'offrir cela.

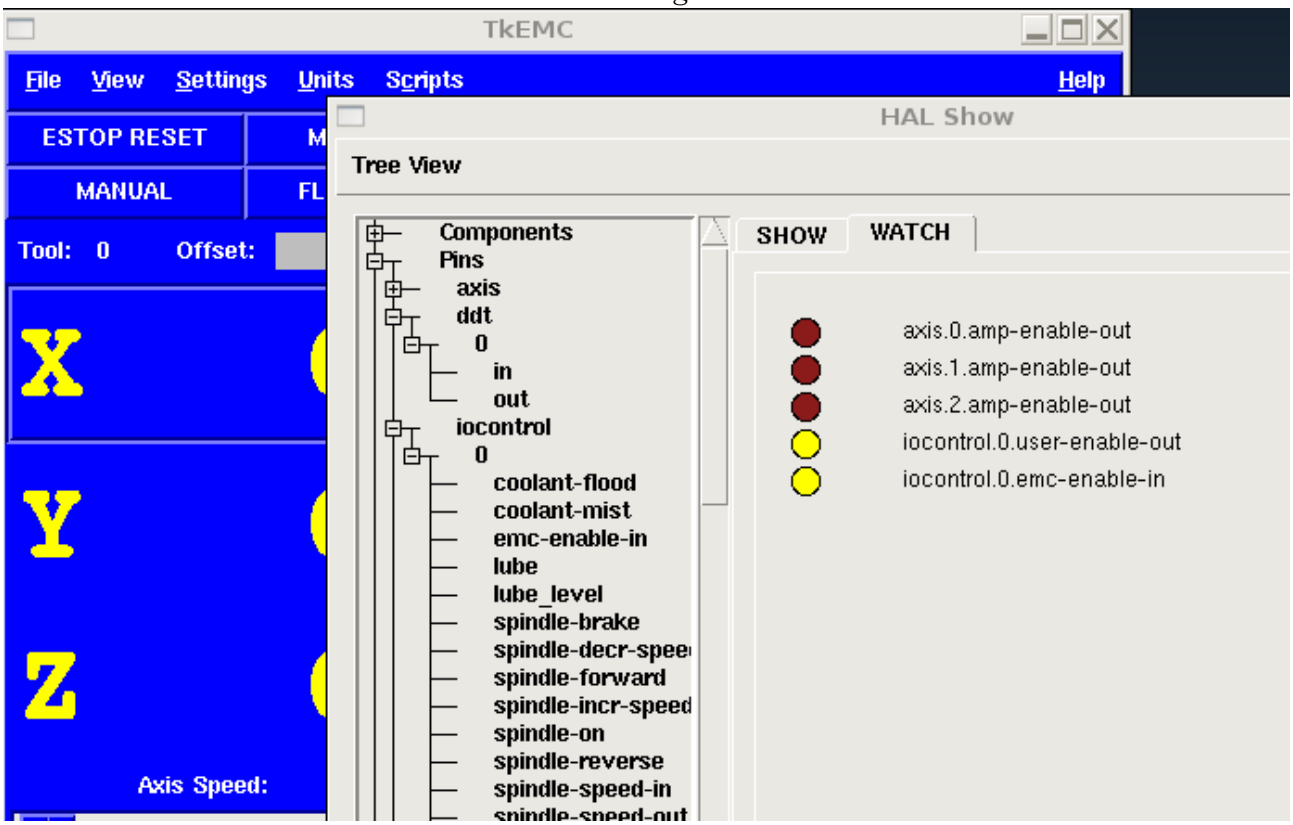
7.4.3 Zone de l'onglet WATCH

En cliquant sur l'onglet WATCH une zone vide s'affichera. Vous pouvez ajouter des signaux et des pins dans cette zone et visualiser leurs valeurs.[1] Vous pouvez ajouter des pins ou des signaux quand l'onglet Watch est ouvert, en cliquant sur leurs noms. La figure 7.5 Montre cette zone avec plusieurs signaux de type "bit". Parmis ces signaux, les enable-out pour les trois premiers axes et deux de la branche iocontrol, les signaux "estop". Notez que les axes ne sont pas activés même si les signaux estop disent qu'EMC n'est pas en estop. Un bref regard sur themc en arrière plan, montre que l'état d'EMC est ESTOP RESET. L'activation des amplis ne deviendra pas vraie tant que la machine ne sera pas mise en marche.

Les cercles de deux couleurs, simili leds, sont toujours bruns foncé quand un signal est faux. Elle sont jaunes quand le signal est vrai. Quand une pin ou un signal est sélectionné mais n'est pas de type bit, sa valeur numérique s'affiche.

[1] Le taux de rafraîchissement de la zone Watch est plus lent que celui de Halmeter ou de Halscope. Si vous avez besoin d'une bonne résolution dans le timming des signaux, ces outils sont plus efficaces.

FIG. 7.5 – L'onglet Watch



Watch permet de visualiser rapidement le résultat de tests sur des contacts ou de voir l'effet d'un changement que vous faites dans EMC en utilisant l'interface graphique. Le taux de rafraîchissement de Watch est un peu trop lent pour visualiser les impulsions de pas d'un moteur mais vous pouvez l'utiliser si vous déplacez un axe très lentement ou par très petits incréments de distance. Si vous avez déjà utilisé IO_Show dans EMC, la page de Watch de halshow peut être réglée pour afficher ce que fait le port parallèle.

Chapitre 8

Les bases de la configuration pour 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 d'EMC2. En raison de l'adaptabilité d'EMC2, il serait très difficile de les documenter tous en gardant ce document relativement concis.

Le système rencontré le plus fréquemment chez les utilisateurs d'EMC2 est un système à moteurs pas à pas. Les interfaces de pilotage de ces moteurs recoivent d'EMC2 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 EMC2 appelée « stepper » et qui se trouve habituellement dans `/etc/emc2/sample-configs/stepper`.

8.2 Fréquence de pas maximum

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 EMC2)

8.3 Brochage

EMC2 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 la section **??** pour des informations complètes à propos de HAL).

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 pin relatifs au brochage sont[1] :

```
signaux : Xstep, Xdir et Xen
pins : parport.0.pin-XX-out et parport.0.pin-XX-in [2]
```

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 ».

8.3.1 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

###
```

[1] Note : pour rester concis, nous ne présenterons qu'un seul axe, tous les autres sont similaires.
[2] Se référer à la section 12.1 pour plus d'information

```
### 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

###
### 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 EMC'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, EMC 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, aidant à la lecture du fichier.

8.3.2 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 12.1 pour plus de détails)

2. Les fonctions de lecture/écriture du pilote sont assignée au thread « Base thread » [3]
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

8.3.3 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' :

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

peut être modifiée pour devenir :

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

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

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

8.3.4 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
```

8.3.5 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 # Ajustez cette valeur à la vitesse max de votre broche en tr
```

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`).

[3] Le thread le plus rapide parmis les réglages d'EMC2, habituellement il n'y a que quelques microsecondes entre les exécutions de ce code.

8.3.6 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 :

```
linksp 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.

8.3.7 Ajouter un bouton d'Arrêt d'Urgence externe

Comme vous pouvez le voir à la section 8.3.1, par défaut la configuration standard n'utilise pas de bouton d'Arrêt d'Urgence externe. [4]

Pour ajouter un simple bouton externe (ou plusieurs en série) vous devez compléter les lignes suivantes :

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

avec

```
net estop-out <= iocontrol.0.user-enable-out
net estop-ext     <= parport.0.pin-01-in
net estop-ext => 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)[5], EMC2 restera dans l'état « Arrêt d'Urgence » (ESTOP). Quand le bouton externe sera relâché, EMC2 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 EMC2.

[4] Une explication complète sur la manière de gérer les circuiteries d'Arrêt d'Urgence se trouve sur le wiki.linuxcnc.org (en) et dans le Manuel de l'intégrateur

[5] Utiliser exclusivement des contacts normalement fermés pour les A/U.

Chapitre 9

Les composants de HAL

9.1 Composants de commandes et composants de l'espace utilisateur

Certaines de ces descriptions sont plus approfondies dans leurs pages man. Certaines auront juste une description limitée. Chaque composant a sa page man. A partir de cette liste vous connaîtrez quels composants existent avec le nom et le N° de leur page man permettant d'avoir plus de détails. Par exemple dans une fenêtre de terminal tapez **`man 1 axis`** pour accéder aux informations de cette page man.

axis-remote.1 = Interface de télécommande d'AXIS

axis.1 = AXIS EMC (The Enhanced Machine Controller) Interface Graphique Utilisateur

bfload.1 = Pour charger un programme « Xilinx Bitfile » dans le FPGA de n'importe quelle carte d'entrées/sorties de Mesa

comp.1 = Créer, compiler et installer des composants de EMC HAL

emc.1 = EMC (The Enhanced Machine Controller)

hal_input.1 = Contrôler des pins d'entrée de HAL avec n'importe quel matériel y compris les matériels USB HID

halcmd.1 = Manipuler HAL depuis la ligne de commandes

halmeter.1 = Observer les pins de HAL, ses signaux et ses paramètres

halrun.1 = Manipuler HAL depuis la ligne de commandes

halsampler.1 = Echantillonner des données temps réel depuis HAL

halstreamer.1 = Créer un flux de données temps réel dans HAL depuis un fichier

halui.1 = Observer des pins de HAL et commander EMC au travers d'NML

io.1 = Commandes NML I/O acceptées et interactions avec HAL dans l'espace utilisateur

iocontrol.1 = Commandes NML I/O acceptées et interactions avec HAL dans l'espace utilisateur

pyvcp.1 = Virtual Control Panel (Panneau de Contrôle Virtuel) pour EMC2

9.2 Composants temps réel et modules du noyau

Certaines de ces descriptions sont plus approfondies dans leur man page. Certaines auront juste une description limitée. Chaque composant a sa man page. A partir de cette liste vous connaîtrez quels composants existent avec le nom et le N° de leur man page permettant d'avoir plus de détails.

abs.9 = Calcule la valeur absolue et le signe d'un signal d'entrée
and2.9 = Porte AND (ET) à deux entrées
at_pid.9 = Contrôleur Proportionelle/Intégrale/dérivée avec réglage automatique
axis.9 = Commandes de mouvement NML acceptées, intéractions en temps réel avec HAL
biquad.9 = Filtre biquad IIR
blend.9 = Provoque une interpolation linéaire entre deux valeurs
blocks.9 = Old style HAL blocks (deprecated)
charge_pump.9 = Crée un signal carré destiné à l'entrée « pompe de charge » de certaines cartes de contrôle
clarke2.9 = Transformation de Clarke, version à deux entrées
clarke3.9 = Transformation de Clarke (3 triphasé/cartésien)
clarkeinv.9 = Transformation de Clarke inverse
classicladder.9 = Automate temps réel programmable en logique Ladder
comp.9 = Comparateur à deux entrées avec hystérésis
constant.9 = Utilise un paramètre pour positionner une pin
conv_bit_s32.9 = Converti une valeur de bit vers s32
conv_bit_u32.9 = Converti une valeur de bit vers u32
conv_float_s32.9 = Converti la valeur d'un flottant vers s32
conv_float_u32.9 = Converti la valeur d'un flottant vers u32
conv_s32_bit.9 = Converti une valeur de s32 en bit
conv_s32_float.9 = Converti une valeur de u32 en bit
conv_s32_u32.9 = Converti une valeur de s32 en u32
conv_u32_bit.9 = Converti une valeur de u32 en bit
conv_u32_float.9 = Converti une valeur de u32 en flottant
conv_u32_s32.9 = Converti une valeur de u32 en s32
counter.9 = counts input pulses (deprecated)
ddt.9 = Calcule la dérivée de la fonction d'entrée
deadzone.9 = Retourne le centre si il est dans le seuil
debounce.9 = Filtre une entrée digitale bruitée
edge.9 = Détecteur de front
encoder.9 = Comptage logiciel des signaux en quadrature d'un codeur
encoder_ratio.9 = Un engrenage électronique pour synchroniser deux axes
estop_latch.9 = Verrou d'Arrêt d'Ugence
flipflop.9 = Bascule D
freqgen.9 = Générateur logiciel d'impulsions de pas
genhexkins.9 = Définition de cinématique pour emc2
hypot.9 = Calculateur d'hypoténuse à trois entrées (distance Euclidienne)
integ.9 = Intégrateur
kins.9 = Définition de cinématique pour emc2
knob2float.9 = Convertisseur de comptage (probablement d'un codeur) vers une valeur en virgule flottante
limit1.9 = Limite le signal de sortie pour qu'il soit entre min et max
limit2.9 = Limite le signal de sortie pour qu'il soit entre min et max
limit3.9 = Limite le signal de sortie pour qu'il soit entre min et max

logic.9 =
lowpass.9 = Filtre passe-bas
lut5.9 = Cinq fonctions logiques arbitraires basée sur une table de recherche
m7i43_hm2.9 = RTAI driver for the Mesa 7i43 EPP Anything IO board with HostMot2 firmware
maj3.9 = Calcule la majorité parmis 3 entrées
match8.9 = Détecteur de coïncidence binaire sur 8 bits
minmax.9 = Suiveur de valeur minimum et maximum de l'entrée vers les sorties
motion.9 = Commandes de mouvement NML acceptées, intéractions en temps réel avec HAL
mult2.9 = Le produit de deux entrées
mux2.9 = Sélection d'une valeur d'entrée sur deux
mux4.9 = Sélection d'une valeur d'entrée sur quatre
not.9 = Inverseur
offset.9 = Additionne un offset à une entrée et la soustrait à la valeur de retour
oneshot.9 = Générateur d'impulsion monostable
or2.9 = Porte OR (OU) à deux entrées
pid.9 = Contrôleur Proportionelle/Intégrale/dérivée, pour plus de détails voir (???)
pluto_servo.9 = Pilote matériel et microprogramme pour le « Pluto-P parallel-port FPGA », utilisation avec servomoteurs
pluto_step.9 = Pilote matériel et microprogramme pour le « Pluto-P parallel-port FPGA », utilisation avec moteurs pas à pas
pwmgen.9 = Générateur logiciel de PWM/PDM, pour plus de détails voir (???)
rotatekins.9 = Définition de cinématique pour emc2
sample_hold.9 = Echantillonneur bloqueur
sampler.9 = Echantillonneur de données de HAL en temps réel
scale.9 =
select8.9 = Détecteur de coïncidence binaire sur 8 bits
serport.9 = Pilote matériel pour les circuits d'entrées/sorties digitales de port série 8250 et 16550
siggen.9 = Générateur de signal11.7
sim_encoder.9 = Simulation d'un codeur en quadrature, pour plus de détails voir (???)
stepgen.9 = Générateur d'impulsions de pas logiciel11.1
steptest.9 = Utilisé par Stepconf pour permettre de tester les valeurs d'accélération et de vitesse d'un axe
streamer.9 = Flux temps réel depuis un fichier vers HAL
sum2.9 = La somme de deux entrées (chacune avec un gain) et l'écart
supply.9 = set output pins with values from parameters (deprecated)
threads.9 = Créer des threads temps réel de HAL
threadtest.9 =
timedelta.9 =
toggle.9 = Bouton à bascule NO/NF
tripodkins.9 = Définition de cinématique pour emc2
tristate_bit.9 = Place un signal sur une pin d'I/O seulement quand elle est validée, similaire à un tampon trois états en électronique
tristate_float.9 = Place un signal sur une pin d'I/O seulement quand elle est validée, similaire à un tampon trois états en électronique
trivkins.9 = Définition de cinématique pour emc2
updown.9 = Compteur/décompteur avec limites optionnelles et wraparound behavior
wcomp.9 = Comparateur à fenêtre
weighted_sum.9 = Convertir un groupe de bits en un entier
xor2.9 = Porte XOR (OU exclusif) à deux entrées

Chapitre 10

Exemples pour HAL

NOTE : En cours de construction.

Tous ces exemples impliquent la disponibilité de deux “threads base” et de “servo-thread”.

10.1 Calculer la vitesse

Cet exemple utilise "ddt" pour calculer la vitesse d’un axe.

```
loadrt ddt count=1
loadrt hypot count =1
addf ddt.0 servo-thread
addf hypot.0 servo-thread
net X-vel ddt.0.out => ddt.1.in
```

Chapitre 11

Composants internes

La plupart des composants ont leurs pages de manuel, style *nix. Pour afficher ces “man pages” pour les composants temps réel, taper dans un terminal “man 9 *nomducomposant*”.

Le présent document se concentre sur les composants les plus complexes qui demandent des figures difficiles à reproduire dans une man page.

11.1 Stepgen

Ce composant fournit un générateur logiciel d’impulsions de pas répondant aux commandes de position ou de vitesse. En mode position, il contient une boucle de position pré-réglée, de sorte que les réglages de PID ne sont pas nécessaires. En mode vitesse, il pilote un moteur à la vitesse commandée, tout en obéissant aux limites de vitesse et d’accélération. C’est un composant uniquement temps réel, dépendant de plusieurs facteurs comme la vitesse du CPU, etc, il est capable de fournir des fréquences de pas maximum comprisent entre 10kHz et 50kHz. La figure 11.1 montre trois schémas fonctionnels, chacun est un simple générateur d’impulsions de pas. Le premier diagramme est pour le type ’0’, (pas et direction). Le second est pour le type ’1’ (up/down, ou pseudo-PWM) et le troisième est pour les types 2 jusqu’à 14 (les différentes séquences de pas). Les deux premiers diagrammes montrent le mode de commande position et le troisième montre le mode vitesse. Le mode de commande et le type de pas, se réglent indépendamment et n’importe quelle combinaison peut être choisie.

11.1.1 L’installer

```
emc2$ halcmd loadrt stepgen step_type=<type-array> [ctrl_type=<ctrl_array>]
```

<type-array> est une série d’entiers décimaux séparés par des virgules. Chaque chiffre provoquera le chargement d’un simple générateur d’impulsions de pas, la valeur de ce chiffre déterminera le type de pas. <ctrl_array> est une série de lettres “**p**” ou “**v**” séparées par des virgules, qui spécifient le mode pas ou le mode vitesse. **ctrl_type** est optionnel, si il est omis, tous les générateurs de pas seront en mode position. Par exemple, la commande :

```
emc2# halcmd loadrt stepgen.o step_type=0,0,2 ctrl_type=p,p,v
```

va installer trois générateurs de pas. Les deux premiers utilisent le type de pas ’0’ (pas et direction) et fonctionnent en mode position. Le dernier utilise le type de pas ’2’ (quadrature) et fonctionne en mode vitesse. La valeur par défaut de <config-array> est “0,0,0” qui va installer trois générateurs de type ’0’ (step/dir). Le nombre maximum de générateurs de pas est de 8 (comme définit par MAX_CHAN dans stepgen.c). Chaque générateur est indépendant, mais tous sont actualisés par la même fonction(s), au même instant. Dans les descriptions qui suivent, <chan> est le nombre de générateurs spécifiques. La numérotation des générateurs commence à 0.

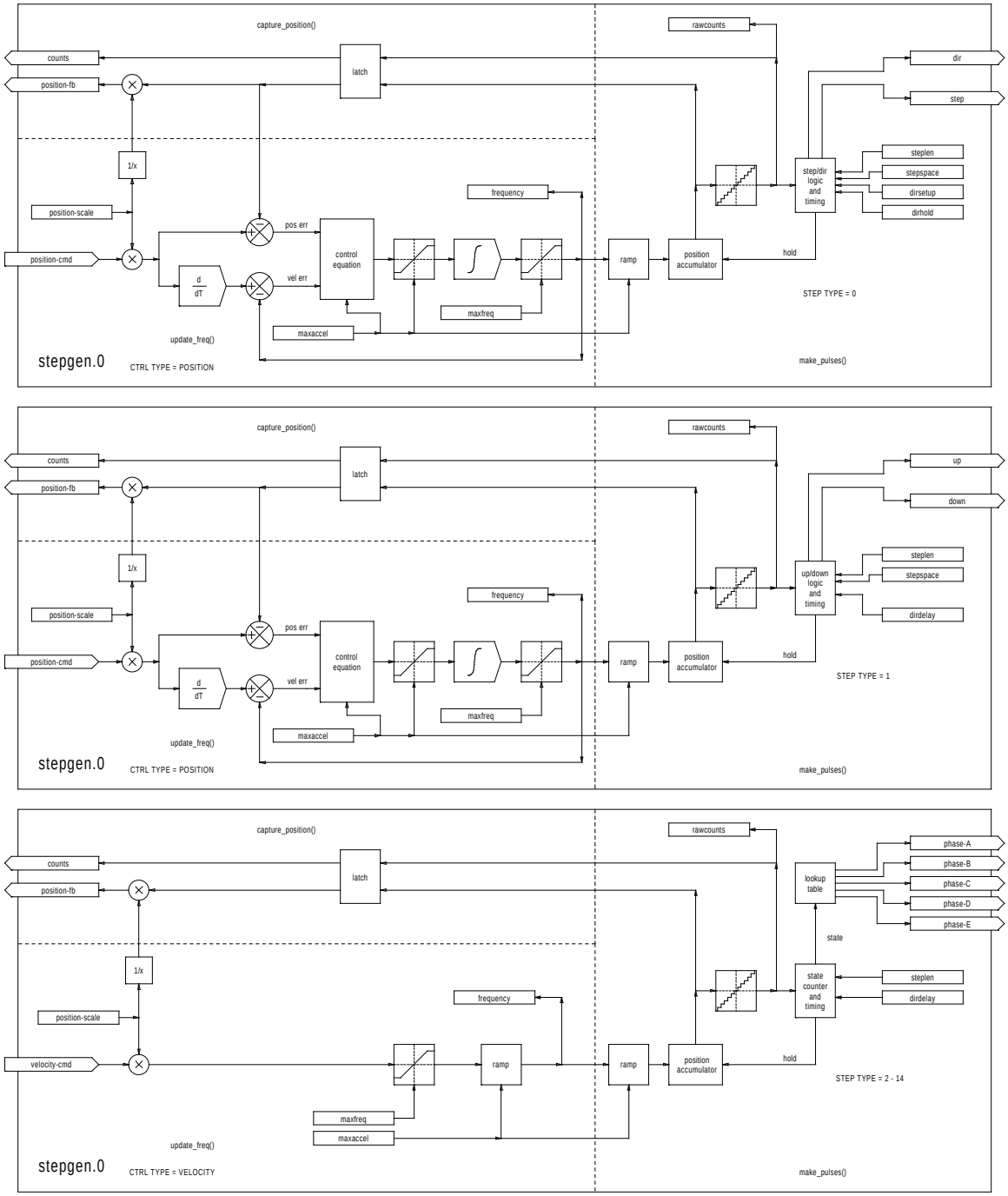


FIG. 11.1 – Diagramme bloc du générateur de pas (en mode position)

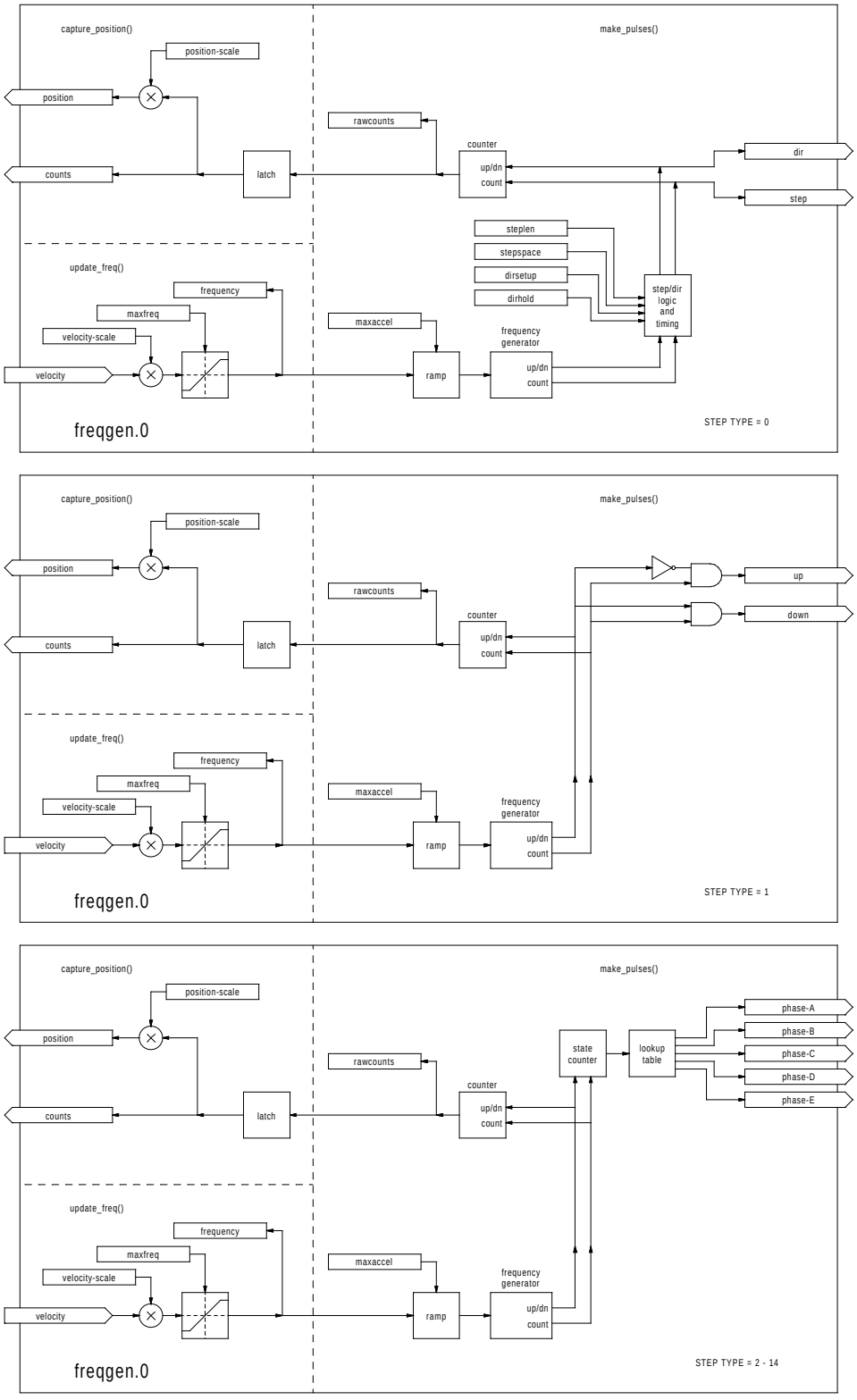


FIG. 11.2 – Diagramme bloc du générateur de pas (en mode vitesse)

11.1.2 Le désinstaller

```
emc2$ halcmd unloadrt stepgen
```

11.1.3 Pins

Chaque générateur d'impulsions de pas n'aura que certaines de ces pins, selon le type de pas et le mode de contrôle sélectionné.

- (FLOAT) stepgen.<chan>.position-cmd – Position désirée du moteur, en unités de longueur (mode position seulement).
- (FLOAT) stepgen.<chan>.velocity-cmd – Vitesse désirée du moteur, en unités de longueur par seconde (mode vitesse seulement).
- (S32) stepgen.<chan>.counts – Rétroaction de la position en unités de comptage, actualisée par la fonction capture_position().
- (FLOAT) stepgen.<chan>.position-fb – Rétroaction de la position en unités de longueur, actualisée par la fonction capture_position().
- (BIT) stepgen.<chan>.step – Sortie des impulsions de pas (type de pas 0 seulement).
- (BIT) stepgen.<chan>.dir – Sortie direction (type de pas 0 seulement).
- (BIT) stepgen.<chan>.up – Sortie UP en pseudo-PWM (type de pas 1 seulement).
- (BIT) stepgen.<chan>.down – Sortie DOWN en pseudo-PWM (type de pas 1 seulement).
- (BIT) stepgen.<chan>.phase-A – Sortie phase A (séquences de pas 2 à 14 seulement).
- (BIT) stepgen.<chan>.phase-B – Sortie phase B (séquences de pas 2 à 14 seulement).
- (BIT) stepgen.<chan>.phase-C – Sortie phase C (séquences de pas 3 à 14 seulement).
- (BIT) stepgen.<chan>.phase-D – Sortie phase D (séquences de pas 5 à 14 seulement).
- (BIT) stepgen.<chan>.phase-E – Sortie phase E (séquences de pas 11 à 14 seulement).

11.1.4 Paramètres

- (FLOAT) stepgen.<chan>.position-scale – Pas par unité de longueur. Ce paramètre est utilisé pour les sorties et les rétroactions.
- (FLOAT) stepgen.<chan>.maxvel – Vitesse maximale, en unités de longueur par seconde. Si égal à 0.0, n'a aucun effet.
- (FLOAT) stepgen.<chan>.maxaccel – Valeur maximale d'accélération, en unités de longueur par seconde au carré. Si égal à 0.0, n'a aucun effet.
- (FLOAT) stepgen.<chan>.frequency – Fréquence des pas, en pas par seconde.
- (FLOAT) stepgen.<chan>.steplen – Durée de l'impulsion de pas (types de pas 0 et 1) ou durée minimum dans un état donné (séquences de pas 2 à 14), en nanosecondes.
- (FLOAT) stepgen.<chan>.stepspace – Espace minimum entre deux impulsions de pas (types de pas 0 et 1 seulement), en nanosecondes.
- (FLOAT) stepgen.<chan>.dirsetup – Durée minimale entre un changement de direction et le début de la prochaine impulsion de pas (type de pas 0 seulement), en nanosecondes.
- (FLOAT) stepgen.<chan>.dirhold – Durée minimale entre la fin d'une impulsion de pas et un changement de direction (type de pas 0 seulement), en nanosecondes.
- (FLOAT) stepgen.<chan>.dirdelay – Durée minimale entre un pas dans une direction et un pas dans la direction opposée (séquences de pas 1 à 14 seulement), en nanosecondes.
- (S32) stepgen.<chan>.rawcounts – Valeur de comptage brute (count) de la rétroaction, réactualisée par la fonction make_pulses().

En mode position, les valeurs de maxvel et de maxaccel sont utilisées par la boucle de position interne pour éviter de générer des trains d'impulsions de pas que le moteur ne peut pas suivre. Lorsqu'elles sont réglées sur des valeurs appropriées pour le moteur, même un grand changement instantané dans la position commandée produira un mouvement trapézoïdal en douceur vers la nouvelle position. L'algorithme fonctionne en mesurant à la fois, l'erreur de position et l'erreur de vitesse, puis en calculant une accélération qui tende à réduire vers zéro, les deux en même temps. Pour plus de détails, y compris les contenus de la boîte "d'équation de contrôle", consulter le code source.

En mode vitesse, maxvel est une simple limite qui est appliquée à la vitesse commandée, maxaccel est utilisé pour créer une rampe avec la fréquence actuelle, si la vitesse commandée change brutalement. Comme dans le mode position, des valeurs appropriées de ces paramètres assurent que le moteur pourra suivre le train d'impulsions généré.

11.1.5 Séquences de pas

Le générateur de pas supporte 15 différentes “séquences de pas”. Le type de pas 0 est le plus familier, c’est le standard pas et direction (step/dir). Quand stepgen est configuré pour le type 0, il y a quatre paramètres supplémentaires qui déterminent le timing exact des signaux de pas et de direction. Voir la figure 11.3 pour la signification de ces paramètres. Les paramètres sont en nanosecondes, mais ils doivent être arrondis à un entier, multiple de la période du thread qui appelle `make_pulses()`. Par exemple, si `make_pulses()` est appelée toutes les 16µs et que “steplen” est à 20000, alors l’impulsion de pas aura une durée de 2 x 16 = 32µs. La valeur par défaut de ces quatre paramètres est de 1ns, mais l’arrondi automatique prendra effet au premier lancement du code. Puisqu’un pas exige d’être haut pendant “`steplen`”ns et bas pendant “`stepspace`”ns, la fréquence maximale est 1.000.000.000 divisé par `(steplen+stepspace)`. Si “`maxfreq`” est réglé plus haut que cette limite, il sera abaissé automatiquement. Si “maxfreq” est à zéro, il restera à zéro, mais la fréquence de sortie sera toujours limitée.

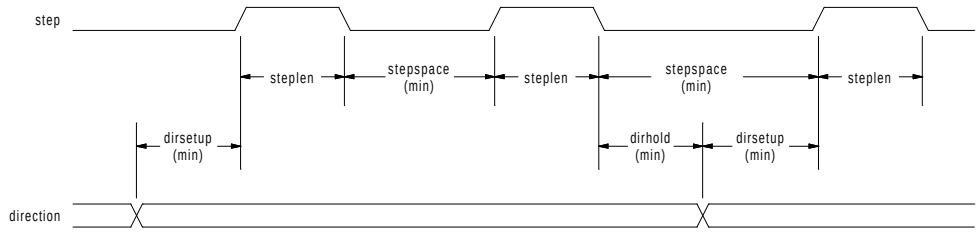


FIG. 11.3 – Timing pas et direction

Le type de pas 1 a deux sorties, up et down. Les impulsions apparaissent sur l’une ou l’autre, selon la direction du déplacement. Chaque impulsion a une durée de “`steplen`”ns et les impulsions sont séparées de “`stepspace`”ns. La fréquence maximale est la même que pour le type 0. Si “`maxfreq`” est réglé plus haut que cette limite il sera abaissé automatiquement. Si “maxfreq” est à zéro, il restera à zéro, mais la fréquence de sortie sera toujours limitée.

Les séquences 2 jusqu’à 14 sont basées sur les états et ont entre deux et cinq sorties. Pour chaque pas, un compteur d”état est incrémenté ou décrémenté. Les figures 11.4, 11.5 et 11.6 montrent les différentes séquences des sorties en fonction de l’état du compteur. La fréquence maximale est 1.000.000.000 divisé par “`steplen`” et comme dans les autres séquences, “`maxfreq`” sera abaissé si il est au dessus de cette limite.

