LinuxCNC V2.8.1-84-g81a16a1fd, 2021-03-29

1章 前書き

このハンドブックは進行中の作業です。 執筆、編集、またはグラフィックの準備を手伝うことができる場合は、執筆チームのメンバーに連絡するか、参加して EMC-USERS@LISTS.SOURCEFORGE.NET に電子メールを送信してください。

2章 LinuxCNC の歴史

2.1 起源

EMC（ENHANCED MACHINE CONTROLLER）は、米国政府の商務省の機関である米国国立標準技術研究所である NIST によって作成されました。

NIST は、コンセプトと標準のテストプラットフォームとしてモーションコントロールパッケージを作成することに最初に興味を持ちました。 ゼネラルモーターズからの初期のスポンサーシップにより、WINDOWS NT の「リアルタイム」バージョンで実行され、大型フライス盤を制御する PMAC インテリジェント制御ボードを使用して、新しいバージョンの EMC が採用されました。

米国連邦政府職員のすべての作業成果物に要求されるように、結果として得られるソフトウェアとそれに関するレポートはパブリックドメインである必要があり、それに関するレポートはインターネットを含めて正式に公開されました。 MATTSHAVER が EMC を発見したのはそこでした。 彼は NIST に連絡し、廃止された、またはまったく機能しなくなった CNC 制御のアップグレードと交換に使用される、より安価なハードウェアの制御に使用するコードの適合について、FREDPROCTOR と話し合いました。 NIST は、もっと安価なものが欲しかったので興味をそそられました。 協力的な取り組みを開始するために、結果として得られるコードと設計がパブリックドメインのままであることを保証する正式な合意が作成されました。

初期の検討事項は、高価で気まぐれな「リアルタイム」WINDOWSNT システムの交換に焦点を合わせていました。 LINUX オペレーティングシステムの比較的新しい（当時の）リアルタイム拡張を試すことが提案されました。このアイデアは成功裏に追求されました。次は、高価なインテリジェントモーションコントロールボードの問題でした。この時までに、PC の処理能力は、モーションルーチンを直接制御するのに十分であると考えられていました。利用可能なハードウェアをすばやく検索した結果、PC にモーターを直接制御させるための最初のプラットフォームとして SERVO-TO-GO インターフェイスボードが選択されました。軌道計画と PID ループ制御のためのソフトウェアが既存のユーザーインターフェースと RS274 インタープリターに追加されました。マットはこのバージョンを使用して、デッドコントロールを備えたいくつかのマシンをアップグレードすることに成功し、これが最初に外界の注目を集めた EMC システムになりました。 REC.CRAFTS.METALWORKING USENET ニュースグループで EMC に言及した結果、JONELSON のようなアーリーアダプターが EMC を利用するようになりました。

NIST は、EMC に関心のある人々のためにメーリングリストを設定しました。 時間が経つにつれて、NIST 外の他の人々は EMC の改善に興味を持つようになりました。 多くの人がコードの小さな改善を要求またはコーディングしました。 RAY HENRY は、ユーザーインターフェイスを改良したいと考えていました。 RAY は、ユーザーインターフェイスが記述された C コードを改ざんしようとすることに消極的だったため、より簡単な方法が求められました。 NIST の FREDPROCTOR は、スクリプト言語を提案し、TCL / TK スクリプト言語を EMC の内部 NML 通信に接続するためのコードを作成しました。 RAY はこのツールを使用して、当時 EMC の主要なユーザーインターフェイスとなった TCL / TK プログラムを作成しました。

NIST の視点については、WILLIAMSHACKLEFORD と FREDERICKPROCTOR が執筆した、EMC の歴史とオープンソースへの移行について説明したこのペーパーを参照してください。

この時までに、EMC への関心は大幅に高まり始めています。 ますます多くの人々が EMC のインストールを試みるにつれて、LINUX カーネルにリアルタイム拡張機能をパッチすることと EMC コードをコンパイルすることの難しさが明白になりました。 プロセスを文書化し、スクリプトを作成するための多くの試みが試みられましたが、中程度の成功を収めたものもありました。 パッチとコンパイラの正しいバージョンを選択したバージョンの LINUX と一致させるという問題が発生し続けました。 PAUL CORNER は、完全に機能するシステム（LINUX、パッチ、および EMC）をインストールできる CD である BDI（ブレインデッドインストール）で救助に来ました。 BDI アプローチは、EMC の世界をはるかに大きなユーザーコミュニティに開放しました。 このコミュニティが成長し続けるにつれて、EMC メーリングリストとコードアーカイブは SOURCEFORGE に移動され、LINUXCNCWEB サイトが設立されました。