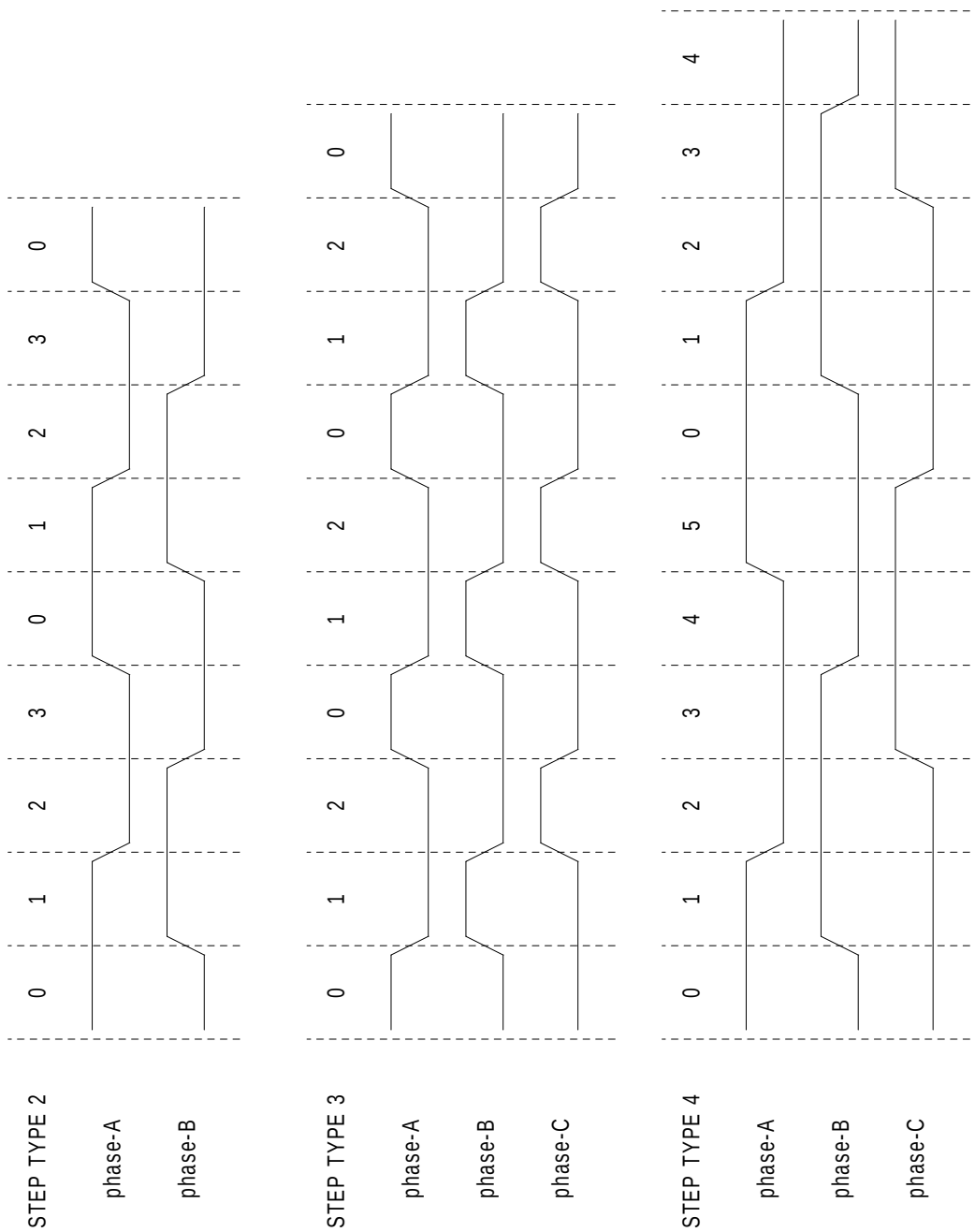


FIG. 11.4 – Séquences de pas à trois phases

11.1.6 Fonctions

Le composant exporte trois fonctions. Chaque fonction agit sur tous les générateurs d'impulsions de pas. Lancer différents générateurs dans différents threads n'est pas supporté.

– `(FUNCT) stepgen.make-pulses` – Fonction haute vitesse de génération et de comptage des impulsions (non flottant).
– `(FUNCT) stepgen.update-freq` – Fonction basse vitesse de conversion de position en vitesse,

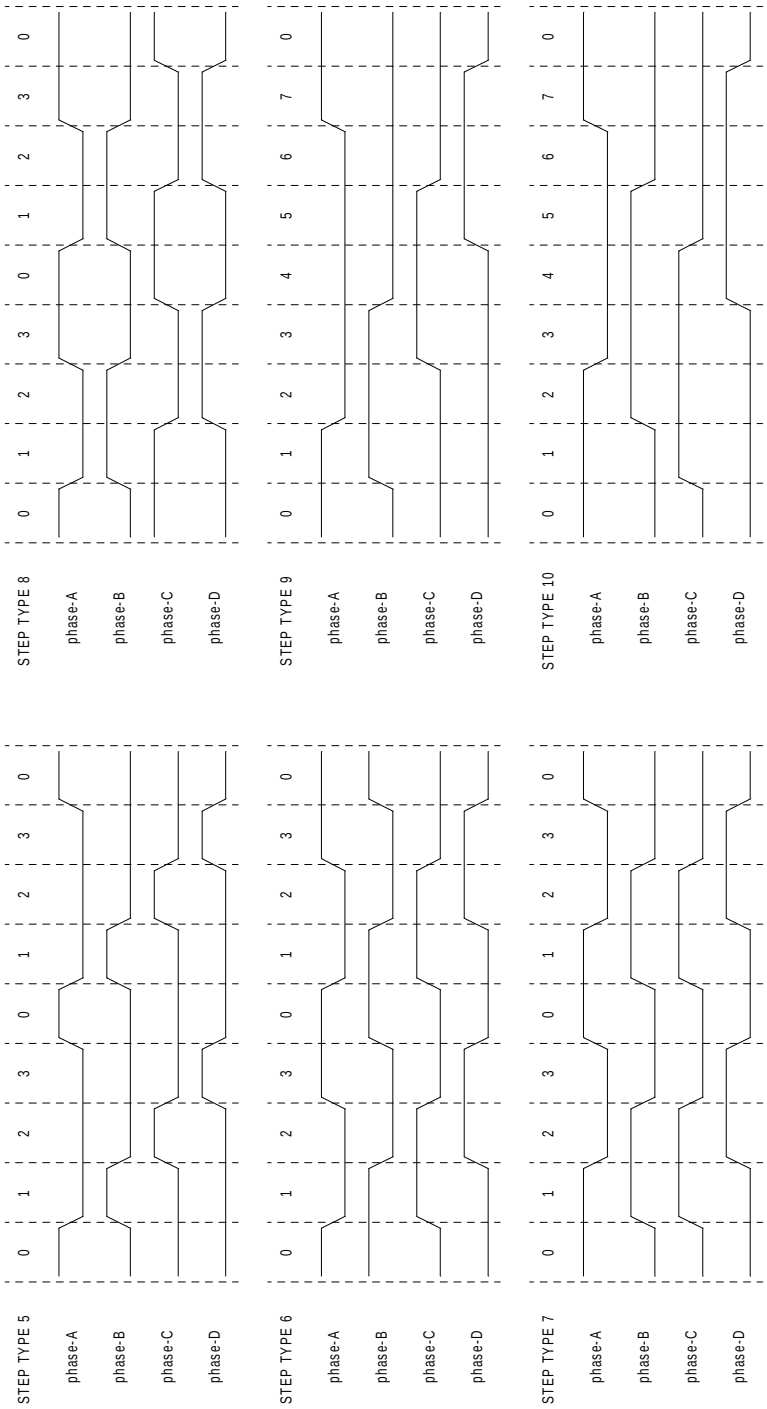


FIG. 11.5 – Séquences de pas à quatre phases

mise à l'échelle et traitement des limitations.

– `(FUNCT) stepgen.capture-position` – Fonction basse vitesse pour la rétroaction, met à jour les latches et les mesures de position.

La fonction à grande vitesse "`stepgen.make-pulses`" devrait être lancée dans un thread très rapide, entre 10 et 50us selon les capacités de l'ordinateur. C'est la période de ce thread qui détermine la fréquence maximale des pas, de `steplen`, `stepspace`, `dirsetup`, `dirhold` et `dirdelay`, `tous sont arrondis` au multiple entier de la période du thread en nanosecondes. Les deux autres fonc-

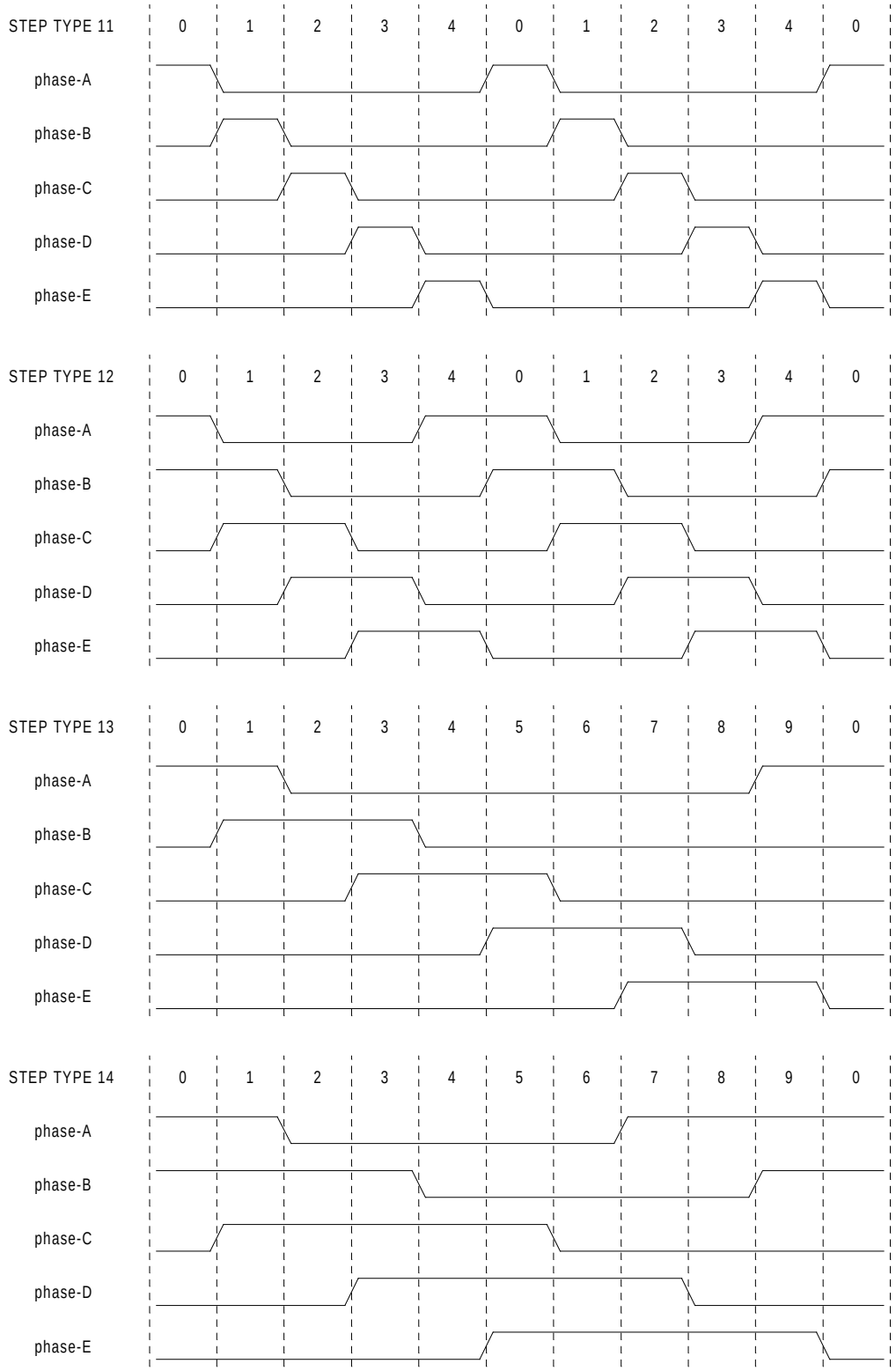


FIG. 11.6 – Séquence de pas à cinq phases

tions peuvent être appelées beaucoup plus lentement.

11.2 PWMgen

Ce composant fournit un générateur logiciel de PWM (modulation de largeur d'impulsions) et PDM (modulation de densité d'impulsions). C'est un composant temps réel uniquement, dépendant de plusieurs facteurs comme la vitesse du CPU, etc, Il est capable de générer des fréquences PWM de quelques centaines de Hertz en assez bonne résolution, à peut-être 10kHz avec une résolution limitée.

11.2.1 L'installer

```
emc2$ halcmd loadrt pwmgen output_type=<config-array>
```

<config-array> est une série d'entiers décimaux séparés par des virgules. Chaque chiffre provoquera le chargement d'un simple générateur de PWM, la valeur de ce chiffre determinera le type de sortie. Par exemple :

```
emc2$ halcmd loadrt pwmgen step_type=0,1,2
```

va installer trois générateurs de PWM. Le premier utilisera une sortie de type '0' (PWM seule), le suivant utilisera une sortie de type 1 (PWM et direction) et le troisième utilisera une sortie de type 2 (UP et DOWN). Il n'y a pas de valeur par défaut, si <config-array> n'est pas spécifié, aucun générateur de PWM ne sera installé. Le nombre maximum de générateurs de fréquences est de 8 (comme définit par MAX_CHAN dans pwmgen.c). Chaque générateur est indépendant, mais tous sont mis à jour par la même fonction(s), au même instant. Dans les descriptions qui suivent, <chan> est le nombre de générateurs spécifiques. La numérotation des générateurs de PWM commence à 0.

11.2.2 Le désinstaller

```
emc2$ halcmd unloadrt pwmgen
```

11.2.3 Pins

Chaque générateur de PWM aura les pins suivantes :

- (FLOAT) pwmgen.<chan>.value – Valeur commandée, en unités arbitraires. Sera mise à l'échelle par le paramètre d'échelle (voir ci-dessous).
- (BIT) pwmgen.<chan>.enable – Active ou désactive les sorties du générateur de PWM.

Chaque générateur de PWM aura également certaines de ces pins, selon le type de sortie choisi :

- (BIT) pwmgen.<chan>.pwm – Sortie PWM (ou PDM), (types de sortie 0 et 1 seulement).
- (BIT) pwmgen.<chan>.dir – Sortie direction (type de sortie 1 seulement).
- (BIT) pwmgen.<chan>.up – Sortie PWM/PDM pour une valeur positive en entrée (type de sortie 2 seulement).
- (BIT) pwmgen.<chan>.down – Sortie PWM/PDM pour une valeur négative en entrée (type de sortie 2 seulement).

11.2.4 Paramètres

- (FLOAT) pwmgen.<chan>.scale – Facteur d'échelle pour convertir les valeurs en unités arbitraires, en coefficients de facteur cyclique.
- (FLOAT) pwmgen.<chan>.pwm-freq – Fréquence de PWM désirée, en Hz. Si égale à 0.0, la modulation sera PDM au lieu de PWM. Si elle est réglée plus haute que les limites internes, au prochain appel de la fonction update_freq() elle sera ramenée aux limites internes. Si elle est différente de zéro et si le lissage est faux, au prochain appel de la fonction update_freq() elle sera réglée au plus proche entier multiple de la période de la fonction make_pulses().

– (BIT) pwmgen.<chan>.dither-pwm – Si vrai, active le lissage pour affiner la fréquence PWM ou le rapport cyclique qui ne pourraient pas être obtenus avec une pure PWM. Si faux, la fréquence PWM et le rapport cyclique seront tous les deux arrondis aux valeurs pouvant être atteintes exactement.
– (FLOAT) pwmgen.<chan>.min-dc – Rapport cyclique minimum compris entre 0.0 et 1.0 (Le rapport cyclique sera à zéro quand il est désactivé, indépendamment de ce paramètre).
– (FLOAT) pwmgen.<chan>.max-dc – Rapport cyclique maximum compris entre 0.0 et 1.0.
– (FLOAT) pwmgen.<chan>.curr-dc – Rapport cyclique courant, après toutes les limitations et les arrondis (lecture seule).

11.2.5 Types de sortie

Le générateur de PWM supporte trois “types de sortie”. Le type 0 a une seule pin de sortie. Seules, les commandes positives sont acceptées, les valeurs négatives sont traitées comme zéro (elle seront affectées par min-dc si il est différent de zéro). Le type 1 a deux pins de sortie, une pour le signal PWM/PDM et une pour indiquer la direction. Le rapport cyclique d'une pin PWM est basé sur la valeur absolue de la commande, de sorte que les valeurs négatives sont acceptables. La pin de direction est fausse pour les commandes positives et vraie pour les commandes négatives. Finalement, le type 2 a également deux sorties, appelées “up” et “down”. Pour les commandes positives, le signal PWM apparaît sur la sortie up et la sortie down reste fausse. Pour les commandes négatives, le signal PWM apparaît sur la sortie down et la sortie up reste fausse. Les sorties de type 2 sont appropriées pour piloter la plupart des ponts en H.

11.2.6 Fonctions

Le composant exporte deux fonctions. Chaque fonction agit sur tous les générateurs de PWM, lancer différents générateurs dans différents threads n'est pas supporté.

– (FUNCT) pwmgen.make-pulses – Fonction haute vitesse de génération de fréquences PWM (non flottant).
– (FUNCT) pwmgen.update – Fonction basse vitesse de mise à l'échelle, limitation des valeurs et traitement d'autres paramètres.

La fonction haute vitesse “pwmgen.make-pulses” devrait être lancée dans un thread très rapide, entre 10 et 50µs selon les capacités de l'ordinateur. C'est la période de ce thread qui détermine la fréquence maximale de la porteuse PWM, ainsi que la résolution des signaux PWM ou PDM. L'autre fonction peut être appelée beaucoup plus lentement.

11.3 Codeur

Ce composant fournit, en logiciel, le comptage des signaux provenant d'encodeurs en quadrature. Il s'agit d'un composant temps réel uniquement, il est dépendant de divers facteurs comme la vitesse du CPU, etc, il est capable de compter des signaux de fréquences comprises entre 10kHz à peut être 50kHz. La figure 11.7 est le diagramme bloc d'un canal de comptage de codeur.

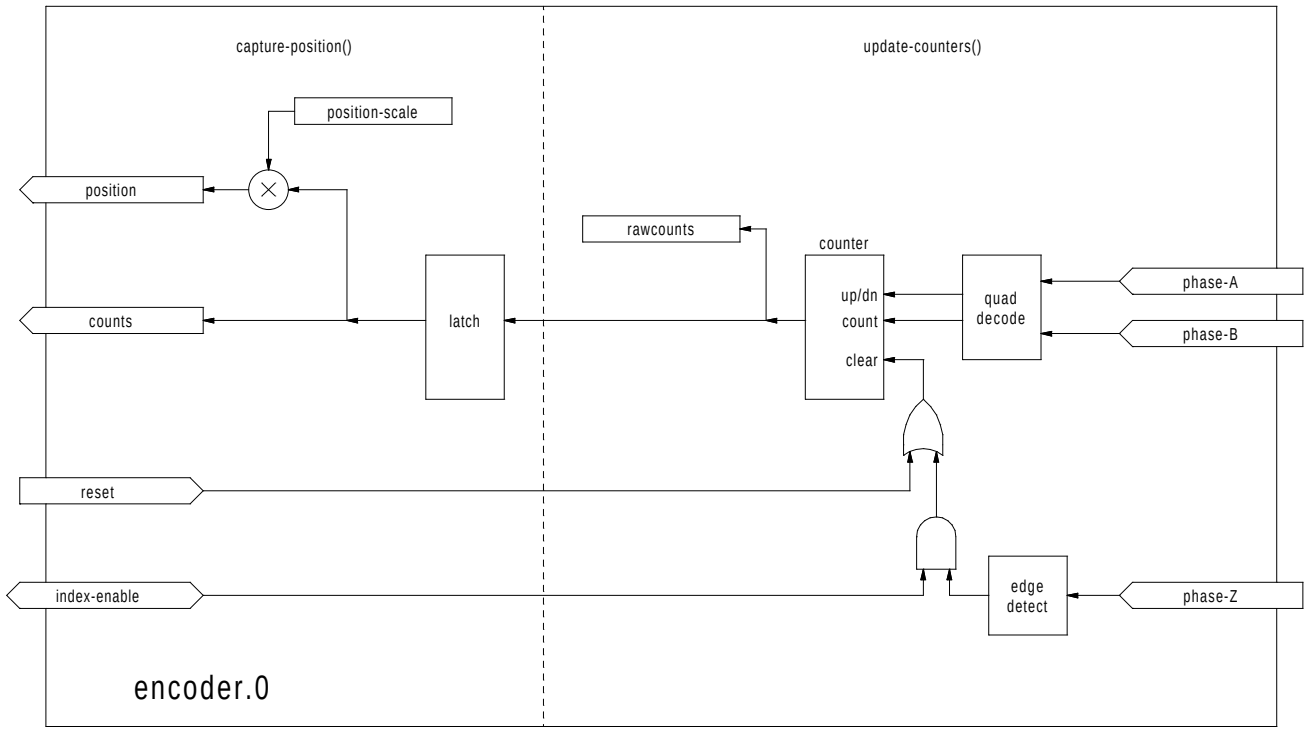


FIG. 11.7 – Diagramme bloc du compteur de codeur

11.3.1 L'installer

```
emc2$ halcmd loadrt encoder [num_chan=<counters>]
```

`<counters>` est le nombre de compteurs de codeur à installer. Si `numchan` n'est pas spécifié, trois compteurs seront installés. Le nombre maximum de compteurs est de 8 (comme définit par MAX_CHAN dans encoder.c). Chaque compteur est indépendant, mais tous sont mis à jour par la même fonction(s) au même instant. Dans les descriptions qui suivent, `<chan>` est le nombre de compteurs spécifiques. La numérotation des compteurs commence à 0.

11.3.2 Le désinstaller

```
emc2$ halcmd unloadrt encoder
```

11.3.3 Pins

- (BIT) encoder.<chan>.phase-A – Signal de la phase A du codeur en quadrature.
- (BIT) encoder.<chan>.phase-B – Signal de la phase B du codeur en quadrature.
- (BIT) encoder.<chan>.phase-Z – Signal de la phase Z (impulsion d'ndex) du codeur en quadrature.
- (BIT) encoder.<chan>.reset – Voir l'interface canonique des codeurs à la section **??**.
- (BIT) encoder.<chan>.velocity – Vitesse estimée du signal de quadrature.
- (BIT) encoder.<chan>.index-enable – Voir l'interface canonique des codeurs.
- (S32) encoder.<chan>.count – Voir l'interface canonique des codeurs.
- (FLOAT) encoder.<chan>.position – Voir l'interface canonique des codeurs.

11.3.4 Paramètres

- (S32) encoder.<chan>.raw-count – Valeur de comptage brute, actualisée par la fonction update-counters
- (BIT) encoder.<chan>.x4-mode – Ajuste le comptage en mode 4x ou 1x. Le mode 1x est intéressant pour les manivelles de jog.
- (FLOAT) encoder.<chan>.position-scale – Voir l'interface canonique des codeurs à la section **??**.

11.3.5 Fonctions

Le composant exporte deux fonctions. Chaque fonction agit sur tous les compteurs de codeur, lancer différents compteurs de codeur dans différents threads n'est pas supporté.

- (FUNCT) encoder.update-counters – Fonction haute vitesse de comptage d'impulsions (non flottant).
- (FUNCT) encoder.capture-position – Fonction basse vitesse d'actualisation des latches et mise à l'échelle de la position.

11.4 PID

Ce composant fournit une boucle de contrôle Proportionnel/Intégrale/Dérivée. C'est un composant temps réel uniquement. Par souci de simplicité, cette discussion suppose que nous parlons de boucles de position, mais ce composant peut aussi être utilisé pour implémenter d'autres boucles de rétroaction telles que vitesse, hauteur de torche, température, etc. La figure 11.8 est le schéma fonctionnel d'une simple boucle PID.

11.4.1 L'installer

```
emc2$ halcmd loadrt pid [num_chan=<loops>] [debug=1]
```

`<loops>` est le nombre de boucles PID à installer. Si `numchan` n'est pas spécifié, une seule boucle sera installée. Le nombre maximum de boucles est de 16 (comme définit par MAX_CHAN dans pid.c). Chaque boucle est complétement indépendante. Dans les descriptions qui suivent, `<loopnum>` est le nombre de boucles spécifiques. La numérotation des boucle PID commence à 0.

Si `debug=1` est spécifié, le composant exporte quelques paramètres destinés au déboguage et aux réglages. Par défaut, ces paramètres ne sont pas exportés, pour économiser la mémoire partagée et éviter d'encombrer la liste des paramètres.

11.4.2 Le désinstaller

```
emc2$ halcmd unloadrt pid
```

11.4.3 Pins

Les trois principales pins sont :

- `(FLOAT) pid.<loopnum>.command` – La position désirée (consigne), telle que commandée par un autre composant système.
- `(FLOAT) pid.<loopnum>.feedback` – La position actuelle (mesure), telle que mesurée par un organe de rétroaction comme un codeur de position.
- `(FLOAT) pid.<loopnum>.output` – Une commande de vitesse qui tend à aller de la position actuelle à la position désirée.

Pour une boucle de position, 'command' et 'feedback' sont en unités de longueur. Pour un axe linéaire, cela pourrait être des pouces, mm, mètres, ou tout autre unité pertinente. De même pour un axe angulaire, ils pourraient être des degrés, radians, etc. Les unités sur la pin 'output' représentent l'écart nécessaire pour que la rétroaction coïncide avec la commande. Pour une boucle de position, 'output' est une vitesse exprimée en pouces/seconde, mm/seconde, degrés/seconde, etc. Les unités de temps sont toujours des secondes et les unités de vitesses restent cohérentes avec les unités de longueur. Si la commande et la rétroaction sont en mètres, la sortie sera en mètres par seconde.

Chaque boucle PID a deux autres pins qui sont utilisées pour surveiller ou contrôler le fonctionnement général du composant.

- `(FLOAT) pid.<loopnum>.error` – Egal à `.command` moins `.feedback`. (consigne - mesure)
- `(BIT) pid.<loopnum>.enable` – Un bit qui active la boucle. Si `.enable` est faux, tous les intégrateurs sont remis à zéro et les sorties sont forcées à zéro. Si `.enable` est vrai, la boucle opère normalement.

11.4.4 Paramètres

Le gain PID, les limites et autres caractèristiques 'accordables' de la boucle sont implémentés comme des paramètres.

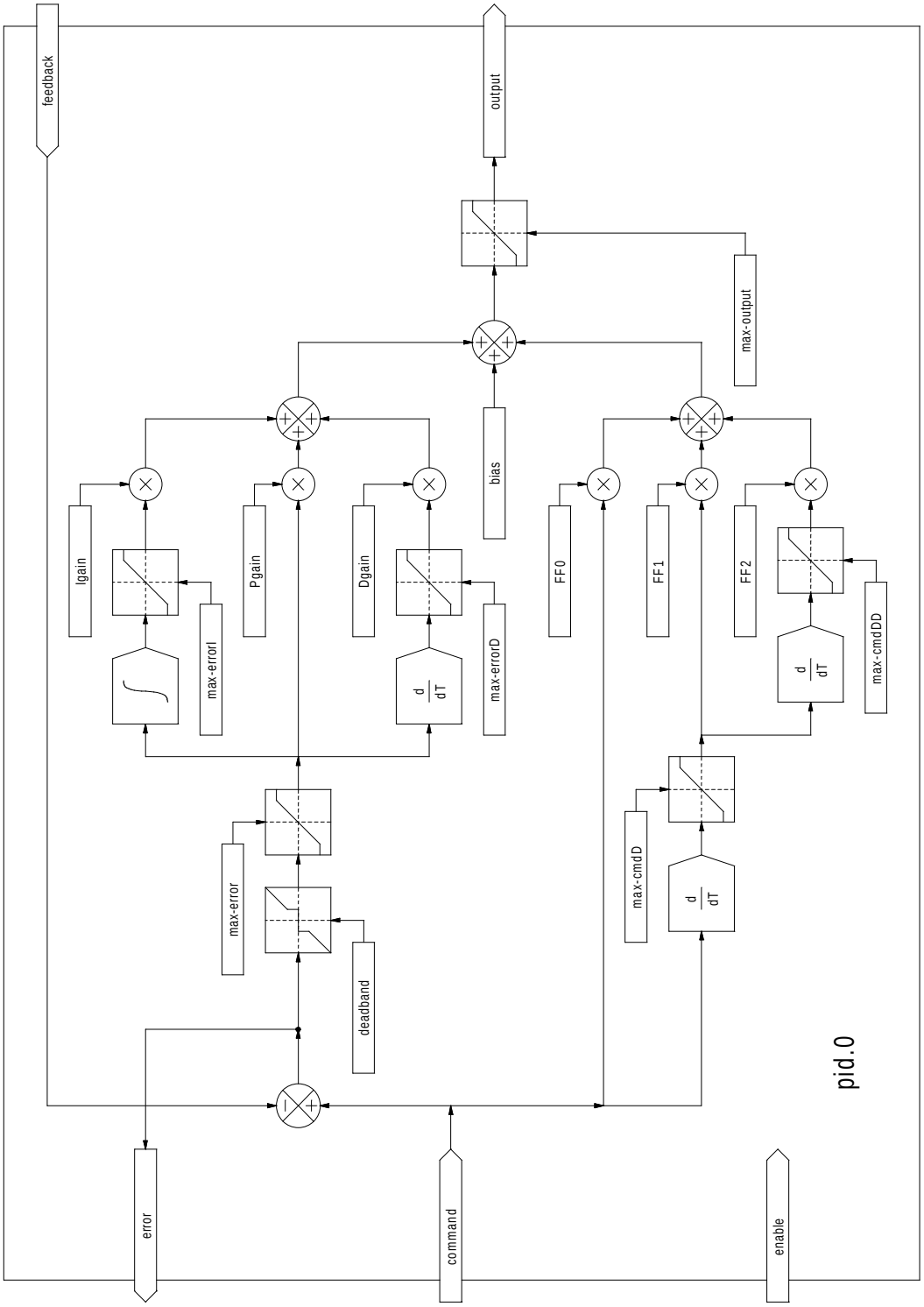


FIG. 11.8 – Diagramme bloc d'une boucle PID

– (FLOAT) pid.<loopnum>.Pgain – Gain de la composante proportionnelle
– (FLOAT) pid.<loopnum>.Igain – Gain de la composante intégrale
– (FLOAT) pid.<loopnum>.Dgain – Gain de la composante dérivée
– (FLOAT) pid.<loopnum>.bias – Constante du décalage de sortie
– (FLOAT) pid.<loopnum>.FF0 – Correcteur prédictif d'ordre zéro (feedforward) - sortie propor-

tionnelle à la commande (position).
- (FLOAT) pid.<loopnum>.FF1 – Correcteur prédictif de premier ordre (feedforward) - sortie proportionnelle à la dérivée de la commande (vitesse).
- (FLOAT) pid.<loopnum>.FF2 – Correcteur prédictif de second ordre (feedforward) - sortie proportionnelle à la dérivée seconde de la commande (accélération)[1].
- (FLOAT) pid.<loopnum>.deadband – Définit la bande morte (tolérance)
- (FLOAT) pid.<loopnum>.maxerror – Limite d'erreur
- (FLOAT) pid.<loopnum>.maxerrorI – Limite d'erreur intégrale
- (FLOAT) pid.<loopnum>.maxerrorD – Limite d'erreur dérivée
- (FLOAT) pid.<loopnum>.maxcmdD – Limite de la commande dérivée
- (FLOAT) pid.<loopnum>.maxcmdDD – Limite de la commande dérivée seconde
- (FLOAT) pid.<loopnum>.maxoutput – Limite de la valeur de sortie

Toutes les limites max???, sont implémentées de sorte que si la valeur de ce paramètre est à zéro, il n'y a pas de limite.

Si debug=1 est spécifié quand le composant est installé, quatre paramètres supplémentaires seront exportés :

- (FLOAT) pid.<loopnum>.errorI – Intégrale de l'erreur.
- (FLOAT) pid.<loopnum>.errorD – Dérivée de l'erreur.
- (FLOAT) pid.<loopnum>.commandD – Dérivée de la commande.
- (FLOAT) pid.<loopnum>.commandDD – Dérivée seconde de la commande.

11.4.5 Fonctions

Le composant exporte une fonction pour chaque boucle PID. Cette fonction exécute tous les calculs nécessaires à la boucle. Puisque chaque boucle a sa propre fonction, les différentes boucles peuvent être incluses dans les différents threads et exécutées à différents rythmes.

- (FUNCT) pid.<loopnum>.do_pid_calcs – Exécute tous les calculs d'une seule boucle PID.

Si vous voulez comprendre exactement l'algorithme utilisé pour calculer la sortie d'une boucle PID, référez vous à la figure 11.8, les commentaires au début du source emc2/src/hal/components/pid.c et bien sûr, au code lui même. Les calculs de boucle sont dans la fonction C calc_pid().

[1] FF2 n'est actuellement pas implémenté, mais il pourrait l'être. Considérez cette note "FIXME" dans le code.

11.5 Codeur simulé

Le codeur simulé est exactement la même chose qu'un codeur. Il produit des impulsions en quadrature avec une impulsion d'index, à une vitesse contrôlée par une pin de HAL. Surtout utile pour les essais.

11.5.1 L'installer

```
emc2$ halcmd loadrt sim-encoder num_chan=<number>
```

<number> est le nombre de codeurs à simuler. Si aucun n'est spécifié, un seul codeur sera installé. Le nombre maximum de codeurs est de 8 (comme définit par MAX_CHAN dans sim_encoder.c).

11.5.2 Le désinstaller

```
emc2$ halcmd unloadrt sim-encoder
```

11.5.3 Pins

- (FLOAT) sim-encoder.<chan-num>.speed – La vitesse commandée pour l'arbre simulé.
- (BIT) sim-encoder.<chan-num>.phase-A – Sortie en quadrature.
- (BIT) sim-encoder.<chan-num>.phase-B – Sortie en quadrature.
- (BIT) sim-encoder.<chan-num>.phase-Z – Sortie de l'impulsion d'index.

Quand .speed est positive, .phase-A mène .phase-B.

11.5.4 Paramètres

- (U32) sim-encoder.<chan-num>.ppr – Impulsions par tour d'arbre.
- (FLOAT) sim-encoder.<chan-num>.scale – Facteur d'échelle pour **speed**. Par défaut est de 1.0, ce qui signifie que **speed** est en tours par seconde. Passer l'échelle à 60 pour des tours par minute, la passer à 360 pour des degrés par seconde, à 6.283185 pour des radians par seconde, etc.

Noter que les impulsions par tour ne sont pas identiques aux valeurs de comptage par tour (counts). Une impulsion est un cycle complet de quadrature. La plupart des codeurs comptent quatre fois pendant un cycle complet.

11.5.5 Fonctions

Le composant exporte deux fonctions. Chaque fonction affecte tous les codeurs simulés.

- (FUNCT) sim-encoder.make-pulses – Fonction haute vitesse de génération d'impulsions en quadrature (non flottant).
- (FUNCT) sim-encoder.update-speed – Fonction basse vitesse de lecture de **speed**, de mise à l'échelle et d'activation de **make-pulses**.

11.6 Anti-rebond

L'anti-rebond est un composant temps réel capable de filtrer les rebonds créés par les contacts mécaniques. Il est également très utile dans d'autres applications, où des impulsions très courtes doivent être supprimées.

11.6.1 L'installer

```
emc2$ halcmd loadrt debounce cfg="<config-string>"
```

<config-string> est une série d'entiers décimaux séparés par des espaces. Chaque chiffre installe un groupe de filtres anti-rebond identiques, le chiffre détermine le nombre de filtres dans le groupe. Par exemple :

```
emc2$ halcmd loadrt debounce cfg="1 4 2"
```

va installer trois groupes de filtres. Le groupe 0 contient un filtre, le groupe 1 en contient quatre et le groupe 2 en contient deux. La valeur par défaut de <config-string> est "1" qui installe un seul groupe contenant un seul filtre. Le nombre maximum de groupes est de 8 (comme définit par MAX_GROUPS dans debounce.c). Le nombre maximum de filtres dans un groupe est limité seulement par l'espace de la mémoire partagée. Chaque groupe est complétement indépendant. Tous les filtres dans un même groupe sont identiques et ils sont tous mis à jour par la même fonction, au même instant. Dans les descriptions qui suivent, <G> est le numéro du groupe et <F> est le numéro du filtre dans le groupe. Le premier filtre est le filtre 0 dans le groupe 0.

11.6.2 Le désinstaller

```
emc2$ halcmd unloadrt debounce
```

11.6.3 Pins

Chaque filtre individuel a deux pins.

– (BIT) debounce.<G>.<F>.in – Entrée du filtre <F> du groupe <G>.
– (BIT) debounce.<G>.<F>.out – Sortie du filtre <F> du groupe <G>.

11.6.4 Paramètres

Chaque groupe de filtre a un paramètre[2].

– (s32) debounce.<G>.delay – Délai de filtrage pour tous les filtres du groupe <G>.

Le délai du filtre est dans l'unité de la période du thread. Le délai minimum est de zéro. La sortie d'un filtre avec un délai de zéro, suit exactement son entrée, il ne filtre rien. Plus le délai augmente, plus larges seront les impulsions rejetées. Si le délai est de 4, toutes les impulsions égales ou inférieures à quatre périodes du thread, seront rejetées.

11.6.5 Fonctions

Chaque groupe de filtres exporte une fonction qui met à jour tous les filtres de ce groupe "simultanément". Différents groupes de filtres peuvent être mis à jour dans différents threads et à différentes périodes.

– (FUNCT) debounce.<G> – Met à jour tous les filtres du groupe <G>.

[2]Chaque filtre individuel a également une variable d'état interne. C'est un switch du compilateur qui peut exporter cette variable comme un paramètre. Ceci est prévu pour des essais et devrait juste être un gaspillage de mémoire partagée dans des circonstances normales.

11.7 Siggen

Siggen est un composant temps réel qui génère des signaux carrés, triangulaires et sinusoïdaux. Il est principalement utilisé pour les essais.

11.7.1 L'installer

```
emc2$ halcmd loadrt siggen [num_chan=<chans>]
```

<chans> est le nombre de générateurs de signaux à installer. Si numchan n'est pas spécifié, un seul générateur de signaux sera installé. Le nombre maximum de générateurs est de 16 (comme définit par MAX_CHAN dans siggen.c). Chaque générateur est complétement indépendant. Dans les descriptions qui suivent, <chan> est le numéro d'un générateur spécifique. Les numéros de générateur commencent à 0.

11.7.2 Le désinstaller

```
emc2$ halcmd unloadrt siggen
```

11.7.3 Pins

Chaque générateur a cinq pins de sortie.

- (FLOAT) siggen.<chan>.sine – Sortie de l'onde sinusoïdale.
- (FLOAT) siggen.<chan>.cosine – Sortie de l'onde cosinusoïdale.
- (FLOAT) siggen.<chan>.sawtooth – Sortie de l'onde en dents de scie.
- (FLOAT) siggen.<chan>.triangle – Sortie de l'onde triangulaire.
- (FLOAT) siggen.<chan>.square – Sortie de l'onde carrée.

Les cinq sorties ont les mêmes fréquence, amplitude et offset.

Trois pins de contrôle s'ajoutent aux pins de sortie :

- (FLOAT) siggen.<chan>.frequency – Réglage de la fréquence en Hertz, par défaut la valeur est de 1 Hz.
- (FLOAT) siggen.<chan>.amplitude – Réglage de l'amplitude de pic des signaux de sortie, par défaut, est à 1.
- (FLOAT) siggen.<chan>.offset – Réglage de la composante continue des signaux de sortie, par défaut, est à 0.

Par exemple, si siggen.0.amplitude est à 1.0 et siggen.0.offset est à 0.0, les sorties oscilleront entre -1.0 et +1.0. Si siggen.0.amplitude est à 2.5 et siggen.0.offset est à 10.0, les sorties oscilleront entre 7.5 et 12.5.

11.7.4 Paramètres

Aucun. [3]

11.7.5 Fonctions

- (FUNCT) siggen.<chan>.update – Calcule les nouvelles valeurs pour les cinq sorties.

[3] Dans les versions antérieures à la 2.1, fréquence, amplitude et offset étaient des paramètres. Ils ont été modifiés en pins pour permettre le contrôle par d'autres composants.

Chapitre 12

Pilotes de périphériques

12.1 Parport

Parport est un pilote pour le port parallèle traditionnel des PC. Le port dispose d'un total de 17 broches physiques. Le port parallèle originel a divisé ces broches en trois groupes : données, contrôles et états. Le groupe "données" consiste en 8 broches de sortie, le groupe "contrôles" consiste en 4 broches et le groupe "états" consiste en 5 broches d'entrée.

Au début des années 1990, le port parallèle bidirectionnel est arrivé, ce qui a permis à l'utilisateur d'ajuster le groupe des données comme étant des sorties ou comme étant des entrées. Le pilote de HAL supporte le port bidirectionnel et permet à l'utilisateur de configurer le groupe des données en entrées ou en sorties. Si il est configuré en sorties, un port fournit un total de 12 sorties et 5 entrées. Si il est configuré en entrées, il fournit 4 sorties et 13 entrées.

Dans certains ports parallèles, les broches du groupe contrôle sont des collecteurs ouverts, ils peuvent aussi être mis à l'état bas par une porte extérieure. Sur une carte avec les broches de contrôle en collecteurs ouverts, le mode "x" de HAL permet un usage plus flexible avec 8 sorties dédiées, 5 entrées dédiées et 4 broches en collecteurs ouverts. Dans d'autres ports parallèles, les broches du groupe contrôles sont en push-pull et ne peuvent pas être utilisées comme des entrées.[1]

Aucune autre combinaison n'est supportée. Un port ne peut plus être modifié pour passer d'entrées en sorties dès que le pilote est installé. La figure 12.1 montre deux diagrammes, un montre le pilote quand le groupe de données est configuré en sorties et le second le montre configuré en entrées.

Le pilote parport peut contrôler au maximum 8 ports (définis par MAX_PORTS dans le fichier hal_parport.c). Les ports sont numérotés à partir de zéro.

12.1.1 Installation

```
loadrt hal_parport cfg="<config-string>"
```

[1] HAL ne peut pas déterminer automatiquement si les broches en mode bidirectionnel "x" sont effectivement en collecteurs ouverts (OC). Si elles n'y sont pas, elles ne peuvent pas être utilisées comme des entrées. Essayer de les passer à l'état BAS par une source extérieure peut détériorer le matériel.

Pour déterminer si votre port a des broches de contrôle en "collecteur ouvert", charger hal_parport en mode "x", positionner les broches de contrôle à une valeur HAUTE. HAL doit lire des pins à l'état VRAI. Ensuite, insérer une résistance de 470Ω entre une des broches de contrôle et GND du port parallèle. Si la tension de cette broche de contrôle est maintenant proche de 0V et que HAL la lit comme une pin FAUSSE, alors vous avez un port OC. Si la tension résultante est loin de 0V ou que HAL ne la lit pas comme étant FAUSSE, votre port ne ne peut pas être utilisé en mode "x".

Le matériel extérieur qui pilote les broches de contrôle devrait également utiliser des portes en collecteur ouvert (ex : 74LS05...). Généralement, une pin de HAL -out devrait être VRAIE quand la pin physique est utilisée comme une entrée.

Sur certaines machines, les paramètres du BIOS peuvent affecter la possibilité d'utiliser le mode "x". Le mode "SPP" est le mode qui fonctionne le plus fréquemment.

La chaine config-string représente l'adresse hexadécimale du port, suivie optionnellement par une direction, le tout répété pour chaque port. Les directions sont "in", "out", ou "x", elles déterminent la direction des broches physiques 2 à 9 et s'il y a lieu de créer des pins d'entrée de HAL pour les broches de contrôle physiques. Si la direction n'est pas précisée, le groupe données sera par défault en sortie. Par exemple :

```
loadrt hal_parport cfg="0x278 0x378 in 0x20A0 out"
```

Cet exemple installe les pilotes pour un port 0x0278, avec les broches 2 à 9 en sorties (par défaut, puisque ni " in ", ni "out" n'est spécifié), un port 0x0378, avec les broches 2 à 9 en entrées et un port 0x20A0, avec les broches 2 à 9 explicitement spécifiées en sorties. Notez que vous devez connaître l'adresse de base des ports parallèles pour configurer correctement les pilotes. Pour les ports sur bus ISA, ce n'est généralement pas un problème, étant donné que les ports sont presque toujours à une adresse "bien connue", comme 0x278 ou 0x378 qui sont typiquement configurées dans le BIOS. Les adresses des cartes sur bus PCI sont habituellement trouvées avec "lspci -v" dans une ligne "I/O ports", ou dans un message du kernel après l'exécution de "`sudo modprobe -a parport_pc`". Il n'y a pas d'adresse par défaut, si `<config-string>` ne contient pas au moins une adresse, un message d'erreur s'affichera.

12.1.2 Pins

- `(BIT) parport.<portnum>.pin-<pinnum>-out` – Pilote une pin de sortie physique.
- `(BIT) parport.<portnum>.pin-<pinnum>-in` – Suit une pin d'entrée physique.
- `(BIT) parport.<portnum>.pin-<pinnum>-in-not` – Suit une pin d'entrée physique, mais inversée.

Pour chaque pin, `<portnum>` est le numéro du port et `<pinnum>` est le numéro de la broche physique du connecteur DB-25.

Pour chaque broche de sortie physique, le pilote crée une simple pin de HAL, par exemple `parport.0.pin-14-ou`
Les pins 2 jusqu'à 9 font partie du groupe données, elles sont des pins de sortie si le port est défini comme un port de sorties (par défaut en sortie). Les broches 1, 14, 16 et 17 sont des sorties dans tous les modes. Ces pins de HAL contrôlent l'état des pins physiques correspondantes.

Pour chaque pin d'entrée physique, le pilote crée deux pins de HAL, par exemple `parport.0.pin-12-in` et `parport.0.pin-12-in-not`. Les pins 10, 11, 12, 13 et 15 sont toujours des sorties. Les pins 2 jusqu'à 9 sont des pins d'entrée seulement si le port est défini comme un port d'entrées. Une pin de HAL `-in` est VRAIE si la pin physique est haute et FAUSSE si la pin physique est basse. Une pin de HAL `-in-not` est inversée, elle est FAUSSE si la pin physique est haute. En connectant un signal à l'une ou à l'autre, l'utilisateur peut décider de la logique de l'entrée. En mode "x", les pins 1, 14, 16 et 17 sont également des pins d'entrée.

12.1.3 Paramètres

- `(BIT) parport.<portnum>.pin-<pinnum>-out-invert` – Inverse une pin de sortie.
- `(BIT) parport.<portnum>.pin-<pinnum>-out-reset` (seulement les pins 2..9) – VRAIE si cette pin doit être réinitialisée quand la fonction de réinitialisation est exécutée.
- `(U32) parport.<portnum>.reset-time` – Le temps (en nanosecondes) entre le moment ou la broche est écrite et le moment ou elle est réititialisée par les fonctions de réinitialisation de HAL.

Le paramètre `-invert` détermine si une pin de sortie est active haute ou active basse. Si `-invert` est FAUX, mettre la pin HAL `-out` VRAIE placera la pin physique à l'état haut et mettre la pin HAL FAUSSE placera la pin physique à l'état bas. Si `-invert` est VRAI, mettre la pin HAL `-out` VRAIE va mettre la pin physique à l'état bas.

Si `-reset` est VRAI, la fonction de réinitialisation va passer la pin à la valeur de `-out-invert`. Ceci peut être utilisé en conjonction avec "stepgen `doublefreq`" pour produire un pas par période.

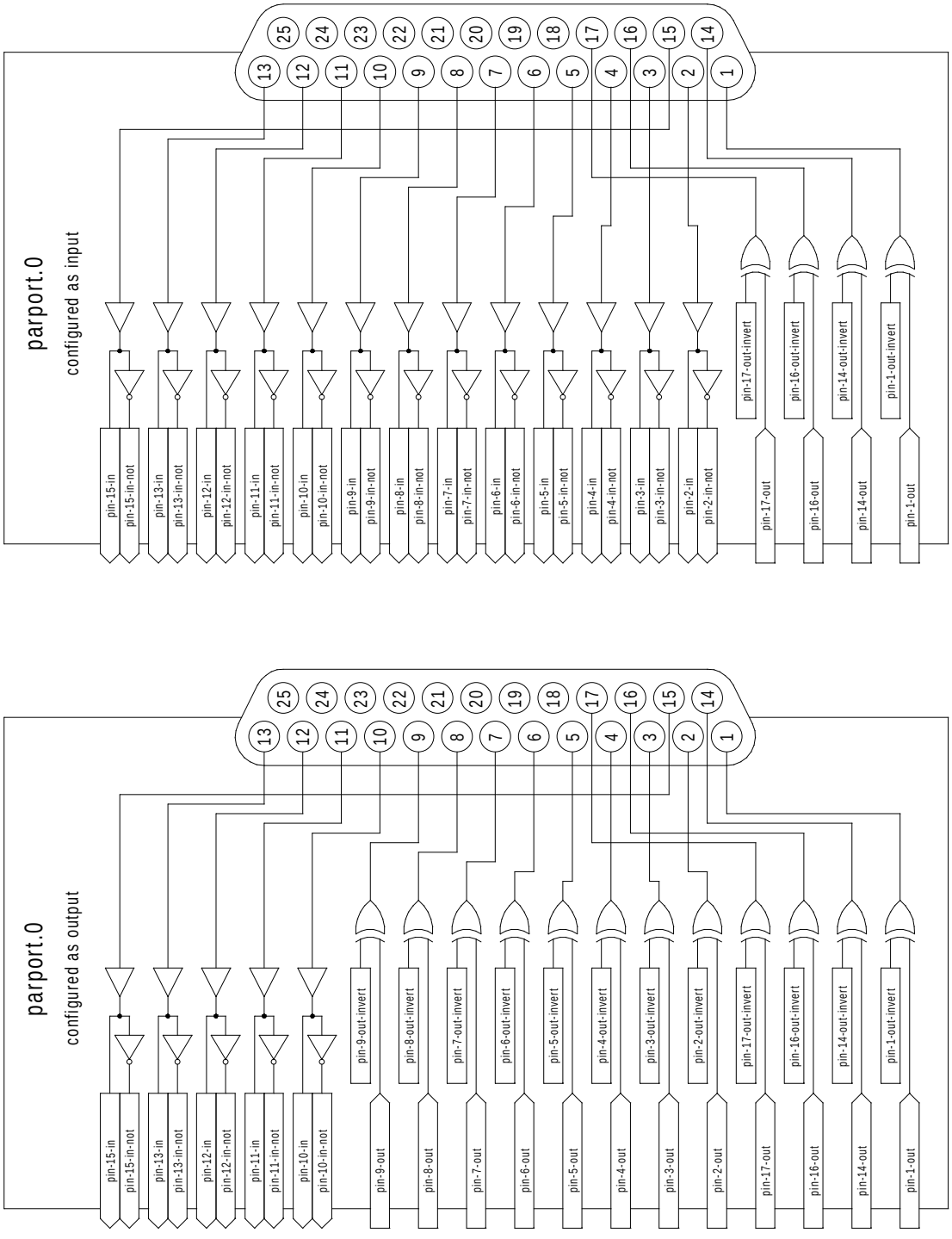


FIG. 12.1 – Diagrammes blocs de Parport

12.1.4 Fonctions

– (FUNCT) `parport.<portnum>.read` – Lit les pins physiques du port `<portnum>` et met à jour les pins de HAL `-in` et `-in-not`.
– (FUNCT) `parport.read-all` – Lit les pins physiques de tous les ports et met à jour les pins de

HAL -in et -in-not.
- (FUNCT) parport.<portnum>.write – Lit les pins de HAL -out du port <portnum> et met à jour les pins de sortie physiques correspondantes.
- (FUNCT) parport.write-all – Lit les pins de HAL -out de tous les ports et met à jour toutes les pins de sortie physiques.
- (FUNCT) parport.<portnum>.reset – Attends que le délai de mise à jour soit écoulé depuis le dernière écriture, remet à jour les pins aux valeurs indiquées par -out-invert et les paramètres de -out-invert. La réinitialisation doit être plus tard dans le même thread que l'écriture.

Les différentes fonctions sont prévues pour les situations où un port doit être mis à jour dans un thread très rapide, mais d'autres ports peuvent être mis à jour dans un thread plus lent pour gagner du temps CPU. Ce n'est probablement pas une bonne idée d'utiliser en même temps, les fonctions -all et une fonction individuelle.

12.1.5 Problèmes courants

Si le chargement du module le message suivant :

```
insmod : error inserting '/home/jepler/emc2/rtlib/hal_parport.ko' :
-1 Device or resource busy
```

s'assurer que le noyau du kernel standard, parport_pc, n'est pas chargé[2] et qu'aucun périphérique dans le système ne revendique les ports concernés.

SI le module est chargé mais ne semble pas fonctionner, l'adresse du port est incorrecte ou le module probe_parport est requis.

12.2 probe_parport

Dans les PC récents, le port parallèle peut exiger une configuration plug and play (PNP) avant d'être utilisable. Le module probe_parport effectue la configuration de tous les ports PNP présents et devrait être chargé avant hal_parport. Sur les machines sans ports PNP, il peut être chargé mais n'a aucun effet.

12.2.1 Installation

```
loadrt probe_parport
loadrt hal_parport ...
```

Si le kernel Linux affiche un message similaire à :

```
parport : PnPBIOS parport detected.
```

Quand le module parport_pc est chargé, avec la commande : sudo modprobe -a parport_pc; sudo rmmod parport_pc, l'utilisation de ce module sera probablement nécessaire.

.

Les modules commerciaux ci-dessous pouront êtres traduits en français sur demande.

The commercial modules below could be translated into French on request.

.

[2] In the emc packages for Ubuntu, the file /etc/modprobe.d/emc2 generally prevents parport_pc from being automatically loaded.

12.3 AX5214H

The Axiom Measurement & Control AX5214H is a 48 channel digital I/O board. It plugs into an ISA bus, and resembles a pair of 8255 chips.[3]

12.3.1 Installing

```
loadrt hal_ax5214h cfg="<config-string>"
```

The config string consists of a hex port address, followed by an 8 character string of "I" and "O" which sets groups of pins as inputs and outputs. The first two character set the direction of the first two 8 bit blocks of pins (0-7 and 8-15). The next two set blocks of 4 pins (16-19 and 20-23). The pattern then repeats, two more blocks of 8 bits (24-31 and 32-39) and two blocks of 4 bits (40-43 and 44-47). If more than one board is installed, the data for the second board follows the first. As an example, the string **"0x220 IIIOIIOO 0x300 OIOOIOIO"** installs drivers for two boards. The first board is at address 0x220, and has 36 inputs (0-19 and 24-39) and 12 outputs (20-23 and 40-47). The second board is at address 0x300, and has 20 inputs (8-15, 24-31, and 40-43) and 28 outputs (0-7. 16-23, 32-39, and 44-47). Up to 8 boards may be used in one system.

12.3.2 Pins

- (BIT) ax5214.<boardnum>.out-<pinnum> – Drives a physical output pin.
- (BIT) ax5214.<boardnum>.in-<pinnum> – Tracks a physical input pin.
- (BIT) ax5214.<boardnum>.in-<pinnum>-not – Tracks a physical input pin, inverted.

For each pin, <boardnum> is the board number (starts at zero), and <pinnum> is the I/O channel number (0 to 47).

Note that the driver assumes active LOW signals. This is so that modules such as OPTO-22 will work correctly (TRUE means output ON, or input energized). If the signals are being used directly without buffering or isolation the inversion needs to be accounted for. The in- HAL pin is TRUE if the physical pin is low (OPTO-22 module energized), and FALSE if the physical pin is high (OPTO-22 module off). The in-<pinnum>-not HAL pin is inverted – it is FALSE if the physical pin is low (OPTO-22 module energized). By connecting a signal to one or the other, the user can determine the state of the input.

12.3.3 Parameters

- (BIT) ax5214.<boardnum>.out-<pinnum>-invert – Inverts an output pin.

The -invert parameter determines whether an output pin is active high or active low. If -invert is FALSE, setting the HAL out- pin TRUE drives the physical pin low, turning ON an attached OPTO-22 module, and FALSE drives it high, turning OFF the OPTO-22 module. If -invert is TRUE, then setting the HAL out- pin TRUE will drive the physical pin high and turn the module OFF.

12.3.4 Functions

- (FUNCT) ax5214.<boardnum>.read – Reads all digital inputs on one board.
- (FUNCT) ax5214.<boardnum>.write – Writes all digital outputs on one board.

[3] In fact it may be a pair of 8255 chips, but I'm not sure. If/when someone starts a driver for an 8255 they should look at the ax5214 code, much of the work is already done.

12.4 Servo-To-Go

The Servo-To-Go is one of the first PC motion control cards[4] supported by EMC. It is an ISA card and it exists in different flavours (all supported by this driver). The board includes up to 8 channels of quadrature encoder input, 8 channels of analog input and output, 32 bits digital I/O, an interval timer with interrupt and a watchdog.

12.4.1 Installing :

```
loadrt hal_stg [base=<address>] [num_chan=<nr>] [dio="<dio-string>"] [model=<model>]
```

The base address field is optional ; if it's not provided the driver attempts to autodetect the board. The num_chan field is used to specify the number of channels available on the card, if not used the 8 axis version is assumed. The digital inputs/outputs configuration is determined by a config string passed to insmod when loading the module. The format consists of a four character string that sets the direction of each group of pins. Each character of the direction string is either "I" or "O". The first character sets the direction of port A (Port A - DIO.0-7), the next sets port B (Port B - DIO.8-15), the next sets port C (Port C - DIO.16-23), and the fourth sets port D (Port D - DIO.24-31). The model field can be used in case the driver doesn't autodetect the right card version[5]. For example :

```
loadrt hal_stg base=0x300 num_chan=4 dio="IOIO"
```

This example installs the stg driver for a card found at the base address of 0x300, 4 channels of encoder feedback, DAC's and ADC's, along with 32 bits of I/O configured like this : the first 8 (Port A) configured as Input, the next 8 (Port B) configured as Output, the next 8 (Port C) configured as Input, and the last 8 (Port D) configured as Output

```
loadrt hal_stg
```

This example installs the driver and attempts to autodetect the board address and board model, it installs 8 axes by default along with a standard I/O setup : Port A & B configured as Input, Port C & D configured as Output.

12.4.2 Pins

- (s32) stg.<channel>.counts – Tracks the counted encoder ticks.
- (FLOAT) stg.<channel>.position – Outputs a converted position.
- (FLOAT) stg.<channel>.dac-value – Drives the voltage for the corresponding DAC.
- (FLOAT) stg.<channel>.adc-value – Tracks the measured voltage from the corresponding ADC.
- (BIT) stg.in-<pinnum> – Tracks a physical input pin.
- (BIT) stg.in-<pinnum>-not – Tracks a physical input pin, but inverted.
- (BIT) stg.out-<pinnum> – Drives a physical output pin

For each pin, <channel> is the axis number, and <pinnum> is the logic pin number of the STG[6].

The in- HAL pin is TRUE if the physical pin is high, and FALSE if the physical pin is low. The in-<pinnum>-not HAL pin is inverted – it is FALSE if the physical pin is high. By connecting a signal to one or the other, the user can determine the state of the input.

[4] a motion control card usually is a board containing devices to control one or more axes (the control devices are usually DAC's to set an analog voltage, encoder counting chips for feedback, etc.)

[5] hint : after starting up the driver, 'dmesg' can be consulted for messages relevant to the driver (e.g. autodetected version number and base address)

[6] if IIOO is defined, there are 16 input pins (in-00 .. in-15) and 16 output pins (out-00 .. out-15), and they correspond to PORTs ABCD (in-00 is PORTA.0, out-15 is PORTD.7)

12.4.3 Parameters

- (FLOAT) stg.<channel>.position-scale – The number of counts / user unit (to convert from counts to units).
- (FLOAT) stg.<channel>.dac-offset – Sets the offset for the corresponding DAC.
- (FLOAT) stg.<channel>.dac-gain – Sets the gain of the corresponding DAC.
- (FLOAT) stg.<channel>.adc-offset – Sets the offset of the corresponding ADC.
- (FLOAT) stg.<channel>.adc-gain – Sets the gain of the corresponding ADC.
- (BIT) stg.out-<pinnum>-invert – Inverts an output pin.

The -invert parameter determines whether an output pin is active high or active low. If -invert is FALSE, setting the HAL out- pin TRUE drives the physical pin high, and FALSE drives it low. If -invert is TRUE, then setting the HAL out- pin TRUE will drive the physical pin low.

12.4.4 Functions

- (FUNCT) stg.capture-position – Reads the encoder counters from the axis <channel>.
- (FUNCT) stg.write-dacs – Writes the voltages to the DACs.
- (FUNCT) stg.read-adcs – Reads the voltages from the ADCs.
- (FUNCT) stg.di-read – Reads physical in- pins of all ports and updates all HAL in- and in-<pinnum>-not pins.
- (FUNCT) stg.do-write – Reads all HAL out- pins and updates all physical output pins.

12.5 Mesa Electronics m5i20 “Anything I/O Card”

The Mesa Electronics m5i20 card consists of an FPGA that can be loaded with a wide variety of configurations, and has 72 pins that leave the PC. The assignment of the pins depends on the FPGA configuration. Currently there is a HAL driver for the “4 axis host based motion control” configuration, and this FPGA configurations is also provided with EMC2. It provides 8 encoder counters, 4 PWM outputs (normally used as DACs) and up to 48 digital I/O channels, 32 inputs and 16 outputs.[7]

Installing :

```
loadrt hal_m5i20 [loadFpga=1|0] [dacRate=<rate>]
```

If **loadFpga** is 1 (the default) the driver will load the FPGA configuration on startup. If it is 0, the driver assumes the configuration is already loaded. **dacRate** sets the carrier frequency for the PWM outputs, in Hz. The default is 32000, for 32KHz PWM. Valid values are from 1 to 32226. The driver prints some useful debugging message to the kernel log, which can be viewed with **dmesg**.

Up to 4 boards may be used in one system.

12.5.1 Pins

In the following pins, parameters, and functions, <board> is the board ID. According to the naming conventions the first board should always have an ID of zero, however this driver uses the PCI board ID, so it may be non-zero even if there is only one board.

- (s32) m5i20.<board>.enc-<channel>-count – Encoder position, in counts.
- (FLOAT) m5i20.<board>.enc-<channel>-position – Encoder position, in user units.
- (BIT) m5i20.<board>.enc-<channel>-index – Current status of index pulse input ?
- (BIT) m5i20.<board>.enc-<channel>-index-enable – when TRUE, and an index pulse appears on the encoder input, reset counter to zero and clear index-enable.

[7] Ideally the encoders, “DACs”, and digital I/O would comply with the canonical interfaces defined earlier, but they don’t. Fixing that is on the things-to-do list.

- (BIT) m5i20.<board>.enc-<channel>-reset – When true, counter is forced to zero.
- (BIT) m5i20.<board>.dac-<channel>-enable – Enables DAC if true. DAC outputs zero volts if false ?
- (FLOAT) m5i20.<board>.dac-<channel>-value – Analog output value for PWM "DAC" (in user units, see -scale and -offset)
- (BIT) m5i20.<board>.in-<channel> – State of digital input pin, see canonical digital input.
- (BIT) m5i20.<board>.in-<channel>-not – Inverted state of digital input pin, see canonical digital input.
- (BIT) m5i20.<board>.out-<channel> – Value to be written to digital output, see canonical digital output.
- (BIT) m5i20.<board>.estop-in – Dedicated estop input, more details needed.
- (BIT) m5i20.<board>.estop-in-not – Inverted state of dedicated estop input.
- (BIT) m5i20.<board>.watchdog-reset – Bidirectional, - Set TRUE to reset watchdog once, is automatically cleared. If bit value 16 is set in watchdog-control then this value is not used, and the hardware watchdog is cleared every time the dac-write function is executed.

12.5.2 Parameters

- (FLOAT) m5i20.<board>.enc-<channel>-scale – The number of counts / user unit (to convert from counts to units).
- (FLOAT) m5i20.<board>.dac-<channel>-offset – Sets the DAC offset.
- (FLOAT) m5i20.<board>.dac-<channel>-gain – Sets the DAC gain (scaling).
- (BIT) m5i20.<board>.dac-<channel>-interlaced – Sets the DAC to interlaced mode. Use this mode if you are filtering the PWM to generate an anaolg voltage.[8]
- (BIT) m5i20.<board>.out-<channel>-invert – Inverts a digital output, see canonical digital output.
- (U32) m5i20.<board>.watchdog-control – Configures the watchdog. The value may be a bitwise OR of the following values :

Bit #	Value	Meaning
0	1	Watchdog is enabled
1	2	Watchdog is automatically reset by DAC writes (the HAL dac-write function)

Typically, the useful values are 0 (watchdog disabled) or 3 (watchdog enabled, cleared by dac-write).
- (U32) m5i20.<board>.led-view – Maps some of the I/O to onboard LEDs. See table below.

12.5.3 Functions

- (FUNCT) m5i20.<board>.encoder-read – Reads all encoder counters.
- (FUNCT) m5i20.<board>.digital-in-read – Reads digital inputs.
- (FUNCT) m5i20.<board>.dac-write – Writes the voltages (PWM duty cycles) to the "DACs".
- (FUNCT) m5i20.<board>.digital-out-write – Writes digital outputs.
- (FUNCT) m5i20.<board>.misc-update – Writes watchdog timer configuration to hardware. Resets watchdog timer. Updates E-stop pin (more info needed). Updates onboard LEDs.

12.5.4 Connector pinout

The Hostmot-4 FPGA configuration has the following pinout. There are three 50-pin ribbon cable connectors on the card : P2, P3, and P4. There are also 8 status LEDs.

[8] With normal 10 bit PWM, 50% duty cycle would be 512 cycles on and 512 cycles off = ca 30 kHz with 33 MHz reference counter. With fully interleaved PWM this would be 1 cycle on, 1 cycle off for 1024 cycles (16.66 MHz if the PWM reference counter runs at 33 MHz) = much easier to filter. The 5I20 configuration interlace is somewhat between non and fully interlaced (to make it easy to filter but not have as many transistions as fully interleaved).

12.5.4.1 Connector P2

m5i20 card connector P2	Function/HAL-pin
1	enc-01 A input
3	enc-01 B input
5	enc-00 A input
7	enc-00 B input
9	enc-01 index input
11	enc-00 index input
13	dac-01 output
15	dac-00 output
17	DIR output for dac-01
19	DIR output for dac-00
21	dac-01-enable output
23	dac-00-enable output
25	enc-03 B input
27	enc-03 A input
29	enc-02 B input
31	enc-02 A input
33	enc-03 index input
35	enc-02 index input
37	dac-03 output
39	dac-02 output
41	DIR output for dac-03
43	DIR output for dac-02
45	dac-03-enable output
47	dac-02-enable output
49	Power +5 V (or +3.3V?)
all even pins	Ground

12.5.4.2 Connector P3

Encoder counters 4 - 7 work simultaneously with in-00 to in-11.

If you are using in-00 to in-11 as general purpose IO then reading enc-<4-7> will produce some random junk number.

m5i20 card connector P3	Function/HAL-pin	Secondary Function/HAL-pin
1	in-00	enc-04 A input
3	in-01	enc-04 B input
5	in-02	enc-04 index input
7	in-03	enc-05 A input
9	in-04	enc-05 B input
11	in-05	enc-05 index input
13	in-06	enc-06 A input
15	in-07	enc-06 B input
17	in-08	enc-06 index input
19	in-09	enc-07 A input
21	in-10	enc-07 B input
23	in-11	enc-07 index input
25	in-12	
27	in-13	
29	in-14	
31	in-15	
33	out-00	
35	out-01	
37	out-02	
39	out-03	
41	out-04	
43	out-05	
45	out-06	
47	out-07	
49	Power +5 V (or +3.3V?)	
all even pins	Ground	

12.5.4.3 Connector P4

The index mask masks the index input of the encoder so that the encoder index can be combined with a mechanical switch or opto detector to clear or latch the encoder counter only when the mask input bit is in proper state (selected by mask polarity bit) and encoder index occurs. This is useful for homing. The behaviour of these pins is controlled by the Counter Control Register (CCR), however there is currently no function in the driver to change the CCR. See REGMAP4[9] for a description of the CCR.

[9] emc2/src/hal/drivers/m5i20/REGMAP4E

m5i20 card connector P4	Function/HAL-pin	Secondary Function/HAL-pin
1	in-16	enc-00 index mask
3	in-17	enc-01 index mask
5	in-18	enc-02 index mask
7	in-19	enc-03 index mask
9	in-20	
11	in-21	
13	in-22	
15	in-23	
17	in-24	enc-04 index mask
19	in-25	enc-05 index mask
21	in-26	enc-06 index mask
23	in-27	enc-07 index mask
25	in-28	
27	in-29	
29	in-30	
31	in-31	
33	out-08	
35	out-09	
37	out-10	
39	out-11	
41	out-12	
43	out-13	
45	out-14	
47	out-15	
49	Power +5 V (or +3.3V?)	
all even pins	Ground	

12.5.4.4 LEDs

The status LEDs will monitor one motion channel set by the `m5i20.<board>.led-view` parameter. A call to `m5i20.<board>.misc-update` is required to update the viewed channel.

LED name	Output
LED0	IRQLatch?
LED1	enc-<channel> A
LED2	enc-<channel> B
LED3	enc-<channel> index
LED4	dac-<channel> DIR
LED5	dac-<channel>
LED6	dac-<channel>-enable
LED7	watchdog timeout?

12.6 Vital Systems Motenc-100 and Motenc-LITE

The Vital Systems Motenc-100 and Motenc-LITE are 8- and 4-channel servo control boards. The Motenc-100 provides 8 quadrature encoder counters, 8 analog inputs, 8 analog outputs, 64 (68?) digital inputs, and 32 digital outputs. The Motenc-LITE has only 4 encoder counters, 32 digital inputs and 16 digital outputs, but it still has 8 analog inputs and 8 analog outputs. The driver automatically identifies the installed board and exports the appropriate HAL objects.[10]

Installing :

[10] Ideally the encoders, DACs, ADCs, and digital I/O would comply with the canonical interfaces defined earlier, but they don't. Fixing that is on the things-to-do list.

```
loadrt hal_motenc
```

During loading (or attempted loading) the driver prints some usefull debugging message to the kernel log, which can be viewed with **dmesg**.

Up to 4 boards may be used in one system.

12.6.1 Pins

In the following pins, parameters, and functions, <board> is the board ID. According to the naming conventions the first board should always have an ID of zero. However this driver sets the ID based on a pair of jumpers on the baord, so it may be non-zero even if there is only one board.

- (S32) motenc.<board>.enc-<channel>-count – Encoder position, in counts.
- (FLOAT) motenc.<board>.enc-<channel>-position – Encoder position, in user units.
- (BIT) motenc.<board>.enc-<channel>-index – Current status of index pulse input.
- (BIT) motenc.<board>.enc-<channel>-idx-latch – Driver sets this pin true when it latches an index pulse (enabled by latch-index). Cleared by clearing latch-index.
- (BIT) motenc.<board>.enc-<channel>-latch-index – If this pin is true, the driver will reset the counter on the next index pulse.
- (BIT) motenc.<board>.enc-<channel>-reset-count – If this pin is true, the counter will immediately be reset to zero, and the pin will be cleared.
- (FLOAT) motenc.<board>.dac-<channel>-value – Analog output value for DAC (in user units, see -gain and -offset)
- (FLOAT) motenc.<board>.adc-<channel>-value – Analog input value read by ADC (in user units, see -gain and -offset)
- (BIT) motenc.<board>.in-<channel> – State of digital input pin, see canonical digital input.
- (BIT) motenc.<board>.in-<channel>-not – Inverted state of digital input pin, see canonical digital input.
- (BIT) motenc.<board>.out-<channel> – Value to be written to digital output, seen canonical digital output.
- (BIT) motenc.<board>.estop-in – Dedicated estop input, more details needed.
- (BIT) motenc.<board>.estop-in-not – Inverted state of dedicated estop input.
- (BIT) motenc.<board>.watchdog-reset – Bidirectional, - Set TRUE to reset watchdog once, is automatically cleared.

12.6.2 Parameters

- (FLOAT) motenc.<board>.enc-<channel>-scale – The number of counts / user unit (to convert from counts to units).
- (FLOAT) motenc.<board>.dac-<channel>-offset – Sets the DAC offset.
- (FLOAT) motenc.<board>.dac-<channel>-gain – Sets the DAC gain (scaling).
- (FLOAT) motenc.<board>.adc-<channel>-offset – Sets the ADC offset.
- (FLOAT) motenc.<board>.adc-<channel>-gain – Sets the ADC gain (scaling).
- (BIT) motenc.<board>.out-<channel>-invert – Inverts a digital output, see canonical digital output.
- (U32) motenc.<board>.watchdog-control – Configures the watchdog. The value may be a bitwise OR of the following values :

Bit #	Value	Meaning
0	1	Timeout is 16ms if set, 8ms if unset
2	4	Watchdog is enabled
4	16	Watchdog is automatically reset by DAC writes (the HAL dac-write function)

Typically, the useful values are 0 (watchdog disabled) or 20 (8ms watchdog enabled, cleared by dac-write).

- (U32) motenc.<board>.led-view – Maps some of the I/O to onboard LEDs ?

12.6.3 Functions

- (FUNCT) motenc.<board>.encoder-read – Reads all encoder counters.
- (FUNCT) motenc.<board>.adc-read – Reads the analog-to-digital converters.
- (FUNCT) motenc.<board>.digital-in-read – Reads digital inputs.
- (FUNCT) motenc.<board>.dac-write – Writes the voltages to the DACs.
- (FUNCT) motenc.<board>.digital-out-write – Writes digital outputs.
- (FUNCT) motenc.<board>.misc-update – Updates misc stuff.

12.7 Pico Systems PPMC (Parallel Port Motion Control)

Pico Systems has a family of boards for doing servo, stepper, and pwm control. The boards connect to the PC through a parallel port working in EPP mode. Although most users connect one board to a parallel port, in theory any mix of up to 8 or 16 boards can be used on a single parport. One driver serves all types of boards. The final mix of I/O depends on the connected board(s). The driver doesn't distinguish between boards, it simply numbers I/O channels (encoders, etc) starting from 0 on the first card.

Installing :

```
loadrt hal_ppmc port_addr=<addr1>[,<addr2>[,<addr3>...]]
```

The **port_addr** parameter tells the driver what parallel port(s) to check. By default, **<addr1>** is 0x0378, and **<addr2>** and following are not used. The driver searches the entire address space of the enhanced parallel port(s) at **port_addr**, looking for any board(s) in the PPMC family. It then exports HAL pins for whatever it finds. During loading (or attempted loading) the driver prints some usefull debugging message to the kernel log, which can be viewed with **dmesg**.

Up to 3 parport busses may be used, and each bus may have up to 8 devices on it.

12.7.1 Pins

In the following pins, parameters, and functions, <board> is the board ID. According to the naming conventions the first board should always have an ID of zero. However this driver sets the ID based on a pair of jumpers on the baord, so it may be non-zero even if there is only one board.

- (S32) ppmc.<port>.encoder.<channel>.count – Encoder position, in counts.
- (S32) ppmc.<port>.encoder.<channel>.delta – Change in counts since last read.
- (FLOAT) ppmc.<port>.encoder.<channel>.position – Encoder position, in user units.
- (BIT) ppmc.<port>.encoder.<channel>.index – Something to do with index pulse.[11]
- (BIT) ppmc.<port>.pwm.<channel>.enable – Enables a PWM generator.
- (FLOAT) ppmc.<port>.pwm.<channel>.value – Value which determines the duty cycle of the PWM waveforms. The value is divided by pwm.<channel>.scale, and if the result is 0.6 the duty cycle will be 60%, and so on. Negative values result in the duty cycle being based on the absolute value, and the direction pin is set to indicate negative.
- (BIT) ppmc.<port>.stepgen.<channel>.enable – Enables a step pulse generator.
- (FLOAT) ppmc.<port>.stepgen.<channel>.velocity – Value which determines the step frequency. The value is multiplied by stepgen.<channel>.scale, and the result is the frequency in steps per second. Negative values result in the frequency being based on the absolute value, and the direction pin is set to indicate negative.
- (BIT) ppmc.<port>.in-<channel> – State of digital input pin, see canonical digital input.
- (BIT) ppmc.<port>.in.<channel>.not – Inverted state of digital input pin, see canonical digital input.
- (BIT) ppmc.<port>.out-<channel> – Value to be written to digital output, seen canonical digital output.

[11] Index handling does _not_ comply with the canonical encoder interface, and should be changed.

12.7.2 Parameters

- (FLOAT) ppmc.<port>.enc.<channel>.scale – The number of counts / user unit (to convert from counts to units).
- (FLOAT) ppmc.<port>.pwm.<channel-range>.freq – The PWM carrier frequency, in Hz. Applies to a group of four consecutive PWM generators, as indicated by <channel-range>. Minimum is 153Hz, maximum is 500KHz.
- (FLOAT) ppmc.<port>.pwm.<channel>.scale – Scaling for PWM generator. If scale is X, then the duty cycle will be 100% when the value pin is X (or -X).
- (FLOAT) ppmc.<port>.pwm.<channel>.max-dc – Maximum duty cycle, from 0.0 to 1.0.
- (FLOAT) ppmc.<port>.pwm.<channel>.min-dc – Minimum duty cycle, from 0.0 to 1.0.
- (FLOAT) ppmc.<port>.pwm.<channel>.duty-cycle – Actual duty cycle (used mostly for troubleshooting.)
- (BIT) ppmc.<port>.pwm.<channel>.bootstrap – If true, the PWM generator will generate a short sequence of pulses of both polarities when it is enabled, to charge the bootstrap capacators used on some MOSFET gate drivers.
- (U32) ppmc.<port>.stepgen.<channel-range>.setup-time – Sets minimum time between direction change and step pulse, in units of 100nS. Applies to a group fof four consecutive PWM generators, as indicated by <channel-range>.
- (U32) ppmc.<port>.stepgen.<channel-range>.pulse-width – Sets width of step pulses, in units of 100nS. Applies to a group fof four consecutive PWM generators, as indicated by <channel-range>.
- (U32) ppmc.<port>.stepgen.<channel-range>.pulse-space-min – Sets minimum time between pulses, in units of 100nS. The maximum step rate is 1/(100nS * (pulse-width + pulse-space-min)). Applies to a group fof four consecutive PWM generators, as indicated by <channel-range>.
- (FLOAT) ppmc.<port>.stepgen.<channel>.scale – Scaling for step pulse generator. The step frequency in Hz is the absolute value of velocity * scale.
- (FLOAT) ppmc.<port>.stepgen.<channel>.max-vel – The maximum value for velocity. Commands greater than max-vel will be clamped. Also applies to negative values. (The absolute value is clamped.)
- (FLOAT) ppmc.<port>.stepgen.<channel>.frequency – Actual step pulse frequency in Hz (used mostly for troubleshooting.)
- (BIT) ppmc.<port>.out.<channel>.invert – Inverts a digital output, see canonical digital output.

12.7.3 Functions

- (FUNCT) ppmc.<port>.read – Reads all inputs (digital inputs and encoder counters) on one port.
- (FUNCT) ppmc.<port>.write – Writes all outputs (digital outputs, stepgens, PWMs) on one port.

12.8 Pluto-P : generalities

The Pluto-P is an inexpensive ($60) FPGA board featuring the ACEX1K chip from Altera.

12.8.1 Requirements

1. A Pluto-P board
2. An EPP-compatible parallel port, configured for EPP mode in the system BIOS

12.8.2 Connectors

- The Pluto-P board is shipped with the left connector presoldered, with the key in the indicated position. The other connectors are unpopulated. There does not seem to be a standard 12-pin IDC connector, but some of the pins of a 16P connector can hang off the board next to QA3/QZ3.

– The bottom and right connectors are on the same .1" grid, but the left connector is not. If OUT2. . . OUT9 are not required, a single IDC connector can span the bottom connector and the bottom two rows of the right connector.

12.8.3 Physical Pins

– Read the ACEX1K datasheet for information about input and output voltage thresholds. The pins are all configured in "LVTTL/LVCMOS" mode and are generally compatible with 5V TTL logic.
– Before configuration and after properly exiting emc2, all Pluto-P pins are tristated with weak pull-ups (20kΩ min, 50kΩ max). If the watchdog timer is enabled (the default), these pins are also tristated after an interruption of communication between emc2 and the board. The watchdog timer takes approximately 6.5ms to activate. However, software bugs in the pluto_servo firmware or emc2 can leave the Pluto-P pins in an undefined state.
– In pwm+dir mode, by default dir is HIGH for negative values and LOW for positive values. To select HIGH for positive values and LOW for negative values, set the corresponding dout-NN-invert parameter TRUE to invert the signal.
– The index input is triggered on the rising edge. Initial testing has shown that the QZx inputs are particularly noise sensitive, due to being polled every 25ns. Digital filtering has been added to filter pulses shorter than 175ns (seven polling times). Additional external filtering on all input pins, such as a Schmitt buffer or inverter, RC filter, or differential receiver (if applicable) is recommended.
– The IN1. . . IN7 pins have 22-ohm series resistors to their associated FPGA pins. No other pins have any sort of protection for out-of-spec voltages or currents. It is up to the integrator to add appropriate isolation and protection. Traditional parallel port optoisolator boards do not work with pluto_servo due to the bidirectional nature of the EPP protocol.

12.8.4 LED

– When the device is unprogrammed, the LED glows faintly. When the device is programmed, the LED glows according to the duty cycle of PWM0 (**LED** = **UP0** *xor* **DOWN0**) or STEPGEN0 (**LED** = **STEP0** *xor* **DIR0**).

12.8.5 Power

– A small amount of current may be drawn from VCC. The available current depends on the unregulated DC input to the board. Alternately, regulated +3.3VDC may be supplied to the FPGA through these VCC pins. The required current is not yet known, but is probably around 50mA plus I/O current.
– The regulator on the Pluto-P board is a low-dropout type. Supplying 5V at the power jack will allow the regulator to work properly.

12.8.6 PC interface

– At present, only a single pluto_servo or pluto_step board is supported. At present there is no provision for multiple boards on one parallel port (because all boards reside at the same EPP address) but supporting one board per parallel port should be possible.

12.8.7 Rebuilding the FPGA firmware

The `src/hal/drivers/pluto_servo_firmware/` and `src/hal/drivers/pluto_step_firmware/` subdirectories contain the Verilog source code plus additional files used by Quartus for the FPGA firmwares. Altera's Quartus II software is required to rebuild the FPGA firmware. To rebuild the

firmware from the .hdl and other source files, open the `.qpf` file and press CTRL-L. Then, recompile emc2.

Like the HAL hardware driver, the FPGA firmware is licensed under the terms of the GNU General Public License.

The gratis version of Quartus II runs only on Microsoft Windows, although there is apparently a paid version that runs on Linux.

12.8.8 For more information

The Pluto-P board may be ordered from http://www.knjn.com/ShopBoards_Parallel.html (US based, international shipping is available). Some additional information about it is available from http://www.fpga4fun.com/board_pluto-P.html and from the developer's blog http://emergent.unpy.net/01165081407.

12.9 pluto-servo : Hardware PWM and quadrature counting

The pluto_servo system is suitable for control of a 4-axis CNC mill with servo motors, a 3-axis mill with PWM spindle control, a lathe with spindle encoder, etc. The large number of inputs allows a full set of limit switches.

This driver features :

- 4 quadrature channels with 40MHz sample rate. The counters operate in "4x" mode. The maximum useful quadrature rate is 8191 counts per emc2 servo cycle, or about 8MHz for EMC2's default 1ms servo rate.
- 4 PWM channels, "up/down" or "pwm+dir" style. 4095 duty cycles from -100% to +100%, including 0%. The PWM period is approximately 19.5kHz (40MHz / 2047). A PDM-like mode is also available.
- 18 digital outputs : 10 dedicated, 8 shared with PWM functions. (Example : A lathe with unidirectional PWM spindle control may use 13 total digital outputs)
- 20 digital inputs : 8 dedicated, 12 shared with Quadrature functions. (Example : A lathe with index pulse only on the spindle may use 13 total digital inputs)
- EPP communication with the PC. The EPP communication typically takes around 100uS on machines tested so far, enabling servo rates above 1kHz.

12.9.1 Pinout

UPx The "up" (up/down mode) or "pwm" (pwm+direction mode) signal from PWM generator X. May be used as a digital output if the corresponding PWM channel is unused, or the output on the channel is always negative. The corresponding digital output invert may be set to TRUE to make UPx active low rather than active high.

DNx The "down" (up/down mode) or "direction" (pwm+direction mode) signal from PWM generator X. May be used as a digital output if the corresponding PWM channel is unused, or the output on the channel is never negative. The corresponding digital ouput invert may be set to TRUE to make DNx active low rather than active high.

QAx, QBx The A and B signals for Quadrature counter X. May be used as a digital input if the corresponding quadrature channel is unused.

QZx The Z (index) signal for quadrature counter X. May be used as a digital input if the index feature of the corresponding quadrature channel is unused.

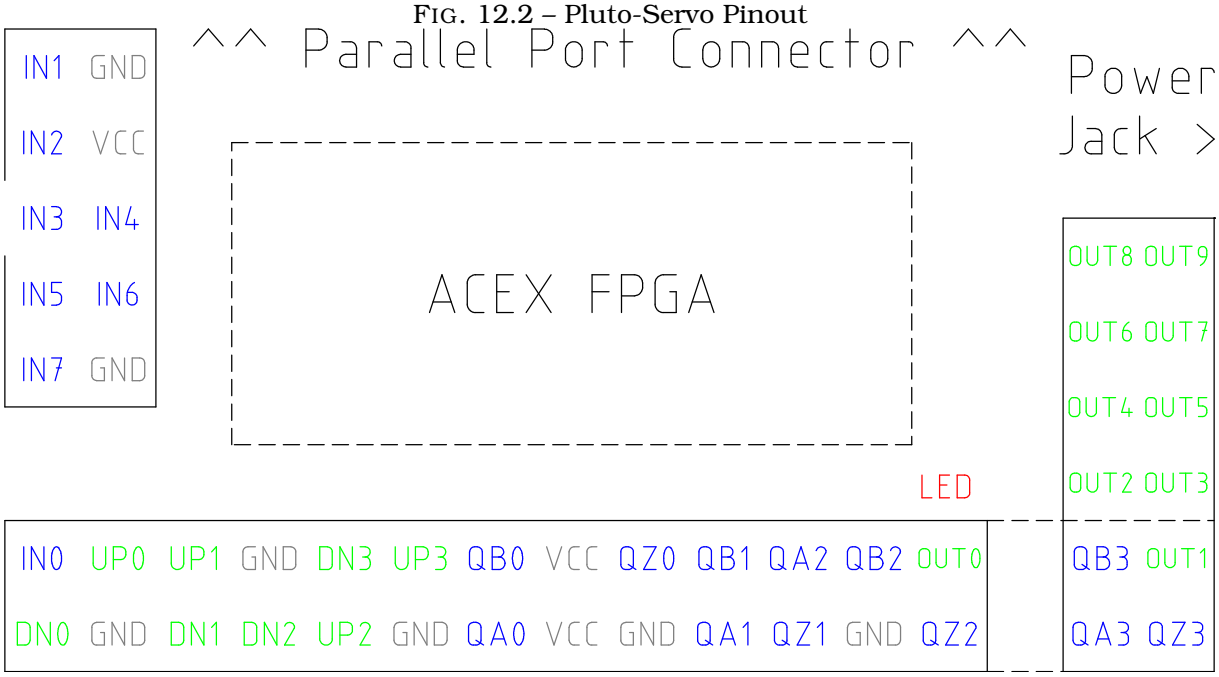
INx Dedicated digital input #x

OUTx Dedicated digital output #x

GND Ground

VCC +3.3V regulated DC

FIG. 12.2 – Pluto-Servo Pinout



TAB. 12.1 – Pluto-Servo Alternate Pin Functions

Primary function	Alternate Function	Behavior if both functions used
UP0	PWM0	When pwm-0-pwmdir is TRUE, this pin is the PWM output
	OUT10	XOR'd with UP0 or PWM0
UP1	PWM1	When pwm-1-pwmdir is TRUE, this pin is the PWM output
	OUT12	XOR'd with UP1 or PWM1
UP2	PWM2	When pwm-2-pwmdir is TRUE, this pin is the PWM output
	OUT14	XOR'd with UP2 or PWM2
UP3	PWM3	When pwm-3-pwmdir is TRUE, this pin is the PWM output
	OUT16	XOR'd with UP3 or PWM3
DN0	DIR0	When pwm-0-pwmdir is TRUE, this pin is the DIR output
	OUT11	XOR'd with DN0 or DIR0
DN1	DIR1	When pwm-1-pwmdir is TRUE, this pin is the DIR output
	OUT13	XOR'd with DN1 or DIR1
DN2	DIR2	When pwm-2-pwmdir is TRUE, this pin is the DIR output
	OUT15	XOR'd with DN2 or DIR2
DN3	DIR3	When pwm-3-pwmdir is TRUE, this pin is the DIR output
	OUT17	XOR'd with DN3 or DIR3
QZ0	IN8	Read same value
QZ1	IN9	Read same value
QZ2	IN10	Read same value
QZ3	IN11	Read same value
QA0	IN12	Read same value
QA1	IN13	Read same value
QA2	IN14	Read same value
QA3	IN15	Read same value
QB0	IN16	Read same value
QB1	IN17	Read same value
QB2	IN18	Read same value
QB3	IN19	Read same value

12.9.2 Input latching and output updating

- PWM duty cycles for each channel are updated at different times.
- Digital outputs OUT0 through OUT9 are all updated at the same time. Digital outputs OUT10 through OUT17 are updated at the same time as the pwm function they are shared with.
- Digital inputs IN0 through IN19 are all latched at the same time.
- Quadrature positions for each channel are latched at different times.

12.9.3 HAL Functions, Pins and Parameters

A list of all 'loadrt' arguments, HAL function names, pin names and parameter names is in the manual page, *pluto_servo.9*.

12.9.4 Compatible driver hardware

A schematic for a 2A, 2-axis PWM servo amplifier board is available (http://emergent.unpy.net/projects/01148303608). The L298 H-Bridge (L298 H-bridge http://www.st.com/stonline/books/pdf/docs/1773.pdf) is inexpensive and can easily be used for motors up to 4A (one motor per L298) or up to 2A (two motors per L298) with the supply voltage up to 46V. However, the L298 does not have built-in current limiting, a problem for motors with high stall currents. For higher currents and voltages, some users have reported success with International Rectifier's integrated high-side/low-side drivers. (http://www.cnczone.com/forums/showthread.php?t=25929)

12.10 Pluto-step : 300kHz Hardware Step Generator

Pluto-step is suitable for control of a 3- or 4-axis CNC mill with stepper motors. The large number of inputs allows for a full set of limit switches.

The board features :

- 4 "step+direction" channels with 312.5kHz maximum step rate, programmable step length, space, and direction change times
- 14 dedicated digital outputs
- 16 dedicated digital inputs
- EPP communuication with the PC

12.10.1 Pinout

STEPx The "step" (clock) output of stepgen channel **x**

DIRx The "direction" output of stepgen channel **x**

INx Dedicated digital input #x

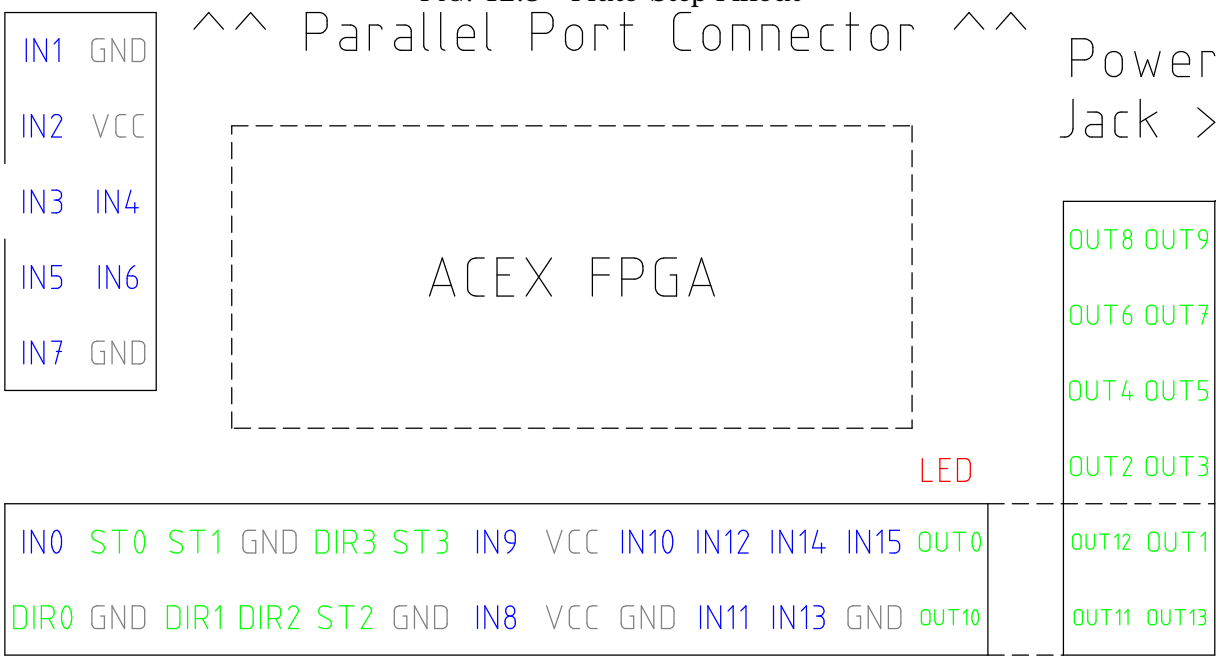
OUTx Dedicated digital output #x

GND Ground

VCC +3.3V regulated DC

While the "extended main connector" has a superset of signals usually found on a Step & Direction DB25 connector–4 step generators, 9 inputs, and 6 general-purpose outputs–the layout on this header is different than the layout of a standard 26-pin ribbon cable to DB25 connector.

FIG. 12.3 – Pluto-Step Pinout



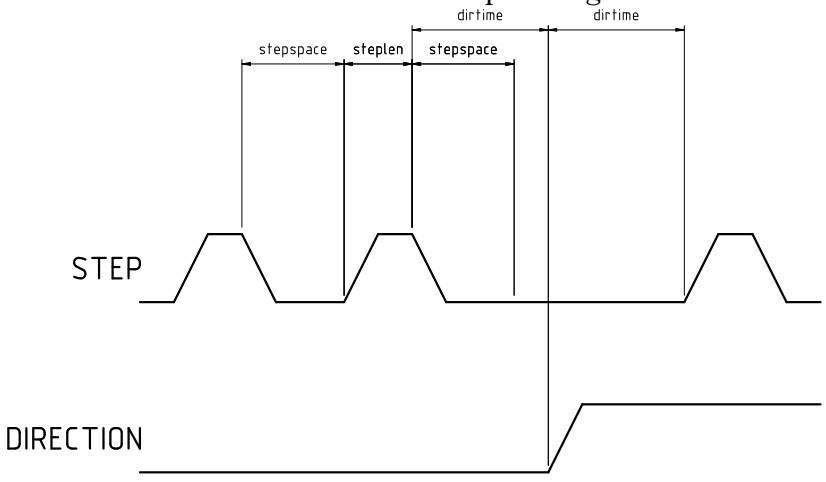
12.10.2 Input latching and output updating

– Step frequencies for each channel are updated at different times.
– Digital outputs are all updated at the same time.
– Digital inputs are all latched at the same time.
– Feedback positions for each channel are latched at different times.

12.10.3 Step Waveform Timings

The firmware and driver enforce step length, space, and direction change times. Timings are rounded up to the next multiple of $1.6\mu s$, with a maximum of $49.6\mu s$. The timings are the same as for the software stepgen component, except that “dirhold” and “dirsetup” have been merged into a single parameter “dirtime” which should be the maximum of the two, and that the same step timings are always applied to all channels.

FIG. 12.4 – Pluto-Step Timings



12.10.4 HAL Functions, Pins and Parameters

A list of all 'loadrt' arguments, HAL function names, pin names and parameter names is in the manual page, *pluto_step.9*.

Chapitre 13

Halui

13.1 Introduction

Halui est une interface utilisateur pour EMC s'appuyant sur HAL, elle connecte les pins de HAL à des commandes NML. La plupart des fonctionnalités (boutons, indicateurs etc.) utilisées par les interfaces graphiques traditionnelles (mini, Axis, etc.), sont fournies par des pins de HAL dans Halui.

La façon la plus facile pour utiliser halui est de modifier votre dossier d'ini pour inclure

```
HALUI = halui
```

dans la section [HAL].

Une solution alternative pour l'invoquer (surtout si vous générez la config avec stepconf) est d'inclure

```
loadusr halui -ini /path/to/inifile.ini
```

dans votre fichier custom.hal.

13.2 Nomenclature des pins d'Halui

13.2.1 Abandon (abort)

– (BIT) halui.abort (in) - pin de requête d'abandon (efface les erreurs)

13.2.2 Axes (axis)

– (FLOAT) halui.axis.n.pos-commanded (out) - Position de l'axe commandée, en coordonnées machine
– (FLOAT) halui.axis.n.pos-feedback (out) - Position de l'axe lue, en coordonnées machine
– (FLOAT) halui.axis.n.pos-relative (out) - Position de l'axe, en coordonnées relatives

13.2.3 Arrêt d'urgence (E-Stop)

– (BIT) halui.estop.activate - pin de requête d'arrêt d'urgence (E-Stop)
– (BIT) halui.estop.reset - pin de requête de relâchement de l'arrêt d'urgence (E-Stop reset)
– (BIT) halui.estop.is-activated - indique si l'arrêt d'urgence est actif

13.2.4 Correcteur de vitesse d'avance (Feed override)

– (BIT) halui.feed-override.count-enable (in) - doit être vraie pour activer le comptage.
– (FLOAT) halui.feed-override.value - Valeur de la correction courante de vitesse d'avance
– (FLOAT) halui.feed-override.scale - pin pour positionner l'échelle des corrections possibles
– (S32) halui.feed-override.counts - comptage depuis un codeur, par exemple pour modifier la correction de vitesse d'avance
– (BIT) halui.feed-override.increase - pin pour augmenter la correction (+=scale)
– (BIT) halui.feed-override.decrease - pin pour diminuer la correction (-=scale)

13.2.5 Brouillard (Mist)

– (BIT) halui.mist.on - pin de requête du brouillard
– (BIT) halui.mist.is-on - indique si le brouillard est actif
– (BIT) halui.mist.off - pin de requête d'arrêt du brouillard

13.2.6 Arrosage (Flood)

– (BIT) halui.flood.on - pin de requête d'arrosage
– (BIT) halui.flood.is-on - indique si l'arrosage est actif
– (BIT) halui.flood.off - pin de requête d'arrêt d'arrosage

13.2.7 Lubrifiant (Lube)

– (BIT) halui.lube.on - pin de requête de graissage
– (BIT) halui.lube.is-on - indique si le graissage est actif
– (BIT) halui.lube.off - pin de requête d'arrêt du graissage

13.2.8 Jog

<n> est un nombre compris entre 0 et 7, ou <selected>.

– (FLOAT) halui.jog.speed - positionne la vitesse de jog
– (FLOAT) halui.jog-deadband - bande morte pour le jogging analogique (les petites vitesses de jog sont sans effet)
– (BIT) halui.jog.<n>.minus - jog en direction négative
– (BIT) halui.jog.<n>.plus - jog en direction positive
– (BIT) halui.jog.<n>.analog - entrée analogique de vitesse de jog (utilisé avec les joysticks ou autres matériels analogiques)
– (BIT) halui.jog.<selected>.minus (in) - jog l'axe « selected » en direction négative et à la vitesse de halui.jog.speed velocity
– (BIT) halui.jog.<selected>.plus (in) - jog l'axe « selected » en direction positive et à la vitesse de halui.jog.speed velocity

13.2.9 Joints

<n> est un nombre compris entre 0 et 7, ou 'selected'.

– (BIT) halui.joint.<n>.home (out) - pin pour la prise d'origine d'un joint spécifique
– (BIT) halui.joint.<n>.on-soft-min-limit - pin de status indiquant que le joint est sur sa limite logicielle négative
– (BIT) halui.joint.<n>.on-soft-max-limit - pin de status indiquant que le joint est sur sa limite logicielle positive

– (BIT) halui.joint.<n>.on-hard-min-limit - pin de status indiquant que le joint est sur son fin de course de limite négative
– (BIT) halui.joint.<n>.on-hard-max-limit - pin de status indiquant que le joint est sur son fin de course de limite positive
– (BIT) halui.joint.<n>.has-fault (out) - pin de status indiquant que le joint est en défaut
– (BIT) halui.joint.<n>.is-homed (out) - pin de status indiquant que le joint est référencé
– (BIT) halui.joint.<n>.is-selected bit (out) - pin indiquant que le joint est « selected » - interne à halui
– (BIT) halui.joint.<n>.select (in) - select joint (0..7) - interne à halui
– (U32) halui.joint.selected (out) - selected joint (0..7) - interne à halui
– (BIT) halui.joint.selected.has-fault (out) - pin de status indiquant que le joint <n> est en défaut
– (BIT) halui.joint.selected.home (in) - pin pour la prise d'origine du joint « selected »
– (BIT) halui.joint.selected.is-homed (out) - pin de status indiquant que le joint « selected » est référencé
– (BIT) halui.joint.selected.on-hard-max-limit (out) - status pin telling that the selected joint is on the positive hardware limit
– (BIT) halui.joint.selected.on-hard-min-limit (out) - pin de status indiquant que le joint « selected » est sur son fin de course de limite négative
– (BIT) halui.joint.selected.on-soft-max-limit (out) - pin de status indiquant que le joint « selected » est sur sa limite logicielle positive
– (BIT) halui.joint.selected.on-soft-min-limit (out) - pin de status indiquant que le joint « selected » est sur sa limite logicielle négative

13.2.10 Marche machine

– (BIT) halui.machine.on - pin de requête de marche machine
– (BIT) halui.machine.off - pin de requête d'arrêt machine
– (BIT) halui.machine.is-on - indique que la machine est en marche

13.2.11 Vitesse maximum

La vitesse linéaire maximum peut être ajustée entre 0 et la valeur de la variable MAX_VELOCITY dans la section [TRAJ] du fichier ini.

– (BIT) halui.max-velocity.count-enable (in) - Si TRUE, la vitesse max est modifiée quand le comptage change
– (S32) halui.max-velocity.counts (in) - vous permet, en agissant sur un codeur, de modifier la vitesse max
– (BIT) halui.max-velocity.decrease (in) - pin pour diminuer la vitesse max
– (BIT) halui.max-velocity.increase (in) - pin pour augmenter la vitesse max
– (FLOAT) halui.max-velocity.scale (in) - Valeur appliquée sur le nombre de fronts montants des pins increase ou decrease en unités machine par seconde.
– (FLOAT) halui.max-velocity.value (out) - Valeur de la vitesse linéaire maximum en unités machine par seconde.

13.2.12 Données manuelles (MDI)

Il arrive que l'utilisateur veuille ajouter des tâches plus complexes devant être effectuées par l'activation d'une pin de HAL. C'est possible en utilisant le schéma de commande en données manuelles (MDI) suivant :

– Une MDI_COMMAND est ajoutée dans la section [HALUI] du fichier ini, par exemple [HALUI] MDI_COMMAND = G0 X0
– Quand halui démarre il va lire/détecter le champ MDI_COMMAND dans le fichier ini et exporter les pins de type (BIT) halui.mdi-command-<nr>, <nr> est un nombre compris entre 00 et le nombre de MDI_COMMAND trouvées dans le fichier ini, avec un maximum de 64 commandes.

– Quand la pin halui.mdi-command-<nr> est activée, halui va essayer d'envoyer au MDI la commande définie dans le fichier ini. Ca ne fonctionnera pas dans tous les modes de fonctionnement où se trouve emc2, par exemple, tant qu'il est en AUTO halui ne peut pas envoyer de commande MDI.

13.2.13 Mode de fonctionnement

– (BIT) halui.mode.manual (in) - pin de requête du mode manuel
– (BIT) halui.mode.is_manual (out) - indique si le mode manuel est actif
– (BIT) halui.mode.auto (in) - pin de requête du mode auto
– (BIT) halui.mode.is_auto (out)- indique si le mode auto est actif
– (BIT) halui.mode.mdi (in) - pin de requête du mode données manuelles
– (BIT) halui.mode.is_mdi (out) - indique si le mode données manuelles est actif
– (BIT) halui.mode.joint (in) - pin de requête du mode jog joint par joint
– (BIT) halui.mode.is-joint (out) - pin indiquant si le mode joint par joint est actif
– (BIT) halui.mode.is-teleop (out) - pin indiquant que le mode jog coordonné est actif

13.2.14 Broche (Spindle)

– (BIT) halui.spindle.start (in) - Marche de la broche
– (BIT) halui.spindle.is-on (out) -
– (BIT) halui.spindle.stop (in) - Arrêt de la broche
– (BIT) halui.spindle.forward (in) - Marche broche en sens horaire
– (BIT) halui.spindle.runs-forward (out) -
– (BIT) halui.spindle.reverse (in) - Marche broche en sens anti-horaire
– (BIT) halui.spindle.runs-backward (out) -
– (BIT) halui.spindle.increase (in) - Augmente la vitesse de broche
– (BIT) halui.spindle.decrease (in) - Diminue la vitesse de broche
– (BIT) halui.spindle.brake-on (in) - pin d'activation du frein de broche
– (BIT) halui.spindle.brake-is-on (out) - indique si le frein est actif
– (BIT) halui.spindle.brake-off (in) - pin de désactivation du frein de broche

13.2.15 Sélection d'un joint

– (U32) halui.joint.select - sélectionne le joint (0..7) - internal halui
– (U32) halui.joint.selected - joint (0..7) sélectionné - internal halui
– (BIT) halui.joint.x.select bit - pins pour sélectinner un joint - internal halui
– (BIT) halui.joint.x.is-selected bit - pin de status indiquant un joint sélectionné - internal halui

13.2.16 Correcteur de vitesse de broche (Spindle override)

– (FLOAT) halui.spindle-override.value (out) - Valeur courante de la correction de vitesse de broche
– (FLOAT) halui.spindle-override.scale (in) - pin pour positionner l'échelle des corrections de vitesse de broche possibles
– (S32) halui.spindle-override.counts (in) - comptage depuis un codeur, par exemple pour modifier la correction de vitesse de broche
– (BIT) halui.spindle-override.increase (in) - pin pour augmenter la correction de vitesse de broche (+=scale)
– (BIT) halui.spindle-override.decrease (in) - pin pour diminuer la correction de vitesse de broche (-=scale)
– (BIT) halui.spindle-override.count-enable (in) - Si TRUE, la correction de vitesse broche varie quand le comptage change.

13.2.17 Outil (Tool)

– (U32) halui.tool.number (out) - indique l'outil courant sélectionné
– (FLOAT) halui.tool.length-offset (out) - indique la correction de longueur d'outil appliquée

13.2.18 Programme

– (BIT) halui.program.is-idle (out) - pin de status indiquant qu'aucun programme n'est lancé
– (BIT) halui.program.is-running (out) - pin de status indiquant qu'un programme est lancé
– (BIT) halui.program.is-paused (out) - pin de status indiquant qu'un programme est en pause
– (BIT) halui.program.run (in) - pin de lancement d'un programme
– (BIT) halui.program.pause (in) - pin pour passer un programme en pause
– (BIT) halui.program.resume (in) - pin pour lancer la reprise d'un programme
– (BIT) halui.program.step (in) - pin pour avancer d'une ligne de programme
– (BIT) halui.program.block-delete.is-on (out) -
– (BIT) halui.program.block-delete.off (in) -
– (BIT) halui.program.block-delete.on (in) -
– (BIT) halui.program.optional-stop.is-on (out) -
– (BIT) halui.program.optional-stop.off (in) -
– (BIT) halui.program.optional-stop.on (in) -
– (BIT) halui.program.stop (in) - pin pour stopper un programme

13.2.19 Général

– (BIT) halui.abort - pin pour envoyer un message d'abandon (efface les autres messages d'erreur)

Chapitre 14

pyVCP

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

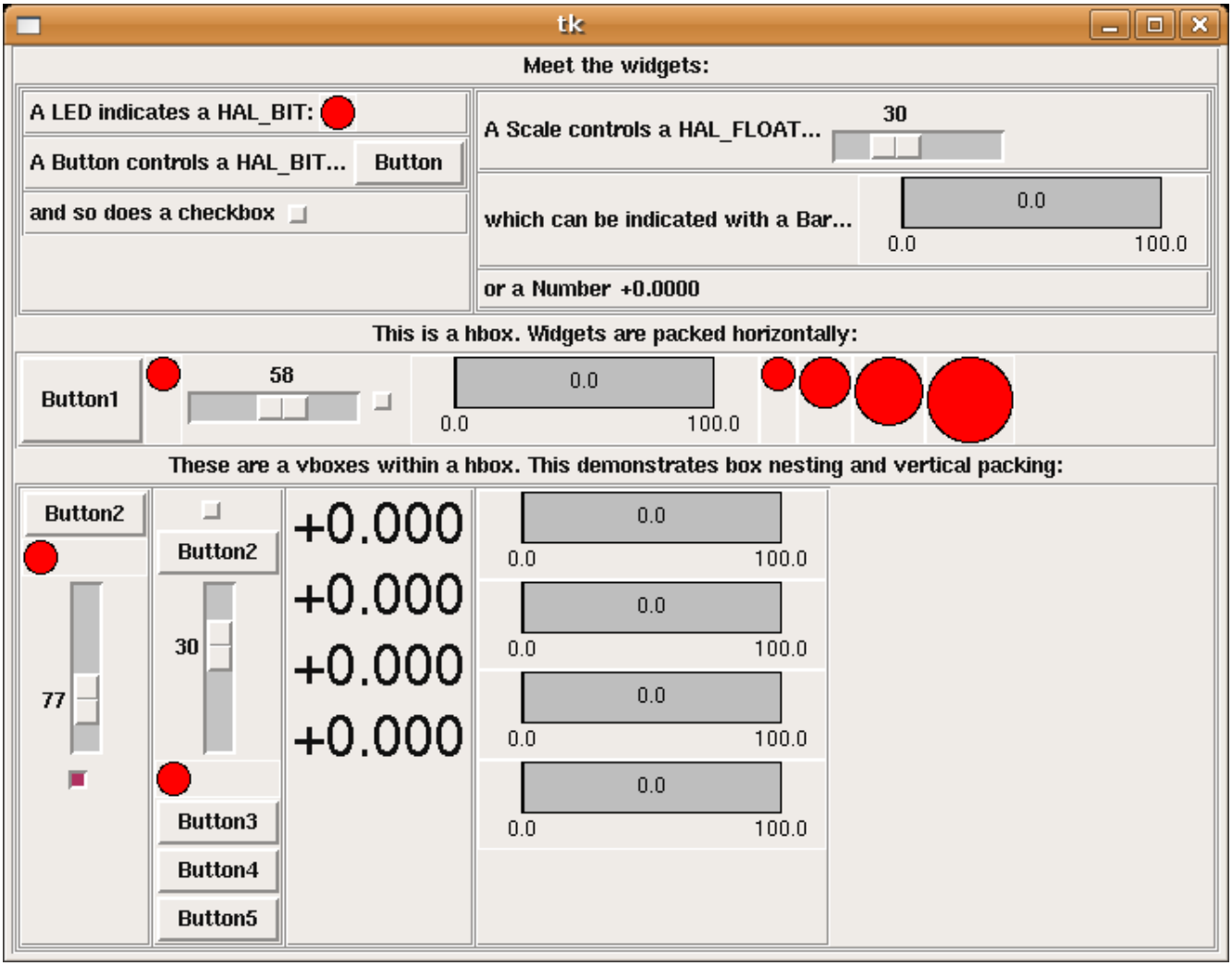
14.1 Introduction

Le panneau de contrôle virtuel pyVCP (python Virtual Control Panel) a été créé pour donner à l'intégrateur la possibilité de personnaliser l'interface graphique 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 virtuels de contrôle peuvent être utilisés pour tester ou monitorer le matériel, les entrées/sorties et 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.

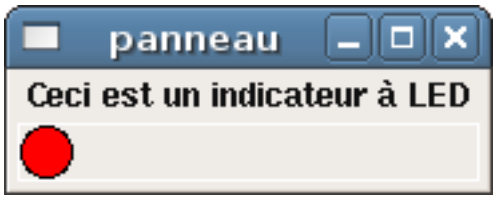
L'image suivante montre plusieurs widgets pyVCP.



14.2 pyVCP

La disposition d'un panneau pyVCP est spécifiée avec 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 :14.5

Un fois que vous avez créé votre panneau, connecter les signaux HAL de la forme à la pin pyVCP se fait avec la commande 'halcmd linksp' habituelle. Si vous débutez avec HAL, le tutoriel de HAL **??** est vivement recommendé.

14.3 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.

14.4 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 réglé la valeur maximum de la barre à 5000 (se reporter à la documentation des widgets, plus loin, 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
```

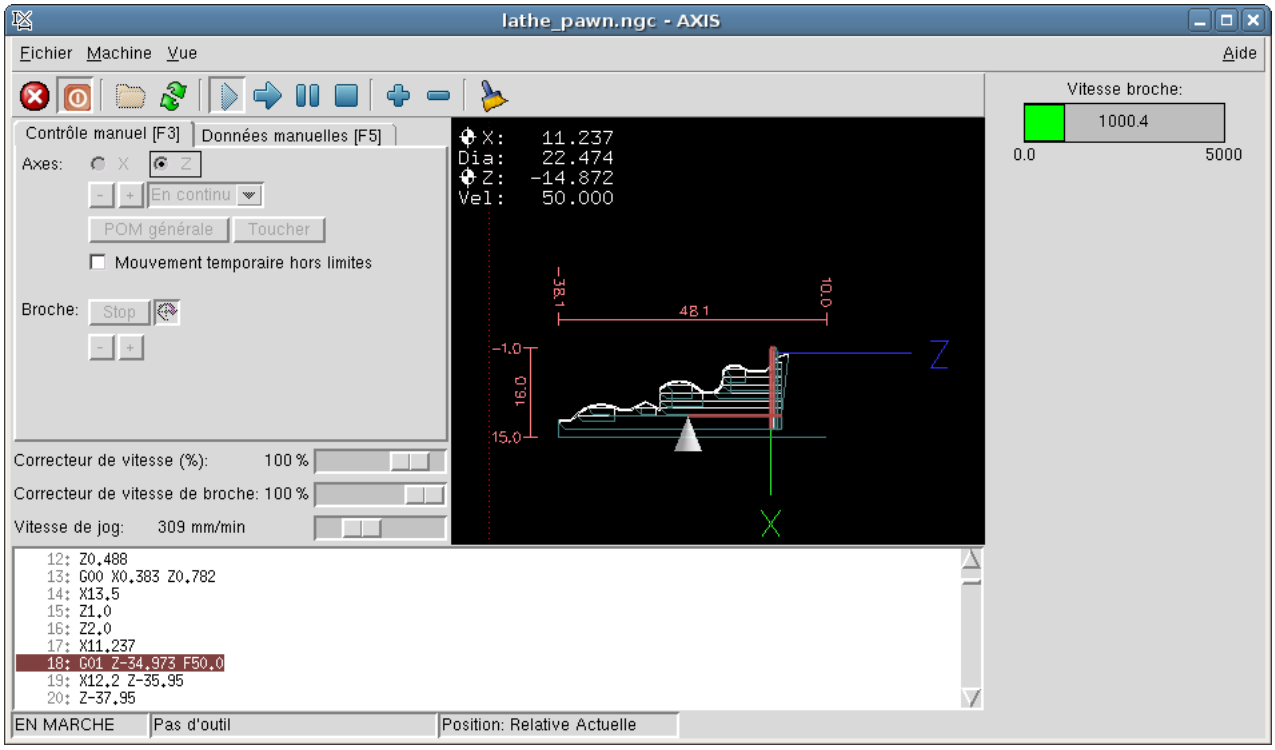
Pour que notre widget affiche réellement la vitesse de la broche “spindle-speed”, il doit être raccordé au signal approprié de HAL. Le fichier .hal qui sera exécuté quand AXIS et pyVCP démarreront doit être spécifié, de la manière suivante, dans la section [HAL] du fichier .ini :

```
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.

14.5 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.

1. Widgets de signalisation, signaux BIT : led, rectled
2. Widgets de contrôle, signaux BIT : button, checkbutton, radiobutton
3. Widgets de signalisation, signaux FLOAT : number, s32, u32, bar, meter
4. Widgets de contrôle, signaux FLOAT : spinbox, scale, jogwheel
5. Widgets d’aide : hbox, vbox, table, label, labelframe

14.5.0.1 Syntaxe

Chaque widget est 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.

14.5.0.2 Notes générales

Ã 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.

14.5.0.3 Commentaires

Pour ajouter un commentaire utiliser la syntaxe de xml.

<!–Mon commentaire–>

14.5.0.4 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.

14.5.0.5 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.

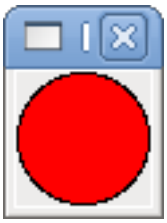
Couleurs les plus courantes (les numéros suivant la couleur indiquent la nuance de la couleur)

– white
– black
– blue and blue1 - 4
– cyan and cyan1 - 4
– green and green1 - 4
– yellow and yellow1 - 4
– red and red1 - 4
– purple and purple1 - 4
– gray and gray0 - 100

14.5.1 LED

Une LED est utilisée pour indiquer l'état d'un signal BIT. La couleur de la LED sera on_color quand le signal BIT est vrai et off_color autrement.

```
<led>
    <halpin>"ma-led"</halpin>
    <size>50</size>
    <on_color>"bleue"</on_color>
    <off_color>"noire"</off_color>
</led>
```



<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"

14.5.2 LED rectangulaire (rectled)

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 cet exemple.
Il affiche également un relief autour de la boîte.

14.5.3 Bouton (button)

Un bouton permet de contrôler une pin 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 le format suivant :

– <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

14.5.3.1 Bouton avec texte (Text Button)

```
<button>
    <halpin>"mon-button"</halpin>
    <text>"OK"</text>
</button>
<button>
    <halpin>"abort-button"</halpin>
    <text>"Abort"</text>
</button>
```

14.5.3.2 Case à cocher (checkbutton)

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

```
<checkbutton>
    <halpin>"ma-case-à-cocher"</halpin>
</checkbutton>
```

Une case non cochée :  et une case cochée : 

14.5.3.3 Bouton radio (radiobutton)

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

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

Noter que dans cet exemple, les pins de HAL seront nommées mon-bouton-radio.un, mon-bouton-radio.deux et mon-bouton-radio.trois. Dans la capture d'écran, la valeur "trois" est sélectionnée.

14.5.4 Affichage d'un nombre (number)

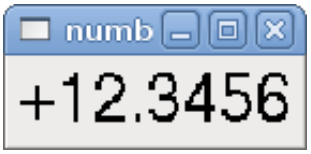
L'affichage d'un nombre peux suivre le format suivant :

- ("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

14.5.4.1 Number

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

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



14.5.4.2 Flottant

Le widget nombre affiche la valeur d'un signal FLOAT.

```
<number>
    <halpin>"mon-nombre"</halpin>
    <font>('Helvetica',50)</font>
    <format>"+4.3f"</format>
</number>
```

<font> est une police de caractères de type Tkinter avec la spécification de sa taille. Noter que sous Ubuntu 6.06 'Helvetica' n'est pas disponible en taille supérieure à 40 ou 50. 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 :

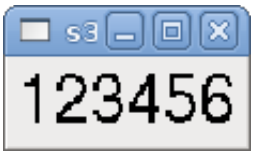
```
<font>('courier 10 pitch',100)</font>
```

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

14.5.4.3 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>
```



14.5.4.4 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>.

14.5.5 Affichage d'images

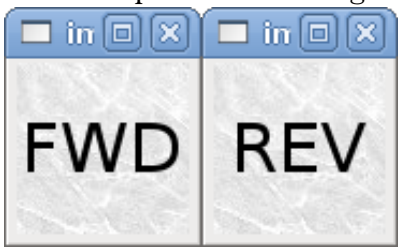
Seul l'affichage d'images au format gif est possible. Toutes les images doivent avoir la même taille. Les images doivent être toutes dans le même répertoire que le fichier ini (ou dans le répertoire courant pour un fonctionnement en ligne de commande avec halrun/halcmd).

14.5.5.1 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.

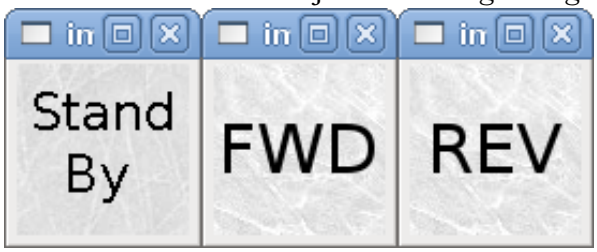


14.5.5.2 Image u32

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.



14.5.6 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.

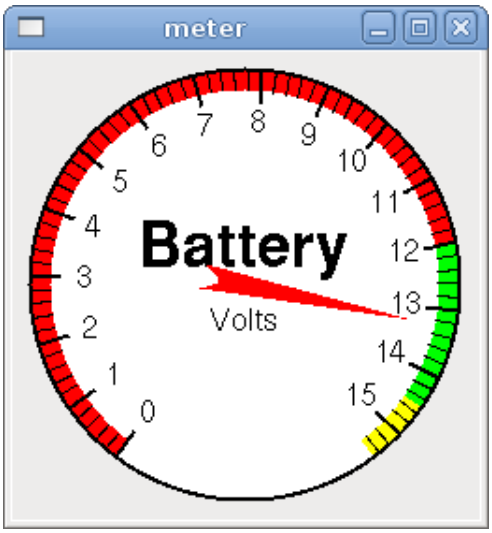
```
<bar>
    <halpin>"ma-bar"</halpin>
    <min_>0</min_>
    <max_>123</max_>
    <bgcolor>"grey"</bgcolor>
    <fillcolor>"red"</fillcolor>
</bar>
```

14.5.7 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>
```



14.5.8 Roue codeuse (spinbox)

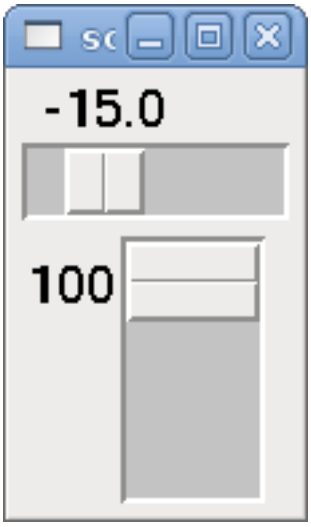
La roue codeuse contrôle une pin FLOAT. La valeur de la pin est augmentée ou diminuée de la valeur de 'resolution', à chaque pression sur une flèche, ou en positionnant la souris sur le nombre puis en tournant la molette de la souris.

```
<spinbox>
    <halpin>"ma-roue-codeuse"</halpin>
    <min_>-12</min_>
    <max_>33</max_>
    <resolution>0.1</resolution>
    <format>"2.3f"</format>
    <font>('Arial',30)</font>
</spinbox>
```

14.5.9 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>
    <halpin>"mon-curseur"</halpin>
    <resolution>0.1</resolution>
    <orient>HORIZONTAL</orient>
    <min_>-33</min_>
    <max_>26</max_>
</scale>
```

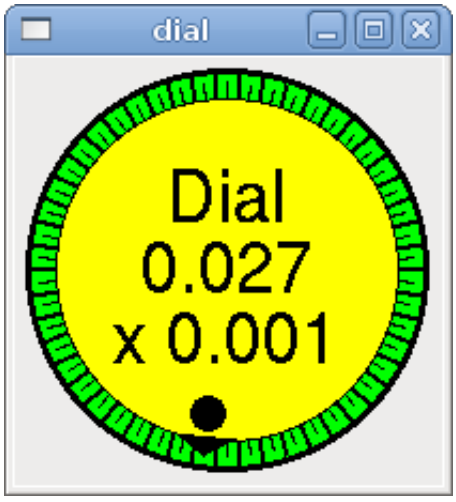


Noter que par défaut c'est min qui est affiché même si il est supérieur à max, à moins que min ne soit négatif.

14.5.10 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).

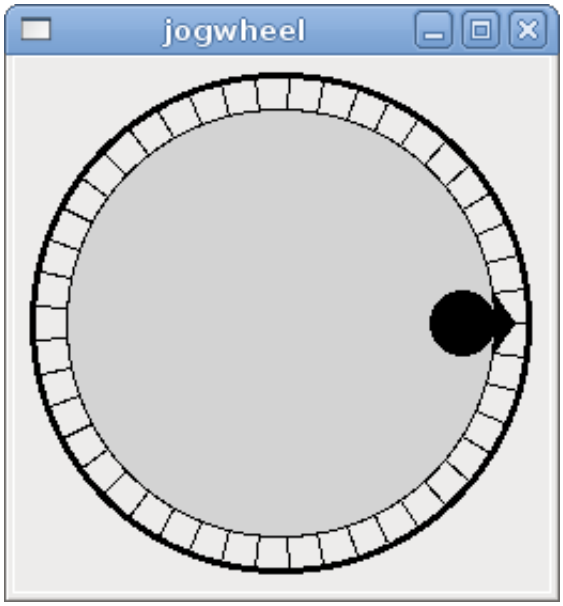
```
<dial>
<size>200</size>
<cpr>100</cpr>
<min_>-15</min_>
<max_>15</max_>
<text>"Dial"</text>
<init>0</init>
<resolution>0.001</resolution>
<halpin>"anaout"</halpin>
<dialcolor>"yellow"</dialcolor>
<edgecolor>"green"</edgecolor>
<dotcolor>"black"</dotcolor>
</dial>
```



14.5.11 Bouton tournant (jogwheel)

Le bouton tournant imite le fonctionnement d'un vrai bouton tournant, 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>"mon-bouton-tournant"</halpin>
    <cpr>45</cpr>
    <size>250</size>
</jogwheel>
```



14.6 Documentation des containers de pyVCP

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

14.6.1 Bordures

Le container bordure est spécifié avec deux tags utilisés ensembles. Le tag <relief> spécifie le type de bordure et le tag <bd> spécifie la largeur de la bordure.

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

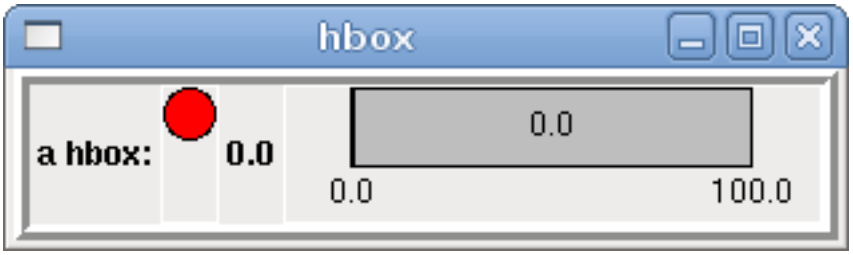
<bd>n</bd> La valeur de "n" fixe la largeur de la bordure.

```
<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>
```

14.6.2 Hbox

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

```
<hbox>
    <label><text>"une hbox :"</text></label>
    <led></led>
    <number></number>
    <bar></bar>
</hbox>
```

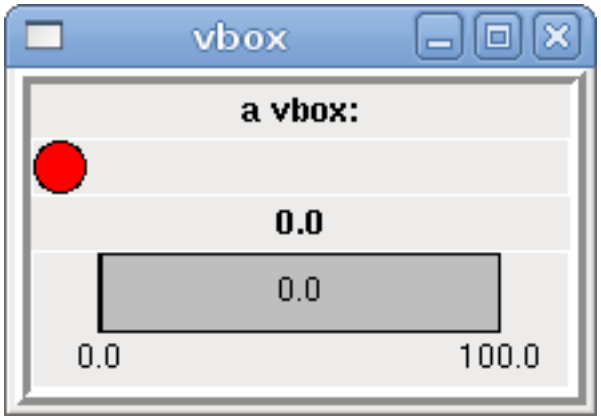


Ã l'intérieur d'une Hbox, vous pouvez utiliser les balises `<boxfill fill=""/>`, `<boxanchor anchor=""/>` et `<boxexpand expand=""/>` pour choisir le comportement des éléments contenus dans la boîte, lors d'un redimensionnement de la fenêtre. Pour des détails sur le comportement de fill, anchor, et expand,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

14.6.3 Vbox

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

```
<vbox>
    <label><text>"une vbox :"</text></label>
    <led></led>
    <number></number>
    <bar></bar>
</vbox>
```

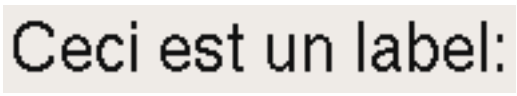


Ã l'intérieur d'une Hbox, vous pouvez utiliser les balises `<boxfill fill=""/>`, `<boxanchor anchor=""/>` et `<boxexpand expand=""/>` pour choisir le comportement des éléments contenus dans la boîte, lors d'un redimensionnement de la fenêtre. Pour des détails sur le comportement de fill, anchor, et expand,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

14.6.4 Label

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

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



14.6.5 Labelframe

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

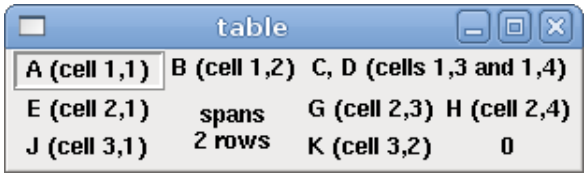
```
<labelframe text="Label : Leds groupées">
  <hbox>
    <led/> <led/></led>
  </hbox>
</labelframe>
```

14.6.6 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 container 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 peut s'étirer sur ses lignes et colonnes flexibles (sticky).

Exemple :

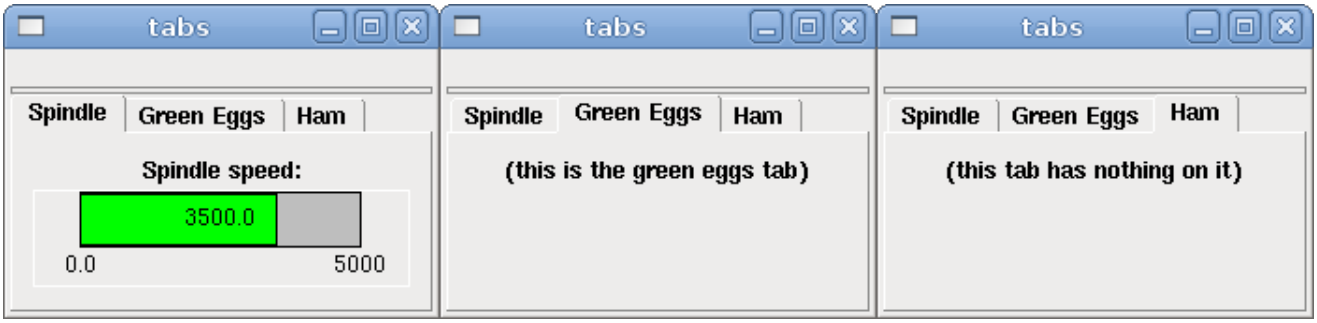
```
<table flexible_rows="[2]" flexible_columns="[1,4]">
   <tablesticky sticky="new"/>
   <tablerow/>
    <label text="A (cell 1,1)"/>
    <label text="B (cell 1,2)"/>
    <tablespan columns="2"/><label text="C, D (cells 1,3 and 1,4)">
   <tablerow/>
    <label text="E (cell 2,1)"/>
    <tablesticky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



14.6.7 Onglets (Tabs)

Une interface à onglets peut économiser énormément d'espace.

```
<tabs>
<names> ["spindle","green eggs"]</names>
</tabs>
<tabs>
<names>["Spindle", "Green Eggs", "Ham"]</names>
<vbox>
<label>
<text>"Spindle speed :"</text>
</label>
<bar>
<halpin>"spindle-speed"</halpin>
<max_>5000</max_>
</bar>
</vbox>
<vbox>
<label>
<text>"(this is the green eggs tab)"</text>
</label>
</vbox>
<vbox>
<label>
<text>"(this tab has nothing on it)"</text>
</label>
</vbox>
</tabs>
```



Chapter 15

VCP Examples

15.1 AXIS

To create a pyVCP panel to use with the AXIS interface that is attached to the right of AXIS you need to do the following basic things.

- Create an .xml file that contains your panel description and put it in your config directory.
- Add the PYVCP entry to the [DISPLAY] section of the ini file with your .xml file name.
- Add the POSTGUI_HALFILE entry to the [HAL] section of the ini file with the name of your postgui hal file name.
- Add the links to HAL pins for your panel in the postgui.hal file to "connect" your pyVCP panel to EMC.

15.2 Floating

To create floating pyVCP panels that can be used with any interface you need to do the following basic things.

- Create an .xml file that contains your panel description and put it in your config directory.
- Add a loadusr line to your .hal file to load each panel.
- Add the links to HAL pins for your panel in the postgui.hal file to "connect" your pyVCP panel to EMC.

The following is an example of a loadusr command to load two pyVCP panels and name each one so the connection names in HAL will be known.

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

The -Wn makes hal "Wait for name" to be loaded before proceeding. The pyvcp -c makes pyVCP name the panel.

The HAL pins from panel1.xml will be named btnpanel.<pin name>

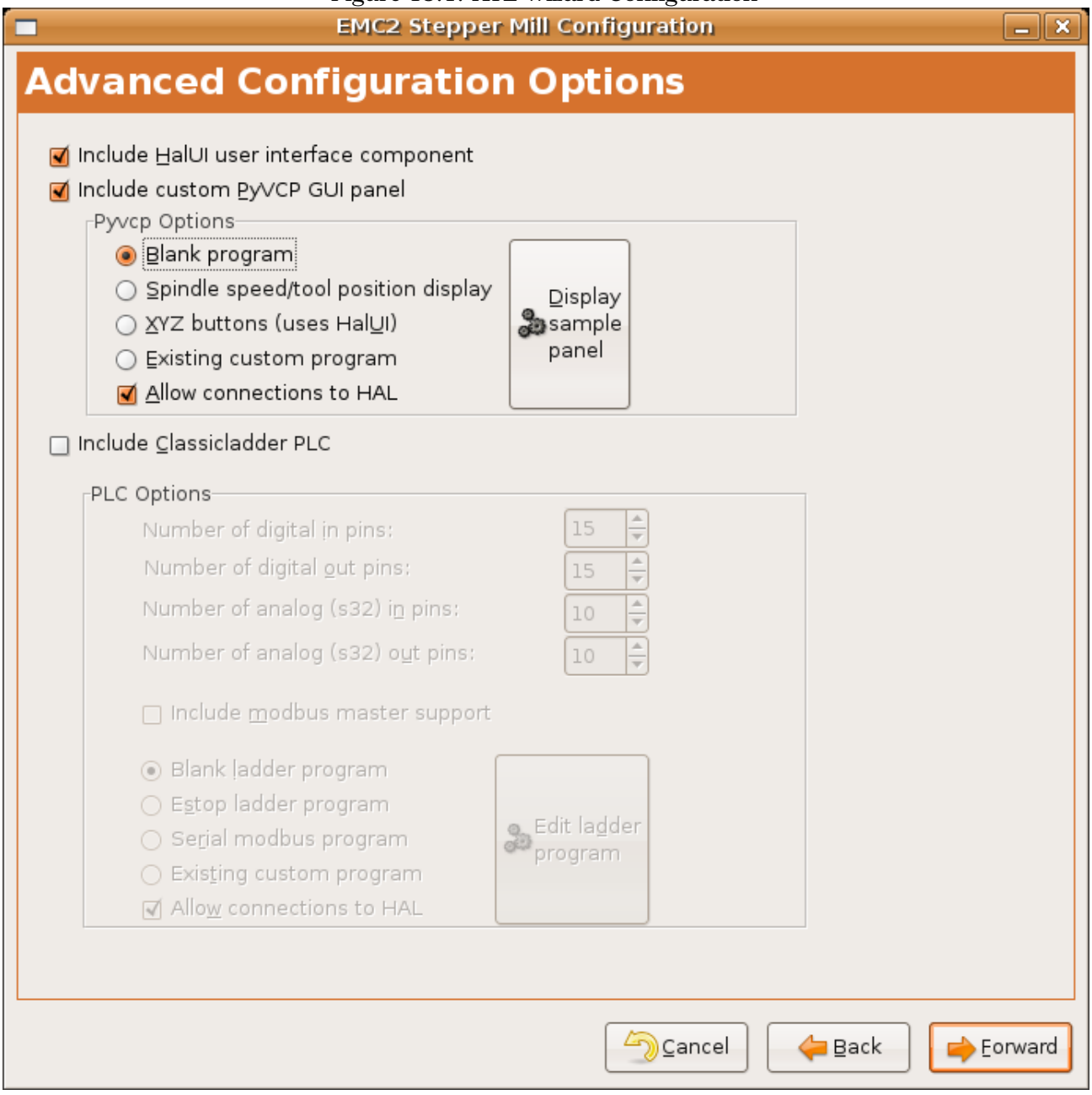
The HAL pins from panel2.xml will be named sppanel.<pin name>

Make sure the loadusr line is before any net's that make use of the pyVCP pins.

15.3 Jog Buttons

In this example we will create a pyVCP panel with jog buttons for X, Y, and Z. This configuration will be built upon a Stepconf Wizard generated configuration. First we run the Stepconf Wizard and configure our machine, then on the Advanced Configuration Options page we make a couple of selections to add a blank pyVCP panel as shown in the following figure. For this example we named the configuration "pyvcp_xyz" on the Basic Machine Information page of the Stepconf Wizard.

Figure 15.1: XYZ Wizard Configuration



The Stepconf Wizard will create several files and place them in the /emc/configs/pyvcp_xyz directory. If you left the create link checked you will have a link to those files on your desktop.

Create the Widgets

Open up the custompanel.xml file by right clicking on it and selecting "open with text editor". Between the <pyvcp></pyvcp> tags we will add the widgets for our panel.

Look in the pyVCP Widgets Reference section of the manual for more detailed information on each widget.

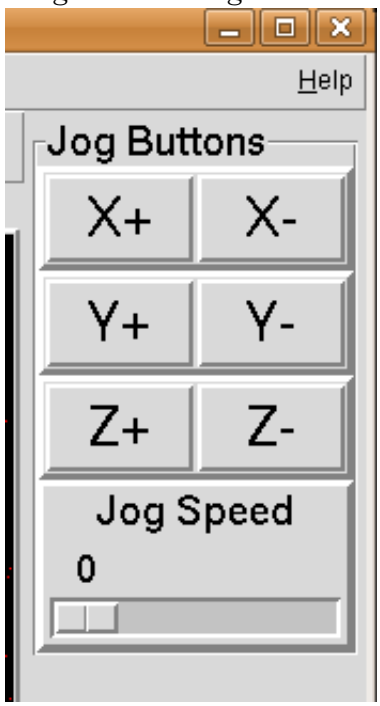
In your custompanel.xml file we will add the description of the widgets.

```
<pyvcp>
<labelframe text="Jog Buttons">
<font>("Helvetica",16)</font>
<!-- the X jog buttons -->
<hbox>
<relief>RAISED</relief>
<bd>3</bd>
<button>
<font>("Helvetica",20)</font>
<width>3</width>
<halpin>"x-plus"</halpin>
<text>"X+"</text>
</button>
<button>
<font>("Helvetica",20)</font>
<width>3</width>
<halpin>"x-minus"</halpin>
<text>"X-"</text>
</button>
</hbox>
<!-- the Y jog buttons -->
<hbox>
<relief>RAISED</relief>
<bd>3</bd>
<button>
<font>("Helvetica",20)</font>
<width>3</width>
<halpin>"y-plus"</halpin>
<text>"Y+"</text>
</button>
<button>
<font>("Helvetica",20)</font>
<width>3</width>
<halpin>"y-minus"</halpin>
<text>"Y-"</text>
</button>
</hbox>
<!-- the Z jog buttons -->
<hbox>
<relief>RAISED</relief>
<bd>3</bd>
<button>
<font>("Helvetica",20)</font>
<width>3</width>
<halpin>"z-plus"</halpin>
<text>"Z+"</text>
```

```
</button>
<button>
<font>("Helvetica",20)</font>
<width>3</width>
<halpin>"z-minus"</halpin>
<text>"Z-"</text>
</button>
</hbox>

<!-- the jog speed slider -->
<vbox>
<relief>RAISED</relief>
<bd>3</bd>
<label>
<text>"Jog Speed"</text>
<font>("Helvetica",16)</font>
</label>
<scale>
<font>("Helvetica",14)</font>
<halpin>"jog-speed"</halpin>
<resolution>1</resolution>
<orient>HORIZONTAL</orient>
<min_>0</min_>
<max_>80</max_>
</scale>
</vbox>

</labelframe>

</pyvcp>
```

After adding the above you now will have a pyVCP panel that looks like the following attached to the right side of AXIS. It looks nice but it does not do anything until you "connect" the buttons to halui. If you get an error when you try and run scroll down to the bottom of the pop up window and usually the error is a spelling or syntax error and it will be there.

Figure 15.2: Jog Buttons



Make Connections

To make the connections needed open up your custom_postgui.hal file and add the following.

```
# connect the X pyVCP buttons
net my-jogxminus halui.jog.0.minus <= pyvcp.x-minus
net my-jogxplus halui.jog.0.plus <= pyvcp.x-plus
# connect the Y pyVCP buttons
net my-jogyminus halui.jog.1.minus <= pyvcp.y-minus
net my-jogyplus halui.jog.1.plus <= pyvcp.y-plus
# connect the Z pyVCP buttons
net my-jogzminus halui.jog.2.minus <= pyvcp.z-minus
net my-jogzplus halui.jog.2.plus <= pyvcp.z-plus
# connect the pyVCP jog speed slider
net my-jogspeed halui.jog-speed <= pyvcp.jog-speed-f
```

After resetting the E-Stop and putting it into jog mode and moving the jog speed slider in the pyVCP panel to a value greater than zero the pyVCP jog buttons should work. You can not jog when running a g code file or while paused or while the MDI tab is selected.

15.4 Port Tester

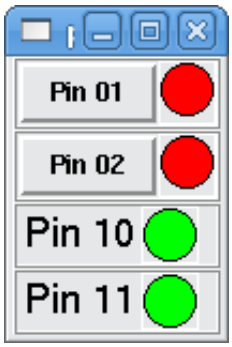
This example shows you how to make a simple parallel port tester using pyVCP and HAL. First create the ptest.xml file with the following code to create the panel description.

```
<!-- Test panel for the parallel port cfg for out -->
<pyvcp>
    <hbox>
        <relief>RIDGE</relief>
        <bd>2</bd>
        <button>
            <halpin>"btn01"</halpin>
            <text>"Pin 01"</text>
        </button>
        <led>
            <halpin>"led-01"</halpin>
            <size>25</size>
            <on_color>"green"</on_color>
            <off_color>"red"</off_color>
       </led>
    </hbox>
    <hbox>
        <relief>RIDGE</relief>
        <bd>2</bd>
        <button>
            <halpin>"btn02"</halpin>
            <text>"Pin 02"</text>
        </button>
        <led>
            <halpin>"led-02"</halpin>
            <size>25</size>
            <on_color>"green"</on_color>
            <off_color>"red"</off_color>
       </led>
    </hbox>
    <hbox>
        <relief>RIDGE</relief>
        <bd>2</bd>
        <label>
            <text>"Pin 10"</text>
            <font>("Helvetica",14)</font>
        </label>
        <led>
            <halpin>"led-10"</halpin>
            <size>25</size>
            <on_color>"green"</on_color>
            <off_color>"red"</off_color>
        </led>
    </hbox>
    <hbox>
        <relief>RIDGE</relief>
        <bd>2</bd>
        <label>
            <text>"Pin 11"</text>
            <font>("Helvetica",14)</font>
        </label>
```

```
            <led>
                <halpin>"led-11"</halpin>
                <size>25</size>
                <on_color>"green"</on_color>
                <off_color>"red"</off_color>
            </led>
        </hbox>
    </pyvcp>
```

This will create the following floating panel which contains a couple of in pins and a couple of out pins.

Figure 15.3: Port Tester Panel



To run the HAL commands that we need to get everything up and running we put the following in our ptest.hal file.

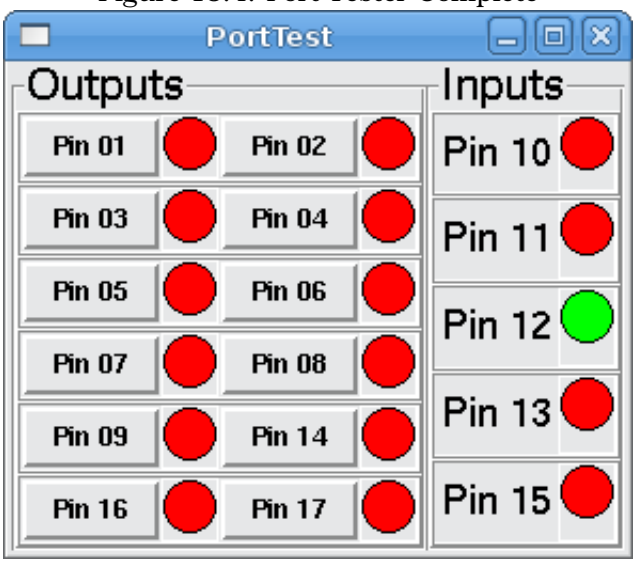
```
loadrt hal_parport cfg="0x378 out"
loadusr -Wn ptest pyvcp -c ptest ptest.xml
loadrt threads name1=porttest period1=1000000
addf parport.0.read porttest
addf parport.0.write porttest
net pin01 ptest.btn01 parport.0.pin-01-out ptest.led-01
net pin02 ptest.btn02 parport.0.pin-02-out ptest.led-02
net pin10 parport.0.pin-10-in ptest.led-10
net pin11 parport.0.pin-11-in ptest.led-11
start
```

To run the HAL file we use the following command from a terminal window.

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

The following figure shows what a complete panel might look like.

Figure 15.4: Port Tester Complete



To add the rest of the parallel port pins just modify the .xml and .hal files.

To show the pins after running the HAL script use the following command at the halcmd prompt:

```
halcmd: show pin
Component Pins:
Owner   Type  Dir         Value  Name
     2  bit   IN          FALSE  parport.0.pin-01-out <== pin01
     2  bit   IN          FALSE  parport.0.pin-02-out <== pin02
     2  bit   IN          FALSE  parport.0.pin-03-out
     2  bit   IN          FALSE  parport.0.pin-04-out
     2  bit   IN          FALSE  parport.0.pin-05-out
     2  bit   IN          FALSE  parport.0.pin-06-out
     2  bit   IN          FALSE  parport.0.pin-07-out
     2  bit   IN          FALSE  parport.0.pin-08-out
     2  bit   IN          FALSE  parport.0.pin-09-out
     2  bit   OUT          TRUE  parport.0.pin-10-in ==> pin10
     2  bit   OUT         FALSE  parport.0.pin-10-in-not
     2  bit   OUT          TRUE  parport.0.pin-11-in ==> pin11
     2  bit   OUT         FALSE  parport.0.pin-11-in-not
     2  bit   OUT          TRUE  parport.0.pin-12-in
     2  bit   OUT         FALSE  parport.0.pin-12-in-not
     2  bit   OUT          TRUE  parport.0.pin-13-in
     2  bit   OUT         FALSE  parport.0.pin-13-in-not
     2  bit   IN          FALSE  parport.0.pin-14-out
     2  bit   OUT          TRUE  parport.0.pin-15-in
     2  bit   OUT         FALSE  parport.0.pin-15-in-not
     2  bit   IN          FALSE  parport.0.pin-16-out
     2  bit   IN          FALSE  parport.0.pin-17-out
     4  bit   OUT         FALSE  ptest.btn01 ==> pin01
     4  bit   OUT         FALSE  ptest.btn02 ==> pin02
     4  bit   IN          FALSE  ptest.led-01 <== pin01
     4  bit   IN          FALSE  ptest.led-02 <== pin02
     4  bit   IN           TRUE  ptest.led-10 <== pin10
     4  bit   IN           TRUE  ptest.led-11 <== pin11
```

This will show you what pins are IN and what pins are OUT as well as any connections.

15.5 GS2 RPM Meter

The following example uses the Automation Direct GS2 VDF driver and displays the RPM and other info in a pyVCP panel. This example is based on the GS2 example in the Hardware Examples section this manual.

The Panel

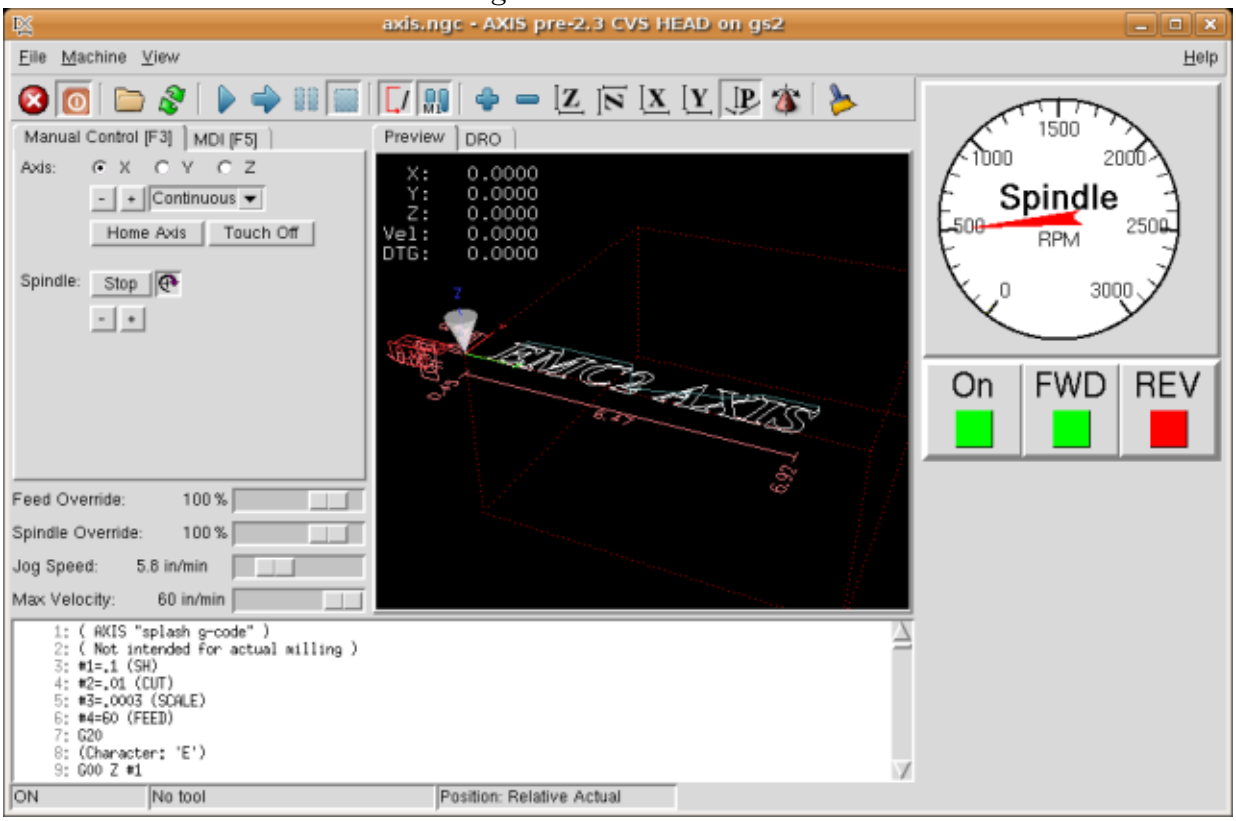
To create the panel we add the following to the .xml file.

```
<pyvcp>
<!– the RPM meter –>
<hbox>
<relief>RAISED</relief>
<bd>3</bd>
<meter>
<halpin>"spindle_rpm"</halpin>
<text>"Spindle"</text>
<subtext>"RPM"</subtext>
<size>200</size>
<min_>0</min_>
<max_>3000</max_>
<majorscale>500</majorscale>
<minorscale>100</minorscale>
<region1>0,10,"yellow"</region1>
</meter>
</hbox>
<!– the On Led –>
<hbox>
<relief>RAISED</relief>
<bd>3</bd>
<vbox>
<relief>RAISED</relief>
<bd>2</bd>
<label>
<text>"On"</text>
<font>("Helvetica",18)</font>
</label>
<width>5</width>
<hbox>
<label width=”2”/> <!– used to center the led –>
<rectled>
<halpin>"on-led"</halpin>
<height>"30"</height>
<width>"30"</width>
<on_color>"green"</on_color>
<off_color>"red"</off_color>
</rectled>
</hbox>
</vbox>
<!– the FWD Led –>
<vbox>
<relief>RAISED</relief>
<bd>2</bd>
```

```
<label>
<text>"FWD"</text>
<font>("Helvetica",18)</font>
<width>5</width>
</label>
<label width="2"/>
<rectled>
<halpin>"fwd-led"</halpin>
<height>"30"</height>
<width>"30"</width>
<on_color>"green"</on_color>
<off_color>"red"</off_color>
</rectled>
</vbox>
<!– the REV Led –>
<vbox>
<relief>RAISED</relief>
<bd>2</bd>
<label>
<text>"REV"</text>
<font>("Helvetica",18)</font>
<width>5</width>
</label>
<label width="2"/>
<rectled>
<halpin>"rev-led"</halpin>
<height>"30"</height>
<width>"30"</width>
<on_color>"red"</on_color>
<off_color>"green"</off_color>
</rectled>
</vbox>
</hbox>
</pyvcp>
```

The above gives us a pyVCP panel that looks like the following.

Figure 15.5: GS2 Panel



The Connections

To make it work we add the following code to the custom_postgui.hal file.

```
# display the rpm based on freq * rpm per hz
loadrt mult2
addf mult2.0 servo-thread
setp mult2.0.in1 28.75
net cypher_speed mult2.0.in0 <= spindle-vfd.frequency-out
net speed_out pyvcp.spindle_rpm <= mult2.0.out

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

# fwd led
net gs2-fwd => pyvcp.fwd-led

# rev led
net running-rev spindle-vfd.spindle-rev => pyvcp.rev-led
```

Some of the lines might need some explanations. The fwd led line uses the signal created in the custom.hal file where as the rev led needs to use the spindle-rev bit. You can't link the spindle-fwd bit twice so you use the signal that it was linked to.

Part IV

Notions de cinématique

Chapter 16

La cinématique dans EMC2

16.1 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 (RS274NGC) 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 d'EMC2 est responsable de la translation entre ces positions et les positions correspondantes de la cinématique de la machine[2].

16.1.1 Les jointures (articulations) par rapport aux axes

Une jointure, 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, jointures d'un bras robotisé). Il peut y avoir n'importe quel nombre de jointures sur une machine. Par exemple, un robot classique dispose de 6 jointures et une fraiseuse classique n'en a que 3.

Sur certaines machines, les jointures sont placées de manière à correspondre aux axes cinématiques (jointure 0 le long de l'axe X, jointure 1 le long de l'axe Y et jointure 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).

16.2 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 jointure (l'actuateur actuel de la machine), est trivial. C'est un simple plan 1:1:

[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 jointures

```
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];
```

Dans l'extrait de code ci-dessus, nous pouvons voir comment le plan est fait: la position X est identique avec la jointure 0, Y avec la jointure 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]

16.3 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 jointures linéaires et rotatives. Ces jointures ne correspondent pas habituellement avec les coordonnées Cartésiennes, cela nécessite une fonction cinématique qui fasse la conversion (en fait 2 fonctions: fonction enă 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).

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 jointures dans ce cas sont les distances entre le mobile et les moteurs de l'appareil (nommées AD et BD sur la figure 16.1).

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 jointures en (AD, BD) en coordonnées Cartésiennes (Dx, Dy) et vice-versa.

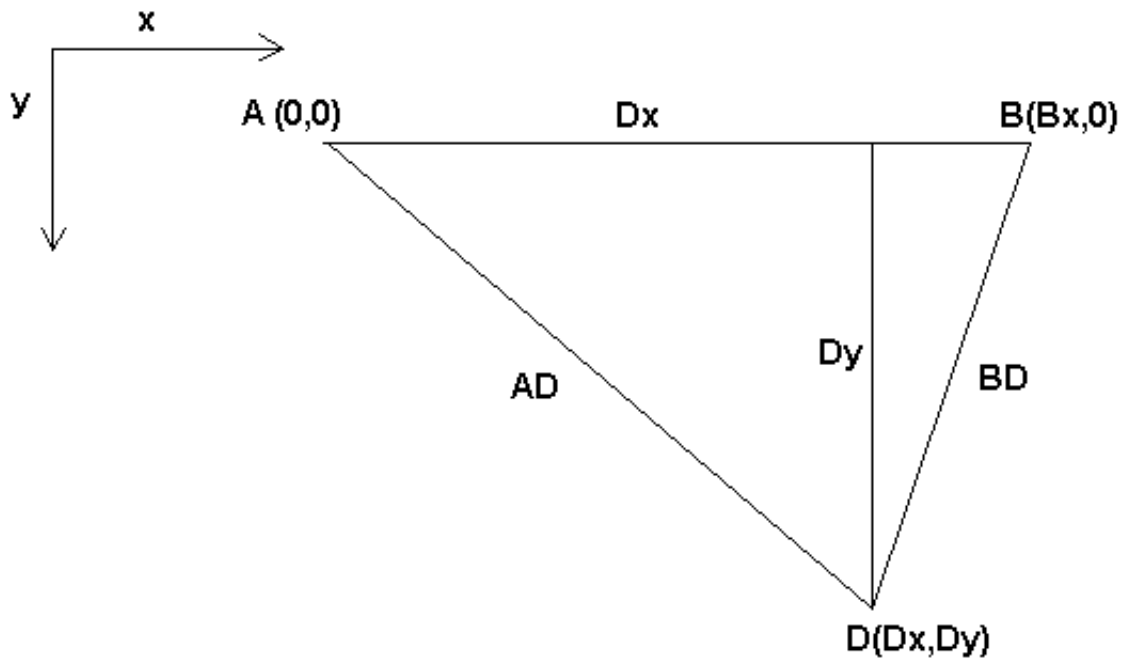
16.3.1 Transformation avant

Pour effectuer la transformation de l'espace jointure 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).

[3] Si la machine (par exemple un tour) est montée avec seulement les axes X, Z et A et que le fichier d'init d'EMC2 contient uniquement la définition de ces 3 jointures, 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 EMC2 il suffit de définir tous les axes (XYZA), EMC2 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.

Figure 16.1: Définir un bipode



Nous pouvons voir aisément que $AD^2 = x^2 + y^2$, de même que $BD^2 = (Bx - x)^2 + y^2$.

Si nous soustrayons l'un de l'autre nous aurons:

$$AD^2 - BD^2 = x^2 + y^2 - x^2 + 2 * x * Bx - Bx^2 - y^2$$

et par conséquent:

$$x = \frac{AD^2 - BD^2 + Bx^2}{2 * Bx}$$

De là nous calculons:

$$y = \sqrt{AD^2 - x^2}$$

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 coordonnées Cartésienne pour la position de cette jointure, 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;
```

16.3.2 Transformation inverse

La cinématique inverse est beaucoup plus simple dans notre exemple, de sorte que nous pouvons l'écrire directement:

$$AD = \sqrt{x^2 + y^2}$$

$$BD = \sqrt{(Bx - x)^2 + y^2}$$

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;
```

16.4 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_FL *fflags, KINEMATICS_INVERSE_FLAGS *iflags)`

  Implémente la fonction cinématique avant, comme décrite dans la section 16.3.1.

- `int kinematicsInverse(const EmcPose * world, double *joints, const KINEMATICS_INVERSE_ *iflags, KINEMATICS_FORWARD_FLAGS *fflags)`

  Implémente la fonction cinématique inverse, comme décrite dans la section 16.3.2.

- `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_FLAGS *iflags)`

  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 installé par "`comp`". Voir la manpage "`comp(1)`" pour d'autres informations.

Part V

Les réglages

Chapter 17

Réglages des pas à pas

17.1 Obtenir le meilleur pilotage logiciel possible

Faire générer les impulsions de pas au logiciel présente un gros avantage, c'est gratuit. Comme chaque PC est équipé d'un port parallèle, il est capable de sortir sur celui-ci, les impulsions générées par le logiciel. Toutefois, cette génération d'impulsions logicielle présente aussi certains désavantages:

- La fréquence maximum des impulsions est limitée.
- Les impulsions générées ne sont pas de très bonne qualité 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.

17.1.1 Effectuer un test de latence

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 Pentium II répondant à chaque interruption et toutes les 10 microsecondes pourra donner de meilleurs résultats qu'un rapide P4 HT.

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.

NE PAS ESSAYER DE LANCER EMC2 PENDANT QUE LE TEST EST EN COURS

Sous Ubuntu Dapper, vous pouvez lancer le test en ouvrant une console et en faisant:

```
sudo mkdir /dev/rtf;
sudo mknod /dev/rtf/3 c 150 3;
sudo mknod /dev/rtf3 c 150 3;
cd /usr/realtime*/testsuite/kern/latency; ./run
```

et vous devriez voir quelques choses comme:

```
ubuntu:/usr/realtime-2.6.12-magma/testsuite/kern/latency$ ./run
*
*
* Type ^C to stop this application. (Tapez CTRL-C pour arrêter cette application)
*
*
## RTAI latency calibration tool ##
# period = 100000 (ns)
# avrgtime = 1 (s)
# do not use the FPU
# start the timer
# timer_mode is oneshot
RTAI Testsuite - KERNEL latency (all data in nanoseconds)
RTH| lat min| ovl min| lat avg| lat max| ovl max| overruns
RTD|   -1571|   -1571|    1622|    8446|    8446|       0
RTD|   -1558|   -1571|    1607|    7704|    8446|       0
RTD|   -1568|   -1571|    1640|    7359|    8446|       0
RTD|   -1568|   -1571|    1653|    7594|    8446|       0
RTD|   -1568|   -1571|    1640|   10636|   10636|       0
RTD|   -1568|   -1571|    1640|   10636|   10636|       0
```

Pendant la durée du test, vous devez "abuser" de l'ordinateur. Déplacez les fenêtres tout autour de l'écran. Surfez sur le web. Déplacez quelques longs fichiers sur le disque. Jouez de la musique. Lancez un programme OpenGL comme glxgears. L'idée étant de pousser le PC au bout de ses limites pendant que le test de latence observe pour trouver les cas les moins bons.

L'avant dernière colonne marquée “ovl max” est la plus importante. Notez sa valeur, vous en aurez besoin par la suite. Elle contient les mesures de latence les plus mauvaises durant tout le test. Dans l'exemple ci-dessus, la valeur est de 10636 nanosecondes, soit 10.6 microsecondes, ce qui est excellent. Toutefois, le test de l'exemple n'a duré que quelques secondes (une ligne s'affiche à chaque seconde). Vous devez lancer le test pendant plusieurs minutes, parfois le plus mauvais temps n'apparaît pas vraiment, ou il apparaît quand vous faites une action particulière. J'ai une carte mère Intel qui marche très bien la plupart du temps, mais chaque 64 secondes arrive une valeur très mauvaise de 300uS. Par chance, ce cas a trouvé une solution, voyez ici: Le Wiky de Linuxcnc http://wiki.linuxcnc.org/cgi-bin/emcinfo.pl?FixingSMIIssues.

Que signifient les résultats du tableau, à quels résultats s'attendre? Si la valeur de votre “ovl max” est inférieure à 15-20 microsecondes (15000-20000 nanosecondes), l'ordinateur donnera d'excellents résultats. Si la latence maximale est près de 30-50 microsecondes, vous aurez de bons résultats, mais la fréquence maximale des pas risque d'être décevante, spécialement si vous utilisez des micropas ou si vous avec une vis avec un pas fin. Si la latence est de 100uS ou plus (100,000 nanosecondes), alors le PC n'est pas utilisable pour générer des trains d'impulsions par logiciel. Des chiffres suppérieurs à 1 milliseconde (1,000,000 nanosecondes) signifie que ce PC ne convient pas du tout pour EMC2, que vous pensiez générer les pas par logiciel ou non.

Notez que si vous obtenez de mauvais résultats, c'est peut être améliorable. Par exemple, un PC qui avait une latence très mauvaise (plusieurs millisecondes) en utilisant la carte graphique intégrée a vu le problème résolu par une carte graphique Matrox à 5$. EMC ne nécessite pas du matériel de pointe.

17.1.2 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 uS min (Step on falling edge)
Step Pulse “1” Time: 4.5 uS min
Direction Setup: 1 uS min (20 uS min hold time after Step edge)

Les spécifications du Gecko G203V indiquent:
Step Frequency: 0 to 333 kHz
Step Pulse “0” Time: 2.0 uS min (Step on rising edge)
Step Pulse “1” Time: 1.0 uS min
Direction Setup:

200 nS (0.2uS) before step pulse rising edge
200 nS (0.2uS) 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 2.0uS
Minimum STEP pulse low time 1.0uS
Step happens on rising edge

Notez les valeurs que vous trouvez, vous en aurez besoin pour la prochaine étape.

17.1.3 Choisir la valeur de BASE_PERIOD

BASE_PERIOD est l’horloge de votre EMC2. 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 0.5uS et à l’état haut pendant 4.5uS, il a besoin que la broche de direction soit stable 1uS avant le front descendant et qu’elle reste stable pendant 20uS après le front descendant. La plus longue durée est de 20uS, c’est le temps de maintien. Une approche simple consisterait à fixer la période à 20uS. Ce qui signifierait que tous les changements d’état des lignes STEP et DIR serait espacés de 20uS. C’est tout bon, non?

Faux! Si la latence était de zéro, et que tous les fronts soient espacés de 20uS, tout irait bien. Mais tous les ordinateurs ont une latence. Si l’ordinateur a 11uS de latence, celà signifie que, ce que l’ordinateur exécute aura parfois un retard de 11uS et la fois suivante pourra être juste à l’heure, le délai entre le premier et le second sera seulement de 9uS. Si le premier génére l’impulsion de pas et le second change la broche de direction, le timing de 20uS 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 20uS et que votre test de latence vous avait donné une latence maximum de 11uS, alors si vous définissez BASE_PERIOD à 20+11 = 31uS (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 31uS, il faut 2x31 = 62uS 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 (0.2uS). Le temps le plus long est de 2uS. Si vous avez 11uS de latence, alors vous pouvez définir BASE_PERIOD aussi bas que 11+2 = 13uS. Se débarrasser du long temps de maintien de 20uS aide vraiment. Avec une période de 13uS, un pas complet ne dure que 26uS = 2x13 et la fréquence maximum est de 38461 pas par seconde!

Mais ne commencez pas à célébrer celà. Notez que 13uS est une période très courte. Si vous essayez d'exécuter le générateur de pas toutes les 13uS, 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 25uS, vous devez commencer à 25uS ou plus, lancer EMC 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 25uS, en essayant descendre à 13uS, vous trouvez que c'est autour de 16uS que se situe la limite la plus basse et qu'en dessous l'ordinateur ne répond plus très bien. Alors, vous utilisez 16uS. Avec une période à 16uS et une latence à 11uS, le temps de sortie le plus court sera de 16-11 = 5uS. La carte demande seulement 2uS, 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 16uS alors qu'un pas prend 32uS. Il fonctionnera à 31250 pas par seconde, ce qui n'est pas mal.

17.1.4 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 16uS ce qui nous a donné une fréquence de pas de 31250 pas par seconde maximum. Alors que la Gecko a été bloquée à 31uS 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 20uS. Ca plus la latence de 11uS nous oblige à utiliser une période longue de 31uS. Mais le générateur de pas logiciel d'EMC2 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 20uS demandé, alors le temps le plus long suivant sera de 4.5uS. Ajoutez les 11uS de latence à ces 4.5uS, et vous obtenez une période minimale de 15.5uS. Lorsque vous essayez 15.5uS, vous trouvez que l'ordinateur est très lent, donc vous régler sur 16uS. Si nous laissons "dirhold" à 1 (par défaut), alors le temps minimum entre le pas et la direction est de 16uS moins la période de latence de 11uS = 5uS, ce qui n'est pas suffisant. Nous avons besoin de 15 autres uS, puisque la période est de 16uS, 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 = 21uS 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 11uS, alors la combinaison d'une période de base de 16uS 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 32uS 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 11uS utilisé dans cet exemple est très bon. Si vous travaillez par le biais de ces exemples avec des latences plus grandes, comme 20 ou 25uS, 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.

17.1.5 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 (StepTimingCalculator http://wiki.linuxcnc.org/uploads/StepTimingCalculator.ods). 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.

Chapter 18

Réglages d'une boucle PID

18.1 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 industriel.

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 process tandant 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 focntion 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).

18.1.1 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 souhaîtée, 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, compostion 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.

18.1.2 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.

18.1.2.0.0.1 Action Proportionelle 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 quantitée 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.

18.1.2.0.0.2 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'enléve 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.

18.1.2.0.0.3 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.

18.1.3 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 àăun 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.

18.1.3.0.0.4 Méthode simple Si le système doit rester en production, une méthode de réglage consiste à mettre les valeurs I et D àă zéro. Augmenter ensuite le gain P jusqu'àăce que la sortie de la boucle oscille. Puis, augmenter le gain I jusqu'àăce que cesse l'oscillation. Enfin, augmenter le gain D jusqu'àă 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

18.1.3.0.0.5 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 acroîssement 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	$.45K_c$	$1.2/P_c$	
PID	$.6K_c$	$2/P_c$	$P \times P_c/8$

Part VI

La logique Ladder

Chapter 19

La programmation en Ladder

19.1 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.

19.2 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.

Chapter 20

Classic Ladder

20.1 Introduction

Classic Ladder is a free implementation of a ladder interpreter, released under the LGPL. It has been written by Marc Le Douarain.

He describes the beginning of the project on his website Original project homepage:

> "I decided to program a ladder language only for test purposes at the start, in February 2001. It was planned, that I would have to participate to a new product after leaving the enterprise in which I was working at that time. And I was thinking that to have a ladder language in those products could be a nice option to considerate. And so I started to code the first lines for calculating a rung with minimal elements and displaying dynamically it under Gtk, to see if my first idea to realize all this works.
>
> And as quickly I've found that it advanced quite well, I've continued with more complex elements : timer, multiples rungs, etc...
>
> Voila, here is this work... and more : I've continued to add features since then."

Classic Ladder has been adapted to work with EMC2's HAL, and is currently being distributed along with EMC2. If there are issues/problems/bugs please report them to the Enhanced Machine Controller project.

20.2 Ladder Concepts

Classic Ladder is a type of programming language originally implemented on industrial PLC's (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. 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. ALL the inputs are read, ALL the logic is figured out, then ALL the outputs are set. This can presents a problem in certain circumstance if the output of one line feeds the input of another. 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.

Classic Ladder version 7.124 has been adapted for EMC 2.3 This document describes that version.

20.3 Languages

The most common language used when working with Classic Ladder is 'ladder'. Classic Ladder also supports Sequential Function Chart (Grafcet).

20.4 Components

There are 2 components to Classic Ladder.

- The real time module classicladder_rt
- The user space module (including a GUI) classicladder

20.4.1 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.

20.4.2 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 effects 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 then one microsecond 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.

20.4.3 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 20.1: 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

20.5 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 with out 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. The -w tells HAL not to close down the HAL environment until Classic Ladder is finished.

```
loadusr -w classicladder
```

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...

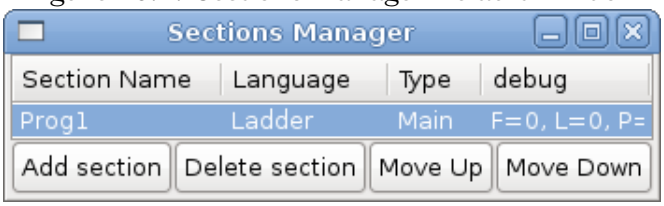
20.6 Classic Ladder GUI

If you load Classic Ladder with the GUI it will display two windows: section display, and section manager.

20.6.1 Sections Manager

When you first start up Classic Ladder you get an empty Sections Manager window.

Figure 20.1: Sections Manager Default Window

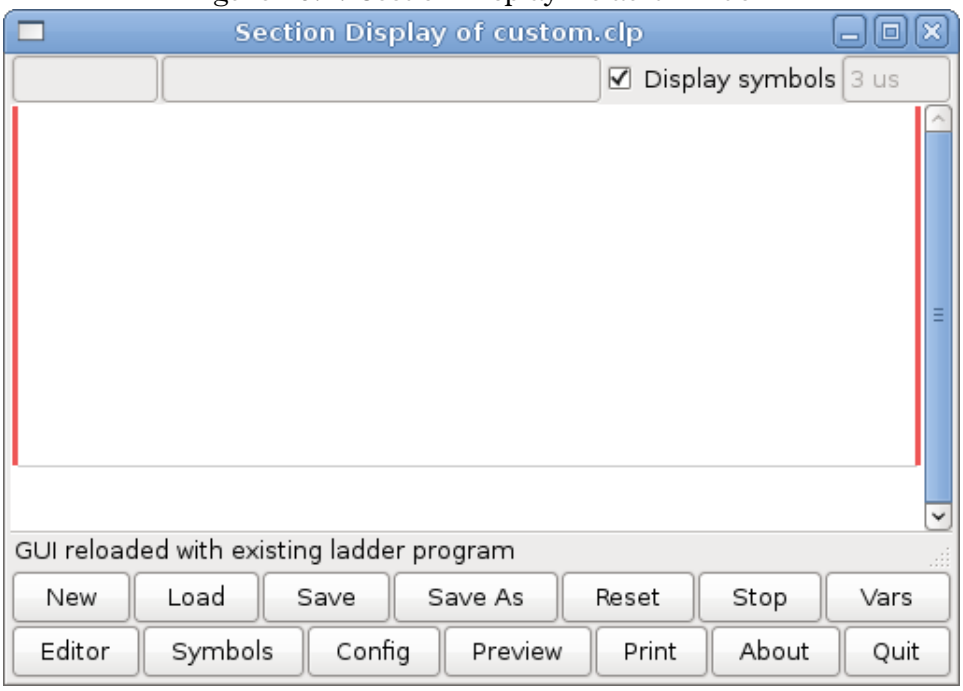


This window allows you to name, create or delete sections and choose what language that section uses. This is also how you name a subroutine for call coils.

20.6.2 Section Display

When you first start up Classic Ladder you get an empty Section Display window.

Figure 20.2: 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 back ground.

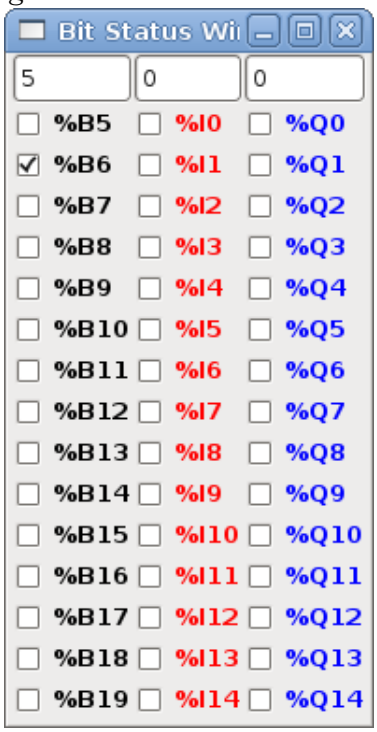
The check box at the top right allows you to select whether variable names or symbol names are displayed

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.

20.6.3 The Variable Windows

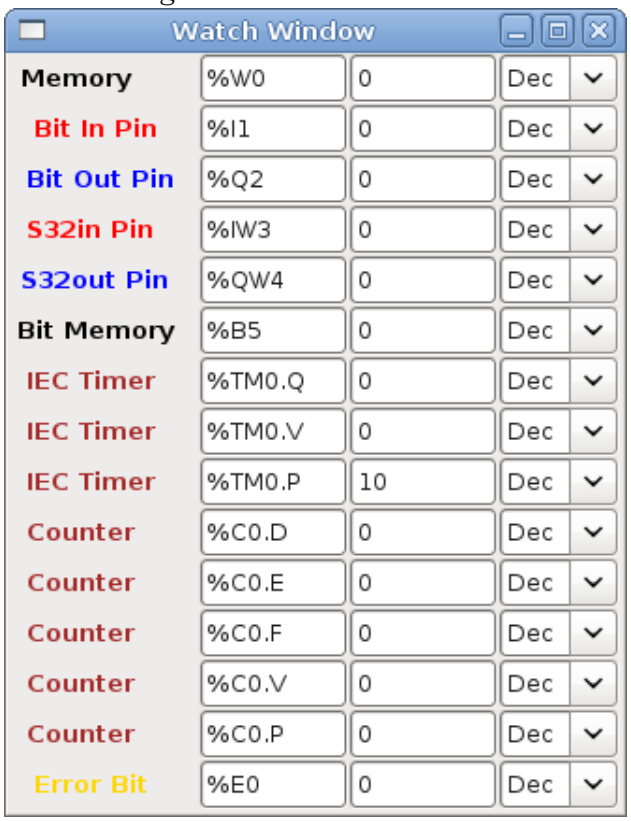
This are two variable windows: bool and signed integer. the vars button is in the section display window, toggle the Vars button to display one, the other, both, then none of the windows.

Figure 20.3: Bit Status Window



The Bool window displays some of all the bool (on/off) variable data . Notice all variable start with the % sign. The %I variable represents 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 variable will be displayed in each column. For instance if there were 30 %B variable and you entered 5 at the top of the column, variable %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.

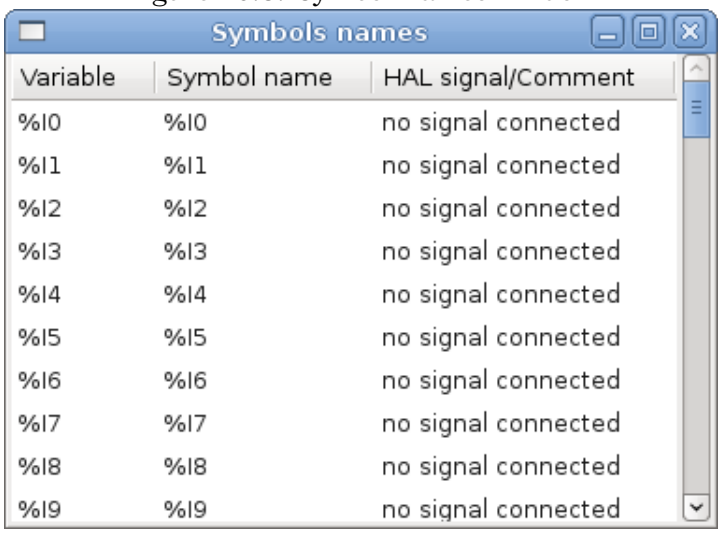
Figure 20.4: Watch Window



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 the the section display window, symbol names will be displayed. To change the variable displayed type the variable number eg. %W2 (if display symbols check box is not checked) or symbol name (if the display symbols checkbox is checked) over an existing variable number/name and press the Enter Key.

20.6.4 Symbol Window

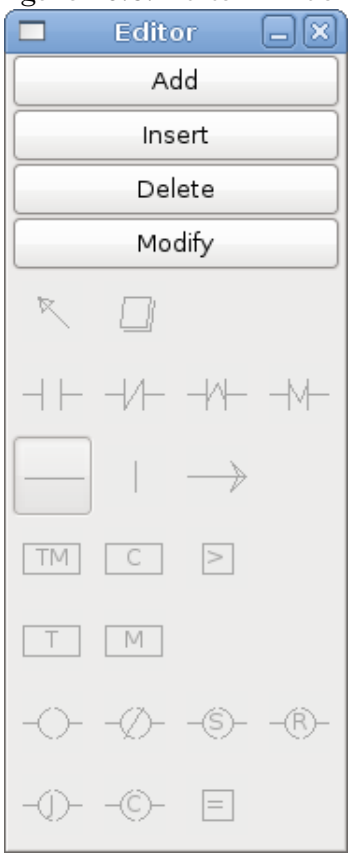
Figure 20.5: Symbol Names window



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 there of. symbol names should be kept short to display better. keep in mind that you can display the longer HAL signal name of %I, %Q and %W variable by clicking on them in the section window. Between the two on should be able to keep track of what the ladder program is connected to!

20.6.5 The Editor window

Figure 20.6: Editor Window



Starting from the top left image:

1. Object Selector, Eraser
2. N.O. Input, N.C. Input, Rising Edge Input , Falling Edge Input
3. Horizontal Connection, Vertical Connection , Long Horizontal Connection
4. Timer IEC Block, Counter Block, Compare Variable
5. Old Timer Block, Old Monostable Block (These have been replaced by the IEC Timer)
6. COILS - N.O. Output, N.C. Output, Set Output, Reset Output
7. Jump Coil, Call Coil, Variable Assignment

A short description of each of the buttons:

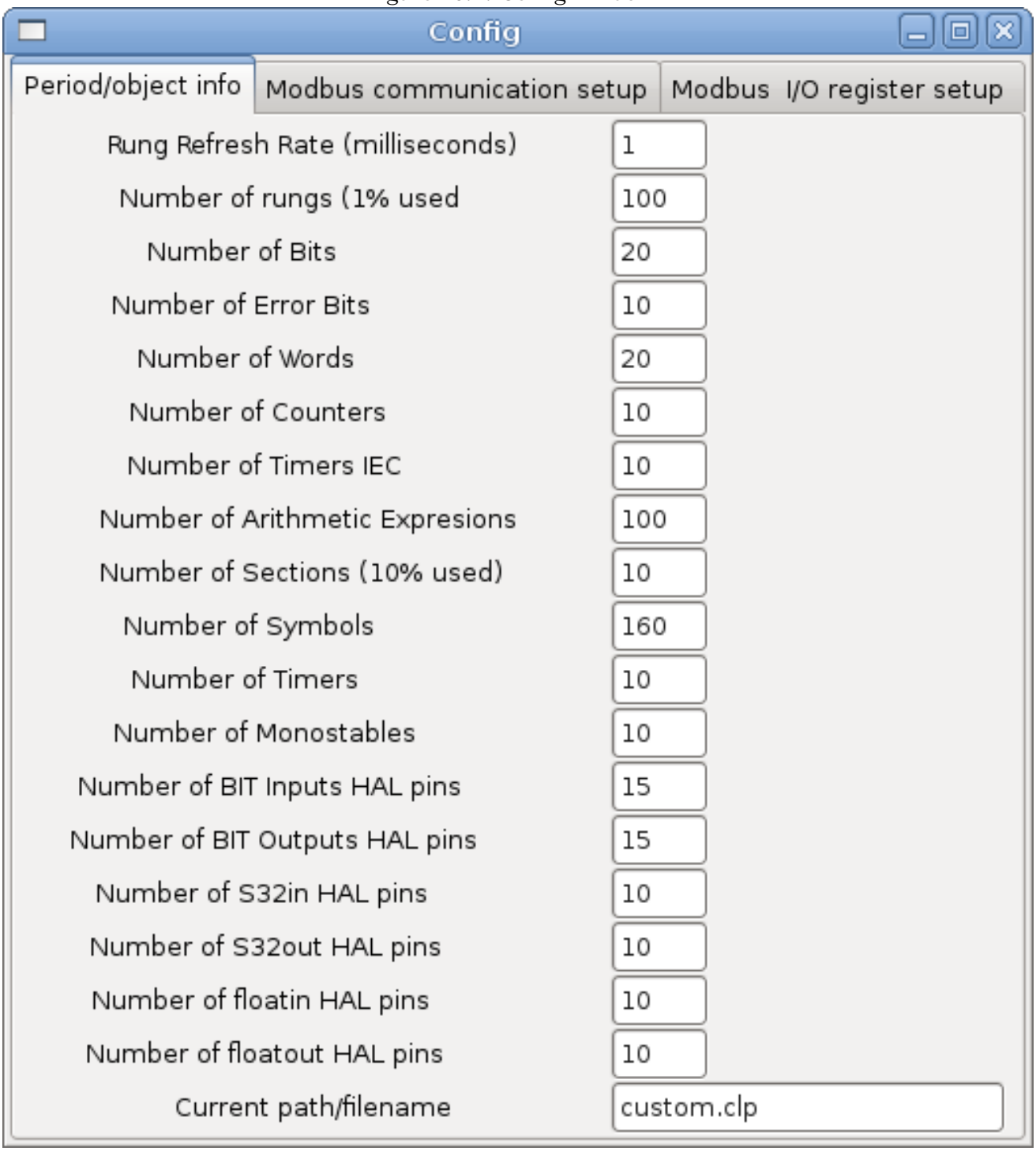
- The SELECTOR ARROW button allows you to select existing objects and modify the information.
- The ERASER erases an object.
- The N.O. CONTACT is a normally open contact. It can be an external HAL-pin (%I) input contact, an internal-bit coil (%B) contact or a external coil (%Q) contact. The Hal-pin 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).

- The N.C. CONTACT is 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.
- The RISING-EDGE CONTACT is a contact that is closed when the HAL-pin goes from False to true, or the coil from not-active to active.
- The FALLING-EDGE CONTACT is a contact that is closed when the HAL-pin goes from true to false or the coil from active to not.
- The HORIZONTAL CONNECTION connects the 'signal' to objects horizontally.
- The VERTICAL CONNECTION connects the 'signal' to objects vertically.
- The HORIZONTAL-RUNNING CONNECTION is a quick way to connect a long run of 'signal wire' horizontally.
- The IEC TIMER replaces the TIMER and the MONSTABLE.
- The TIMER is a Timer Module.
- The MONOSTABLE is monostable module (one-shot)
- The COUNTER is a counter module.
- The COMPARE button allows you 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.
- The VARIABLE ASSIGNMENT button allows 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.

20.6.6 Config Window

The config window shows the current project status and has the Modbus setup tabs.

Figure 20.7: Config Window



20.7 Ladder objects

20.7.1 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). Eg. 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.

20.7.2 IEC TIMERS

Represent new count down timers! IEC Timers replace Timers and Monostables.

IEC Timers have 2 contacts.

- I = input
- Q = output

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)

20.7.3 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)

20.7.4 MONOSTABLES

Represent are one-shot timers. This is deprecated and replaced by IEC Timers.

Monostables have 2 contacts I and R.

- I (input) -input -will start the mono timer running.
- R (output) -running -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)

20.7.5 COUNTERS

- Represent up/down counters.

- There are 7 contacts:

  - R (input) -reset -will reset the count to 0.
  - P (input) -preset -will set the count to the preset number assigned from the edit menu.
  - U (input) -up count -will add one to the count.
  - D (input) -down count -will subtract one from the count.
  - E (output) -under flow -will be true when the count rolls over from 0 to 9999.
  - D (output) -done -will be true when the count equals the preset.
  - F (output) -overflow -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 rather).

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)

20.7.6 COMPARE

For arithmetic comparison. eg 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 +,-,*,/,=,<,>,<=,>=,(,),^ (exponent),% (modulus),& (and),| (or),! (not). - You can also use math functions. They are ABS (absolute), MOY (french for average) ,AVG (average). eg 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 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 use to.

20.7.7 VARIABLE ASSIGNMENT

For variable assignment. eg 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 and 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:

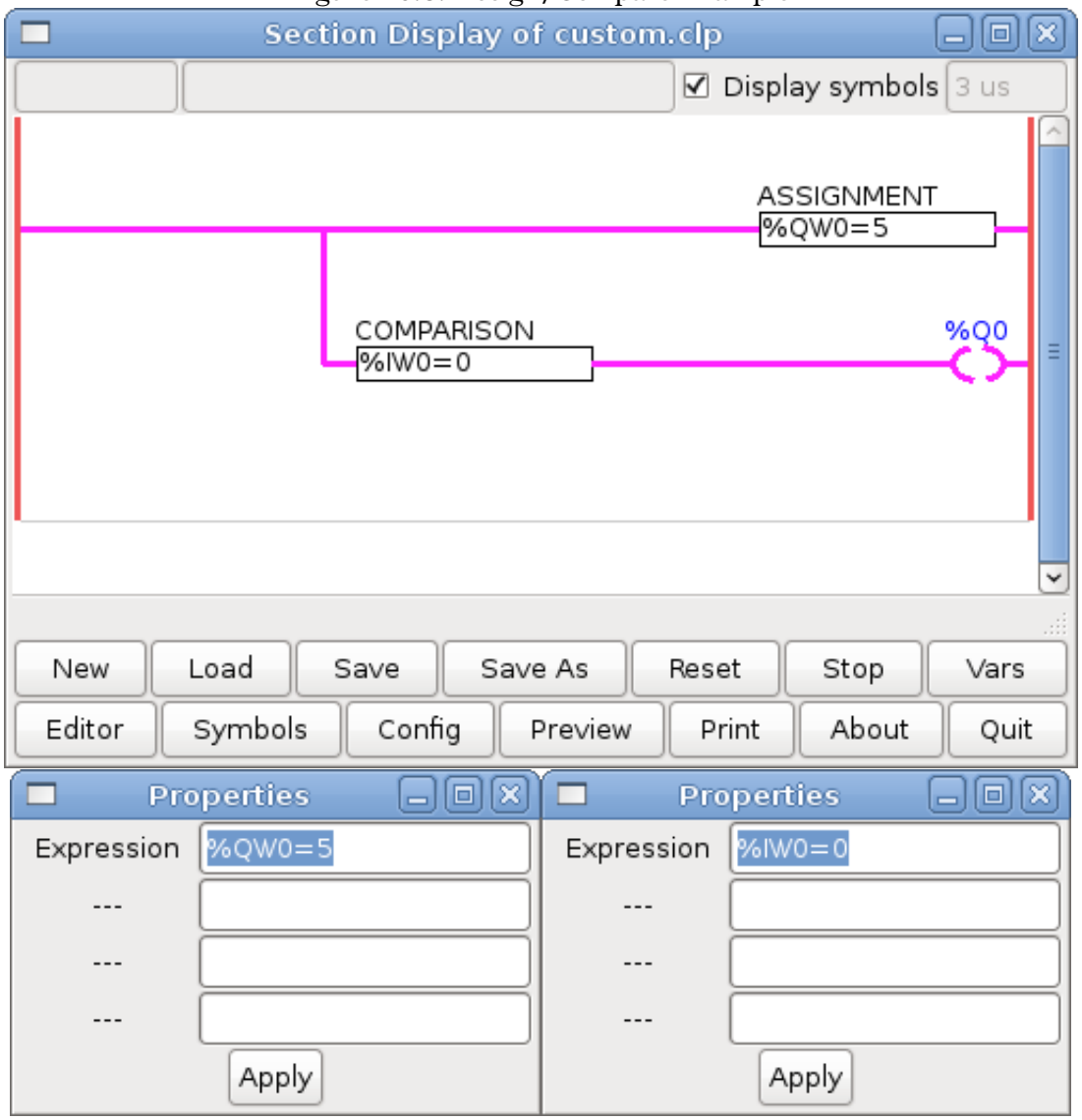
%TM0.P=10

To assign the value of 12 to S32out bit 3 the syntax would be:

%QW3=12

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 20.8: Assign/Compare Example



20.7.8 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. eg 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)

WARNING 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!

20.7.8.1 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 right most position and change the label to the rung you want to JUMP to.

20.7.8.2 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 right most position in the rung.

Remember to change the label to the subroutine number you choose before.

20.8 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 the 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

20.9 GRAFCET Programming

* WARNING -These is probably the least used/known about 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?)

20.10 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 eg. loadusr -w classicladder --modmaster myprogram.clp (assuming myprogram.clp is present -w makes HAL wait till you close Classic Ladder before closing realtime session) my idea behind this is to get a working modbus solution out there then we can decide how it should be done in the best way. As it stands now Classic Ladder also loads a TCP modbus slave (if you add --modserver on command line) - I have not tested this nor have I tested the TCP modbus master. I have done some testing with the serial port and had to add some functions to get it to talk to my VFD -but it does work. Modbus function 1,2,3,4,5,6,8,15,16 (read coils,read inputs, read holding registers, read input registers, write single coils, write single register, echo test, write multiple coils, write multiply registers) are currently available. If you do not specify a --modmaster when loading the Classic Ladder user program this (next) page will not be displayed.

Figure 20.9: Config I/O

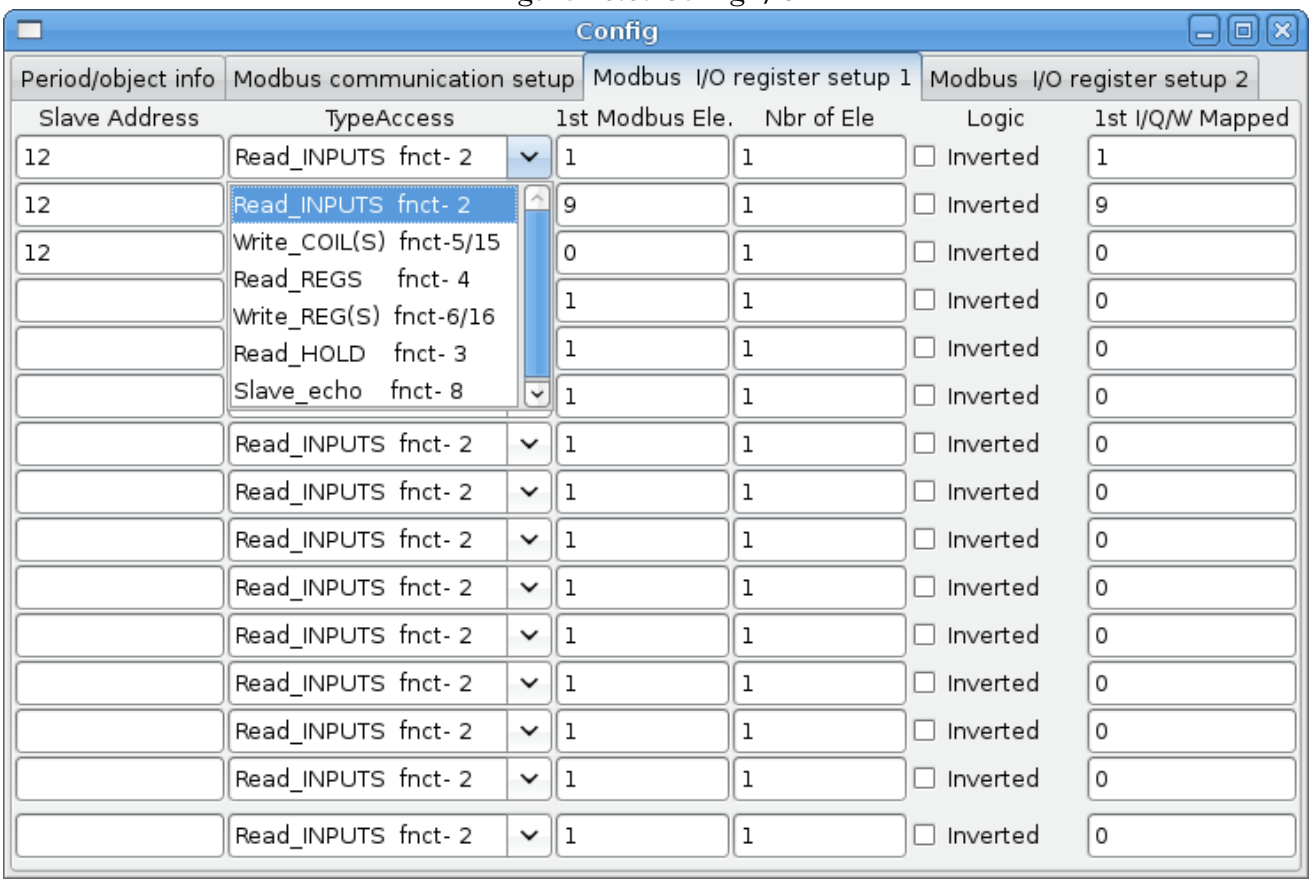
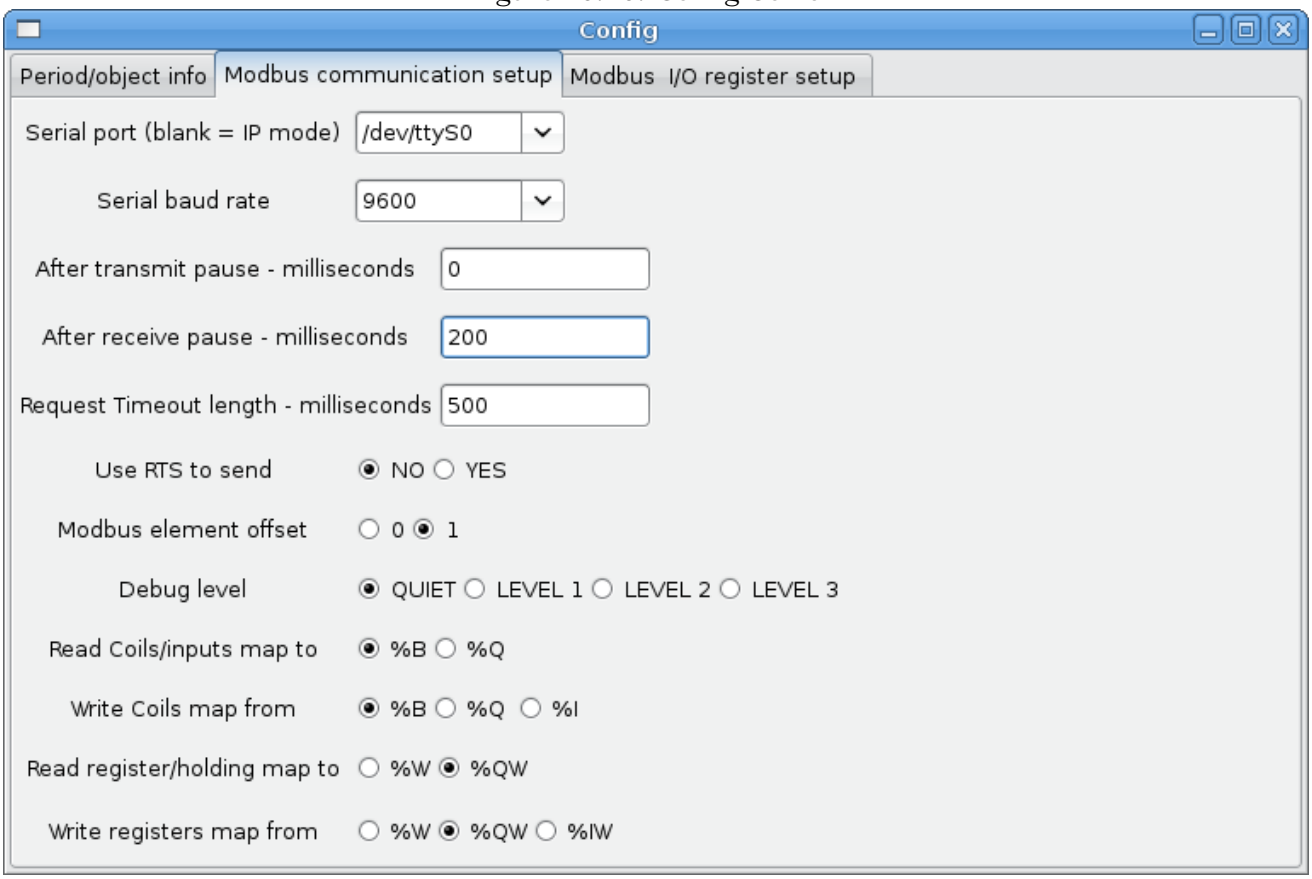


Figure 20.10: Config Coms



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 (eg usb-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 are read/written to I, B, or Q variables (user selects)

registers and holding registers map to W, IW, or QW variables (usr 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 and maps them on to Classic Ladder's %B variables starting at %B1 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 and maps them on to Classic Ladder's %Q variables starting at %Q9 ending at %Q10.

The third line is set to write 2 registers (16 bit each) starting at the 0th register number (register 0) so register numbers 0-1 and maps them on to 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.

20.10.1 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)

20.10.2 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.

Info on modbus protocol are available here:

http://www.modbus.org/

http://www.sourceforge.net/projects/jamod

http://www.modicon.com/techpubs/toc7.html

20.10.3 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 and 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.

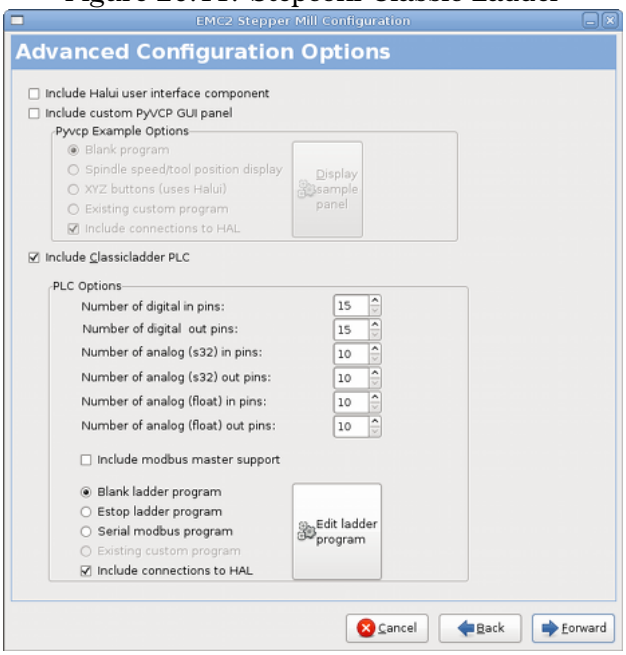
20.10.4 MODBUS Bugs

- 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 multiply registers in Modbus has checksum errors

20.11 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 20.11: Stepconf Classic Ladder



20.11.1 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
```

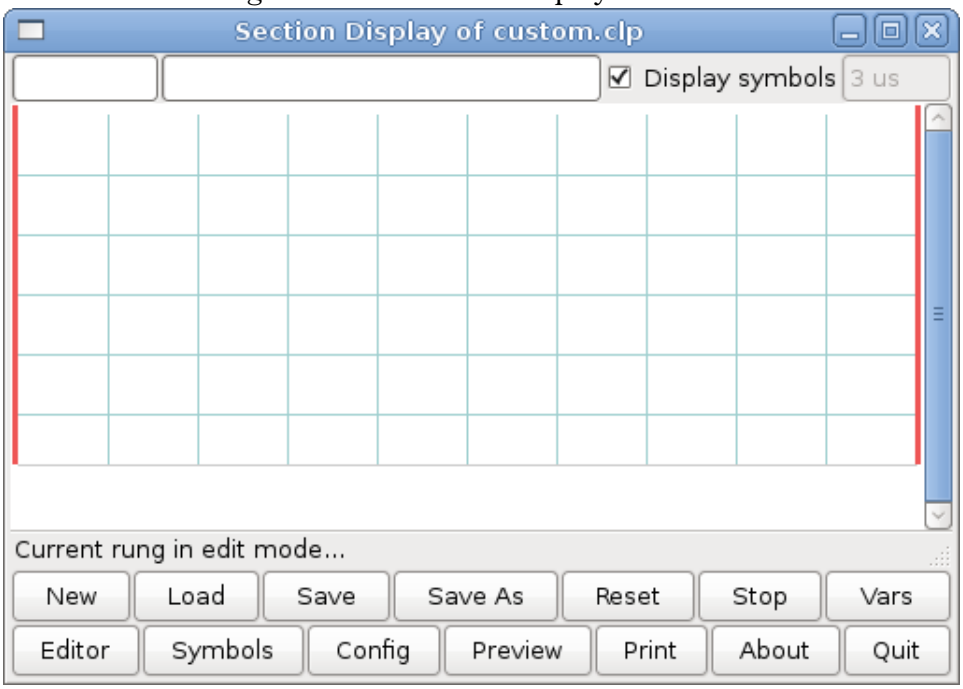
This line adds the Classic Ladder function to the servo thread:

```
addf classicladder.0.refresh servo-thread
```

20.11.2 Adding Ladder Logic

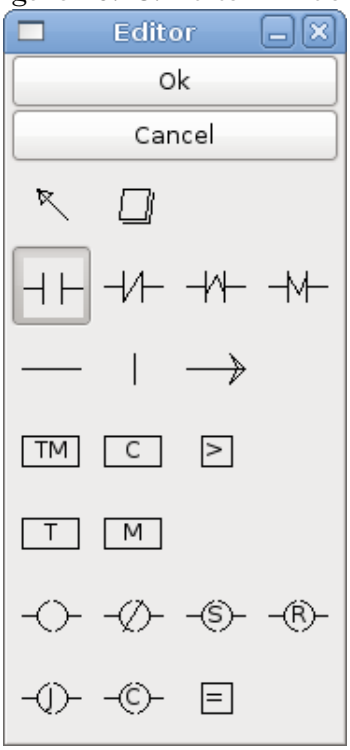
Now start up your config and select File/Ladder Editor... to open up the Classic Ladder GUI. You should see a blank Section Display and Sections Manager window as shown above. In the Section Display window open the Editor. In the Editor window select Modify. Now a Properties window pops up and the Section 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.

Figure 20.12: Section Display with Grid



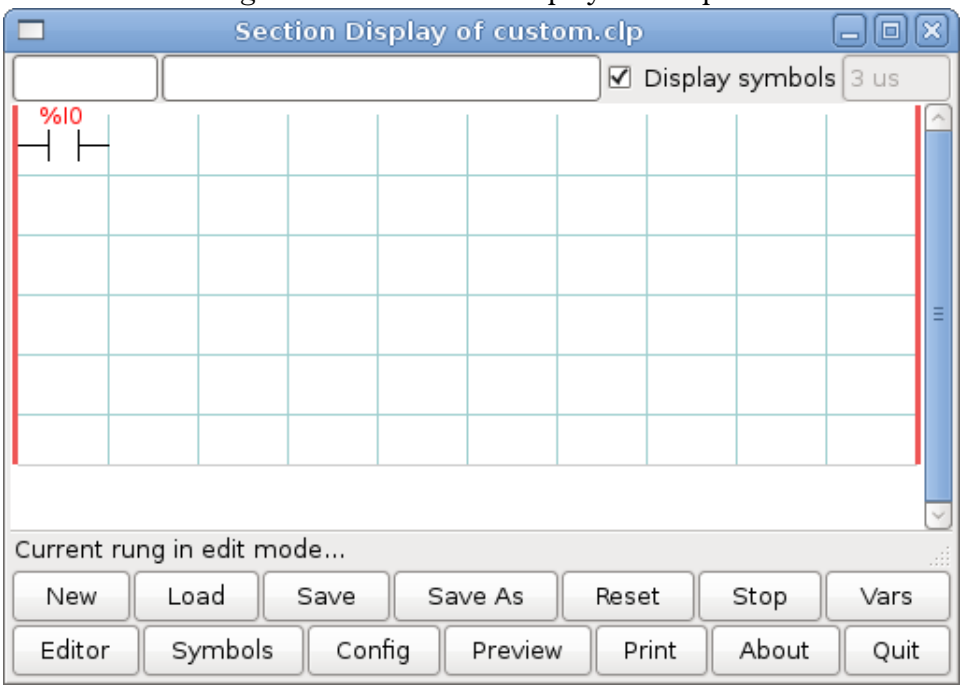
Now click on the N.O. Input in the Editor Window.

Figure 20.13: Editor Window



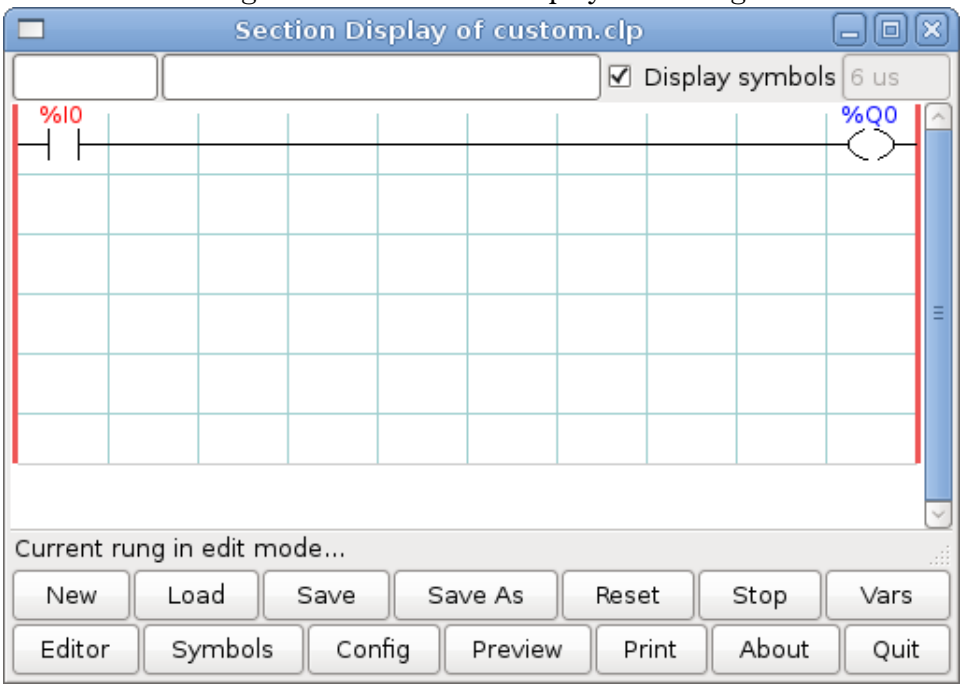
Now click in the upper left grid to place the N.O. Input into the ladder.

Figure 20.14: Section Display with Input



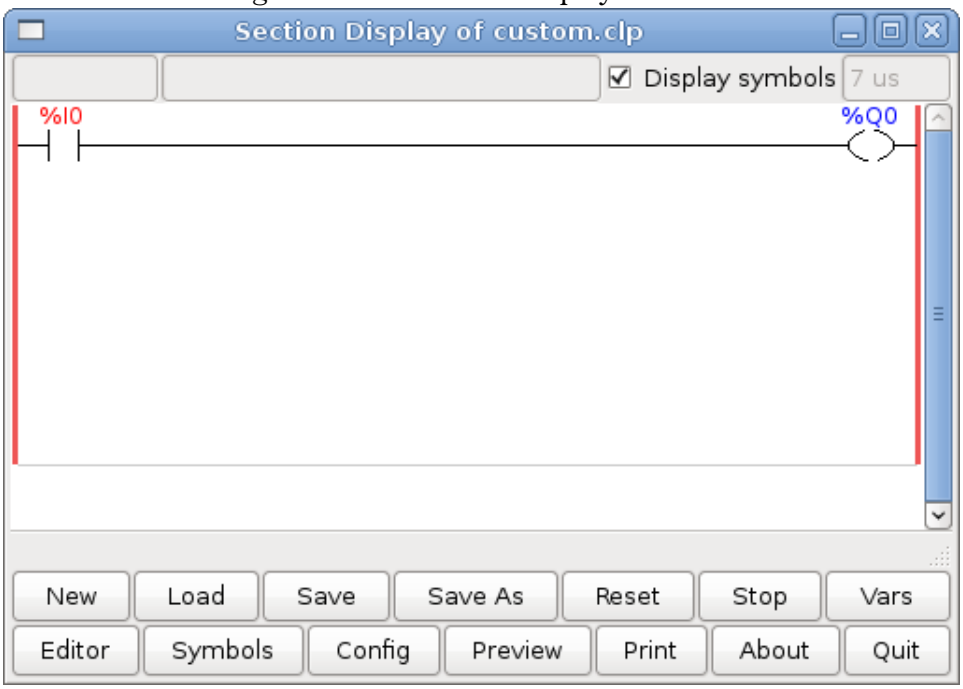
Repeat the above steps to add a N.O. Output to the upper right grid and use the Horizontal Connection to connect the two. It should look like the following. If not use the Eraser to remove unwanted sections.

Figure 20.15: Section Display with Rung



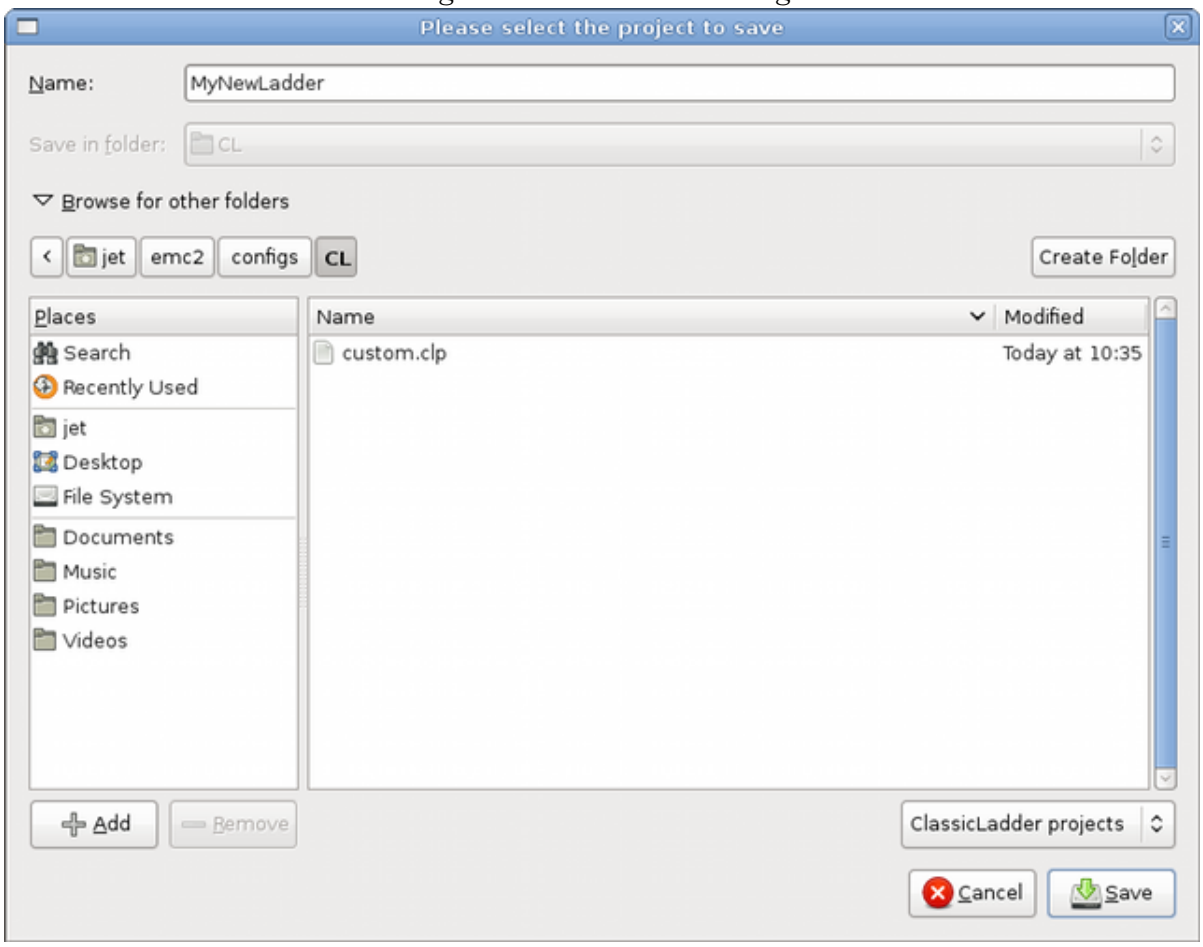
Now click on the OK button in the Editor window. Now your Section Display should look like this.

Figure 20.16: 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 20.17: 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 EMC2 session and add this line to your custom.hal file.

```
loadusr -w classicladder --nogui MyLadder.clp
```

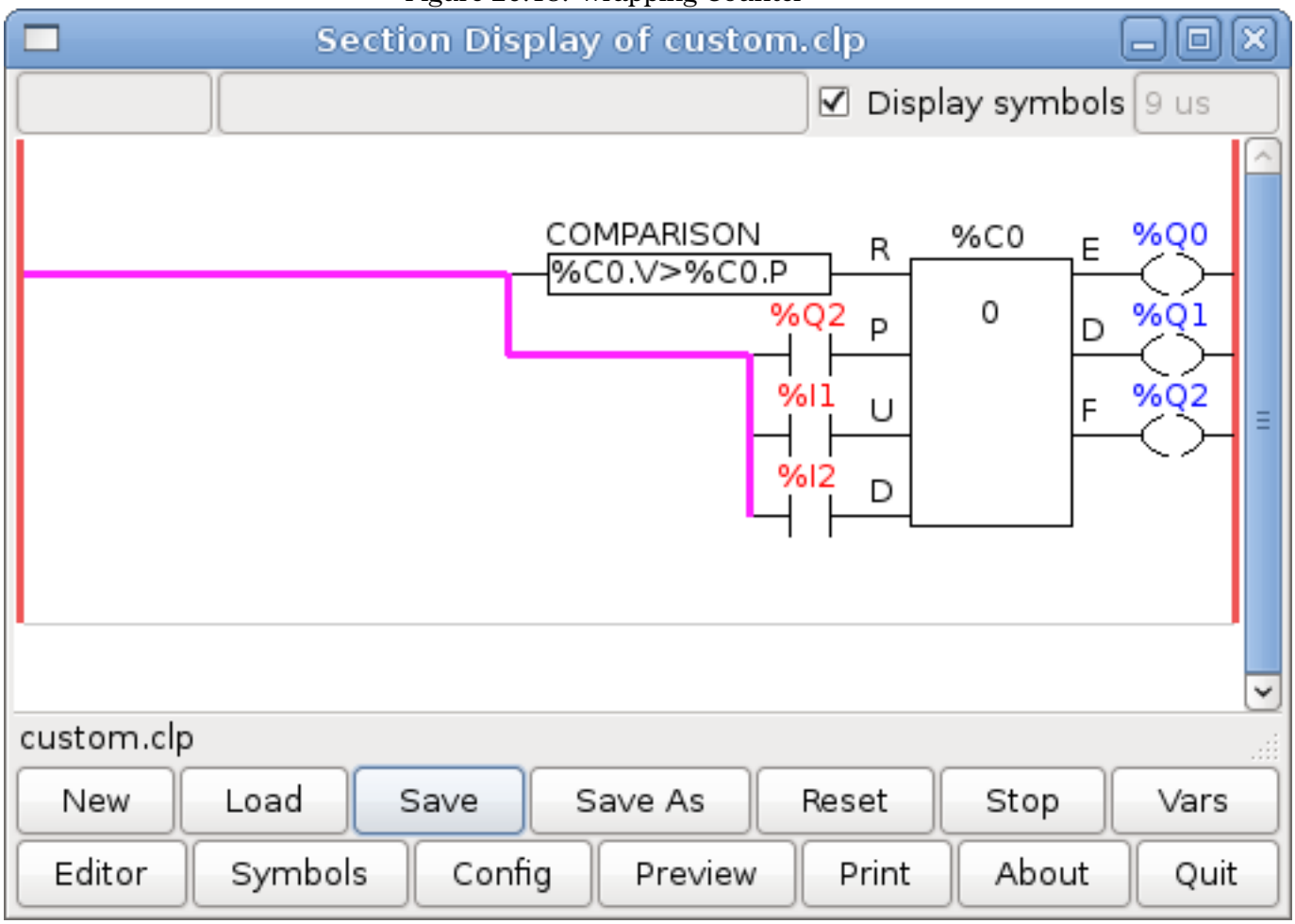
Now if you start up your EMC2 config your ladder program will be running as well. If you select File/Ladder Editor... the program you created will show up in the Section Display window.

20.12 Ladder Examples

20.12.1 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.

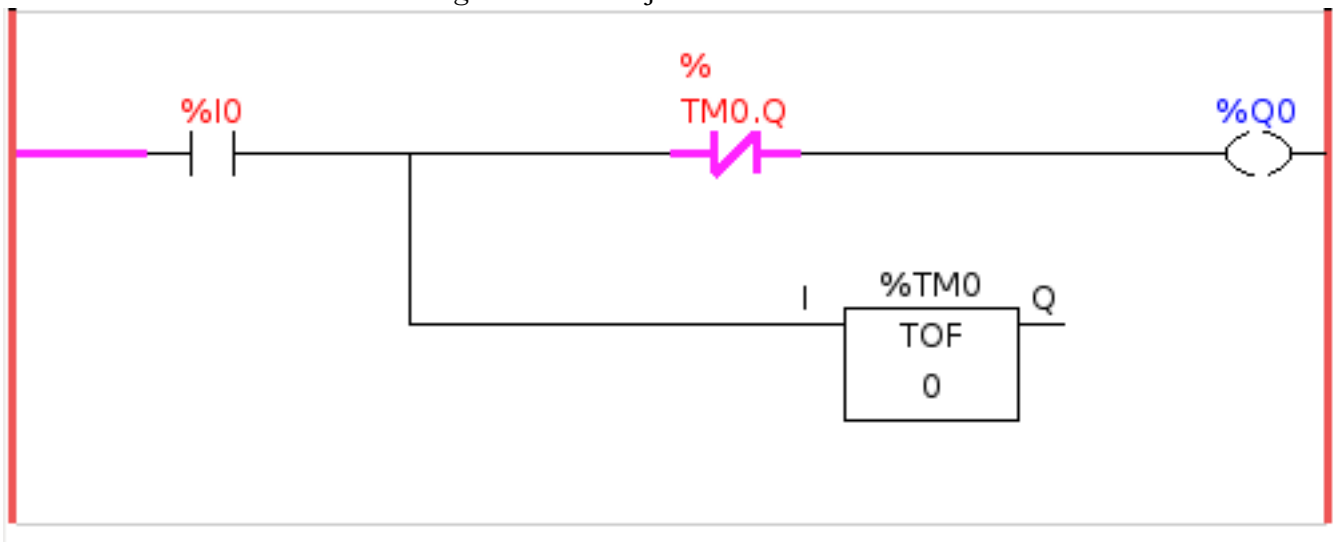
Figure 20.18: Wrapping Counter



20.12.2 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 20.19: Reject Extra Pulse



20.12.3 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 EMC and have the external E-Stop work together with the internal E-Stop requires a couple of connections through ClassicLadder.

First we have to open the E-Stop loop in the main hal file by commenting out by adding the pound sign as shown or removing the following lines.

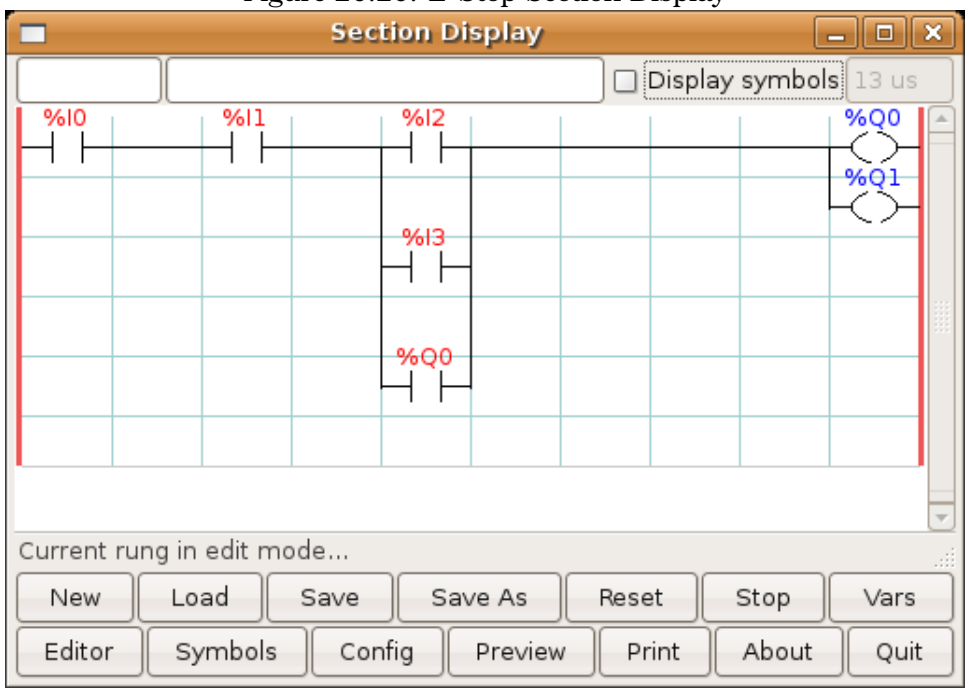
```
# net estop-out <= iocontrol.0.user-enable-out
# net estop-out => iocontrol.0.emc-enable-in
```

Next we add ClassicLadder to our custom.hal file by adding these two lines:

```
loadrt classicladder_rt
addf classicladder.0.refresh servo-thread
```

Next we run our config and build the ladder as shown here.

Figure 20.20: 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 EMC's E-Stop
- %I2 = Input from EMC's E-Stop Reset Pulse
- %I3 = Input from the pyVCP panel reset button
- %Q0 = Ouput to EMC 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 EMC
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
```

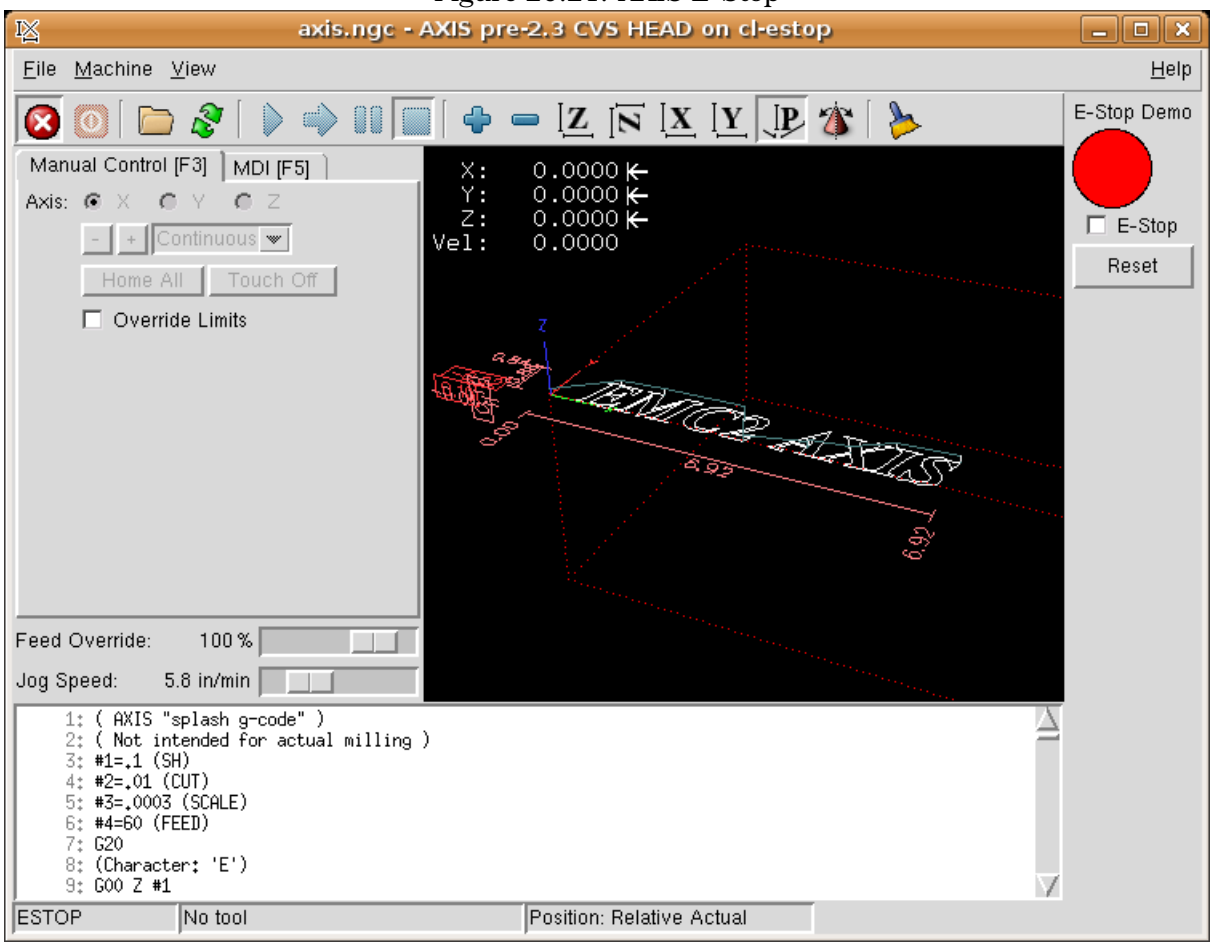
```
# This line resets the E-Stop from EMC
net emc-reset-estop iocontrol.0.user-request-enable => classicladder.0.in-02

# This line enables EMC to unlatch the E-Stop in classicladder
net emc-estop iocontrol.0.user-enable-out => classicladder.0.in-01

# This line turns on the green indicator when out of E-Stop
net estop-all-ok => pyvcp.py-es-status
```

Next we add the following lines to the panel.xml file. Note you have to open it with the text editor not the default html viewer.

```
<pyvcp>
<vbox>
<label><text>"E-Stop Demo"</text></label>
<led>
<halpin>"py-es-status"</halpin>
<size>50</size>
<on_color>"green"</on_color>
<off_color>"red"</off_color>
</led>
<checkbutton>
<halpin>"py-estop"</halpin>
<text>"E-Stop"</text>
</checkbutton>
</vbox>
<button>
<halpin>"py-reset"</halpin>
<text>"Reset"</text>
</button>
</pyvcp>
```

Now start up your config and it should look like this.

Figure 20.21: AXIS E-Stop

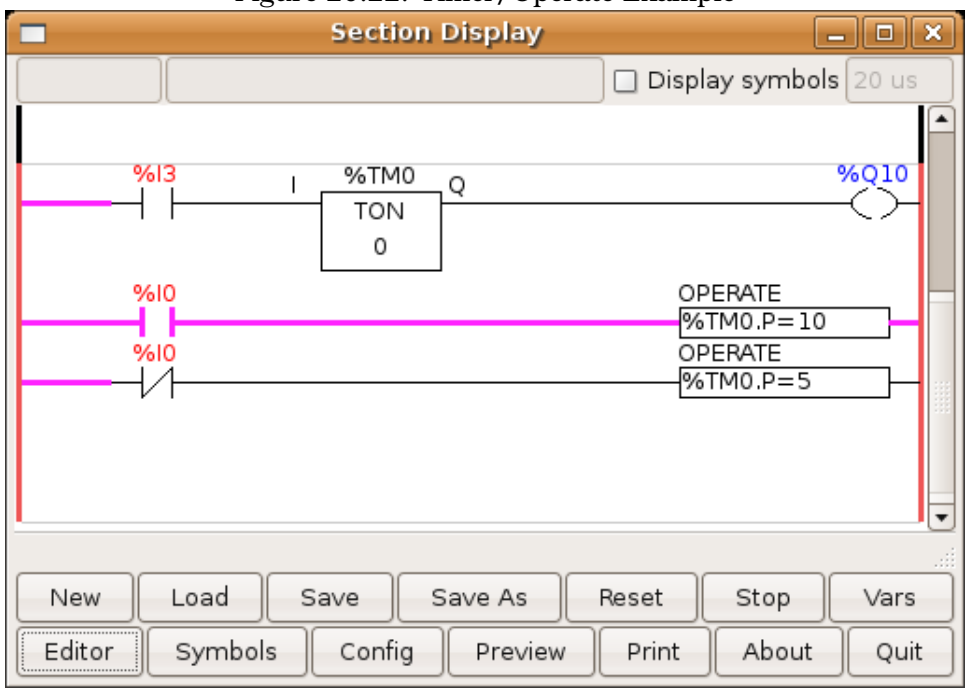


Note that in this example like in real life you must clear the remote E-Stop (simulated by the checkbox) before the AXIS E-Stop or the external Reset will put you in OFF mode. If the E-Stop in the AXIS screen was pressed you must press it again to clear it. You cannot reset from the external after you do an E-Stop in AXIS.

20.12.4 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 20.22: 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.

20.12.5 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 ClassicLadder'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 EMC.You would have to load ClassicLadder's real time module specifying that you want S32 in and out pins. See 'loading options' above. [display turret sample]

20.12.6 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 then 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 mouth full!! [display sequential program]

Part VII

Exemples d'utilisation

Chapter 21

Deuxième port parallèle

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 EMC2. 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:

0000:00:10.0 Communication controller: NetMos Technology PCI 1 port parallel adapter (rev 01)

Subsystem: LSI Logic / Symbios Logic: Unknown device 0010

Flags: medium devsel, IRQ 11

I/O ports at a800 [size=8]

I/O ports at ac00 [size=8]

I/O ports at b000 [size=8]

I/O ports at b400 [size=8]

I/O ports at b800 [size=8]

I/O ports at bc00 [size=16]

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"
```

Notez 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.

Chapter 22

Contrôle de la broche

22.1 Vitesse broche en 0-10V

Si la vitesse de votre broche est contrôlée par un variateur de fréquence avec une consigne vitesse en 0 à 10V et que vous utilisez une carte de conversion (DAC) comme la m5i20 pour sortir le signal.

Premièrement vous devez 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. $10/5000 = 0.002$ notre facteur d'échelle sera donc de 0.002

Si votre carte DAC ne dispose pas d'une fonction échelle, nous devrons 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>

22.2 Vitesse de broche en PWM

Si la vitesse de votre broche peut être contrôlée par un signal de PWM, utilisez 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

setp pwmgen.0.scale 1800 # Adaptez selon la vitesse maximale de votre broche en tr/mn

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, suivez l'exemple "nist-lathe" fourni dans les exemples de configuration pour ajouter un composant d'échelle.

22.3 Marche broche

Si vous voulez un signal de marche broche reliant “motion.spindle-on” à une broche de sortie physique. Pour relier ces pins à une broche du port parallèle, ajouter une ligne comme la suivante dans votre fichier .hal, il faut bien sûr qu’elle soit câblée à votre interface de contrôle.

net spindle-enable motion.spindle-on => parport.0.pin-14-out

22.4 Sens de rotation de la broche

Si vous voulez contrôler le sens de rotation de votre 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 utilisez, par exemple, les lignes suivantes dans votre fichier .hal, bien sûr ces broches doivent être câblées à votre 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

Chapter 23

Vitesse de broche avec signal de retour

Une information de retour est nécessaire pour qu'EMC puisse réaliser les 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 pin 10 du port parallèle.
- L'index de ce codeur est connecté à la pin 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
```

Chapter 24

Manivelle (MPG)

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 votre fichier "custom.hal" ou dans un autre fichier .hal, ajoutez ce qui suit en vérifiant bien que les composants mux4 ou encoder ne soient pas déjà utilisés. Si c'était le cas vous auriez juste à augmenter le nombre de ces composants.

```
# 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 calibré,
# la durée du mouvement total peut dépasser la durée de rotation de la manivelle.
# C'est le mode par défaut.
setp axis.N.jog-vel-mode 0

# Mode vitesse
# L'axe s'arrête quand la manivelle s'arrête, même si la pas de jog est incomplet.
# Décommenter la ligne suivante pour obtenir ce mode de fonctionnement,
# et commenter le mode position.
#setp axis.N.jog-vel-mode 1

# Chaque axe est ajusté indépendemment 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

# Sélecteur 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
```

```
# Sélecteur 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
```

Part VIII

Diagnostic

Chapter 25

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.

25.1 Problèmes communs

25.1.1 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.

25.1.2 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.

25.1.3 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.

25.2 Messages d'erreur

25.2.1 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. EMC 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. EMC2 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.

25.2.2 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.

EMC2 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.

25.3 Tester

25.3.1 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 emc2/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.

```
( Faite Z=0 au dessus de la table avant de démarrer! )
( 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 delai après chaque mouvement )
( test des réglages d'accélération 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 arrêt ici)
#1000=100 ( boucle 100 fois )
( Les boucles suivantes n'ont plus de délai en fin de mouvements )
( test des "hold times" de la conf. du pilote et le réglage de max accel )
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
```

Part IX

Petite FAQ Linux

Chapter 26

Petite FAQ Linux

Voici quelques commandes et techniques de base pour l'utilisateur débutant sous Linux. Beaucoup d'autres informations peuvent être trouvées sur le site web http://www.linuxcnc.org/ ou dans les man pages.

26.1 Login automatique

Quand vous installez EMC2 avec le CD-Live Ubuntu, par défaut vous devez passer par la fenêtre de connexion à chaque démarrage du PC. Pour activer le login automatique ouvrez le menu "Système" ->"Administration"->"Fenêtre de connexion". Si l'installation est récente la fenêtre de connexion peut prendre quelques secondes pour s'ouvrir. Vous devez entrer le mot de passe utilisé pour l'installation pour accéder à la fenêtre des préférences. Ouvrez alors l'onglet "Sécurité", cochez la case "Activer les connexions automatiques" et saisissez votre nom d'utilisateur ou choisissez en un dans la liste déroulante. Vous êtes maintenant dispensé de la fenêtre de connexion.

26.2 Les Man Pages

Les Man pages sont des pages de manuel générées automatiquement le plus souvent. Les Man pages existent pour quasiment tous les programmes et les commandes de Linux.

Pour visualiser une man page ouvrez un terminal depuis Applications > Accessoires > Terminal. Par exemple si vous voulez trouver quelques choses concernant la commande "find", tapez alors dans le terminal:

```
man find
```

Utilisez les touches "Vers le haut" et "Vers le bas" pour faire défiler le texte et la touche "Q" pour quitter.

26.3 Lister les modules du noyau

En cas de problème il est parfois utile de connaître la liste des modules du noyau qui sont chargés. Ouvrez une console et tapez:

```
lsmod
```

Si vous voulez, pour le consulter tranquillement, envoyer le résultat de la commande dans un fichier, tapez la sous cette forme:

```
lsmod > mes_modules.txt
```

Le fichier mes_modules.txt résultant se trouvera alors dans votre répertoire home si c'est de là que vous avez ouvert la console.

26.4 Editer un fichier en root

Editer certains fichiers du système en root peut donner des résultats inattendus! Soyez trés vigilant quand vous éditez en root. Vous pouvez ouvrir et lire de nombreux fichiers systèmes appartenant au root qui sont en mode lecture seule.

26.4.1 A la ligne de commande

Ouvrir un terminal depuis Applications > Accessoires > Terminal.

Dans ce terminal, tapez:

```
sudo gedit
```

Ouvrez un fichier depuis Fichiers, Ouvrir puis éditez le.

26.4.2 En mode graphique

1. Faites un click droit sur le bureau et choisissez Créer un lanceur.
2. Tapez un nom tel que "éditeur, dans la zone "Nom"".
3. Entrez `gksudo "gnome-open %u"` dans la zone "Commande" et validez.
4. Glissez un fichier et déposez le sur votre lanceur, il s'ouvrira alors dans l'éditeur.

26.5 Commandes du terminal

26.5.1 Répertoire de travail

Pour afficher le chemin du répertoire courant dans le terminal tapez:

```
pwd
```

26.5.2 Changer de répertoire

Pour remonter dans le répertoire précédent, tapez dans le terminal:

```
cd ..
```

Pour remonter de deux niveaux de répertoire, tapez dans le terminal:

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

Pour aller directement dans le sous-répertoire emc2/configs tapez:

```
cd emc2/configs
```

26.5.3 Lister les fichiers du répertoire courant

Pour voir le contenu du répertoire courant tapez:

dir

ou

ls

26.5.4 Trouver un fichier

La commande “find” peut être un peu déroutante pour le nouvel utilisateur de Linux. La syntaxe de base est:

find répertoire_de_départ paramètres actions

Par exemple, pour trouver tous les fichiers .ini dans votre répertoire emc2 utilisez d’abord la commande “pwd” pour trouver le répertoire courant. Ouvrez un nouveau terminal et tapez:

pwd

il vous est retourné par exemple le résultat suivant:

/home/robert

Avec cette information vous pouvez taper, par exemple, la commande:

find /home/robert/emc2 –name *.ini –print

Le *-name* est le nom de fichier que vous recherchez et le *-print* indique à find d’afficher le résultat dans le terminal. Le **.ini* indique à find de retourner tous les fichiers contenant l’extension *.ini*

26.5.5 Rechercher un texte

Tapez dans un terminal:

grep –i –r ’texte à rechercher’ *

Pour trouver tous les fichiers contenant le texte *’texte à rechercher’* dans le répertoire courant et tous ses sous-répertoires et en ignorant la casse. Le paramètre -i demande d’ignorer la casse et le -r demande une recherche récursive (qui inclus tous les sous-répertoires dans la recherche). Le caractère * est un jocker indiquant “tous les fichiers”.

26.5.6 Messages du boot

Pour visualiser les messages du boot utilisez la commande “dmesg” depuis un terminal. Pour enregistrer ces messages dans un fichier redirigez les avec:

```
dmesg > dmesg.txt
```

Le contenu de ce fichier pourra alors être copié et collé à destination des personnes en ligne qui vous aideront à diagnostiquer votre problème.

26.6 Problémes matériels

26.6.1 Informations sur le matériel

Pour voir la liste du matériel utilisé par votre carte mère, tapez la commande suivante dans un terminal:

```
lspci –v
```

26.6.2 Résolution du moniteur

Lors de l'installation d'Ubuntu les réglages du moniteur sont détectés. Il peut arriver que ça fonctionne mal et que la résolution ne soit que celle d'un moniteur générique en 800x600.

Pour résoudre ce cas, suivez les instructions données ici:

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

Part X

Annexes

Chapter 27

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 control 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. Additional linear axes parallel to X, Y, and Z are called U, V, and W respectively. Angular axes like rotary tables are referred to as A, B, and C.

Axis One of the Graphical User Interfaces available to users of EMC2. Features the modern use of menus and mouse buttons while automating and hiding some of the more traditional EMC2 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 EMC2 HAL components.

Configuration(n) A directory containing a set of configuration files. Custom configurations are normally saved in the users home/emc2/configs directory. These files include EMC'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 EMC2 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 device attached to the slides of a machine tool or other device which has parts that move in a precise manner to indicate the current location of the tool with respect to some reference position. Nearly all DRO's use linear quadrature encoders to pick up position information from the machine.

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 cam make corners with a radius only slightly larger than the radius on the corner of the convex EDM electrode.

EMC The Enhanced Machine Controller. Initially a NIST project. EMC is able to run a wide range of motion devices.

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

EMCMOT The module within EMC 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 emc2.

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

Feed rate The speed at which a motion occurs. In manual mode, jog speed can be set from the graphical interface. In auto or mdi mode feed rate is commanded using a (f) word. F10 would mean ten units per minute.

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

Feed rate 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, EMC uses RS274/NGC.

GUI Graphical User Interface.

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

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

HAL **H**ardware **A**bstraction **L**ayer. 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 EMC 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.

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 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 used in the inifile. 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

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

Program Units The linear and angular units used for part programs.

Python General-purpose, very high-level programming language. Used in EMC2 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 material during a rapid, it is probably a bad thing!

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 EMC can use to achieve real-time performance.

RTLINUX See http://www.rtlinux.org, one of two real-time extensions for Linux that EMC 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 EMC part programs.

Servo Motor A special kind of motor that uses error-sensing feedback to correct the position of an actuator.

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 negitive sign. (-12) In HAL it is known as s32.

Spindle 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 assume that the EMC2 software has been configured to control spindle speed.

Stepconf An EMC2 configuration wizard. It is able to handle many step-and-direction motion command based machines. Writes a full configuration after the user answers a few questions about the computer and machine to be run with.

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 EMC that coordinates the overall execution and interprets the part program.

Tcl/Tk A scripting language and graphical widget toolkit with which several of the EMC's GUI's 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. (123) In HAL it is known as u32.

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

Chapter 28

Legal Section

28.1 Copyright Terms

28.2 GNU Free Documentation License

GNU Free Documentation License Version 1.1, March 2000

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 parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Part XI

Index de l'ouvrage

Index