より多くのユーザーコミュニティが参加することで、EMC は NAMES で開催中の CNC 展示会で大きな関心を集め、NAMES は EMC の年次総会イベントになりました。 最初の数年間は、利害関係者が NAMES にいたため、会議が開催されました。 2003 年、EMC ユーザーコミュニティは最初の公開会議を発表しました。 NAMES ショーが開催されたアリーナのロビーで NAMES 後の月曜日に開催されました。 組織は緩いものでしたが、ハードウェアアブストラクションレイヤー（HAL）のアイデアが生まれ、開発を容易にするためにコードを再構築する動き（EMC2）が提案されました。

2.2 名前の変更

2011 年の春、LINUXCNC の取締役会は、LINUXCNC.ORG で提供されるソフトウェアを特定するための「EMC」および「EMC2」の使用について、EMC CORPORATION（WWW.EMC.COM）を代表する法律事務所から連絡を受けました。 EMC CORPORATION は、EMC および EMC2（上付き数字 2 の EMC）に関連するさまざまな商標を登録しています。

EMC CORPORATION の代表者と何度も話し合った後、最終的な結果として、ソフトウェアの次のメジャーリリース以降、LINUXCNC.ORG は、「EMC」または「EMC」、あるいはこれらの用語の後に続くソフトウェアの識別を停止します。 数字。 LINUXCNC 理事会が、LINUXCNC.ORG で提供されるソフトウェアを識別するために使用される名前を管理する範囲で、理事会はこれに同意しました。

その結果、ソフトウェアの新しい名前を選択する必要がありました。 理事会が検討したオプションの中で、「LINUXCNC」が最良のオプションであるというコンセンサスがありました。これは、長年にわたって当社の WEB サイトの名前であったためです。

新しい名前の準備として、LINUX FOUNDATION から LINUX®商標のサブライセンスを受け取りました（WWW.LINUXFOUNDATION。LINUXCNC 名の使用を保護します。LINUX®は、米国およびその他の地域における LINUSTORVALDS の登録商標です。 国。）

ブランド変更の取り組みには、LINUXCNC.ORG WEB サイト、IRC チャネル、および 2.5.0 以降のソフトウェアとドキュメントのバージョンが含まれます。

2.3 追加情報

NIST は、RS274NGC 言語とそれが制御する抽象的なマシニングセンター、および EMC の初期の実装について説明した論文を発表しました。 このペーパーは、HTTP：//LINUXCNC.ORG/FILES/RS274NGCV3.PDF でも入手できます。

NIST は、EMC の歴史とオープンソースへの移行に関する論文も発表しました。 このペーパーは、HTTP：//LINUXCNC.ORG/-FILES/USE-OF-OPEN-SOURCE-DISTRIBUTION-FOR-A-MACHINE-TOOL-CONTROLLER.PDF でも入手できます。

3章 概要

3.1 ユーザーの序文

LinuxCNC はモジュール式で柔軟性があります。 これらの属性により、多くの人はそれを小さなことの紛らわしいごちゃ混ぜと見なし、なぜそれがそのようになっているのか疑問に思います。 このページは、あなたが物事の奥深くに入る前に、その質問に答えようとします。

LinuxCNC は、米国国立標準技術研究所で始まりました。 それはオペレーティングシステムとして Unix を使用して育ちました。 Unix はそれを変えました。 初期の Unix 開発者の間では、Unix の方法と呼ばれる一連のコード記述のアイデアが生まれました。 これらの初期の LinuxCNC 作成者は、これらの方法に従いました。

Eric S. Raymond は、著書 The Art of Unix Programming で、Unix 哲学を、広く使用されているエンジニアリング哲学「Keep it Simple、Stupid」（KISS Principle）として要約しています。 次に、この全体的な哲学が文化的な Unix の規範として適用されていると彼がどのように信じているかを説明しますが、実際の Unix の実践では、当然のことながら、次のほとんどの重大な違反を見つけることは難しくありません。

- モジュール性のルール：クリーンなインターフェースで接続された単純なパーツを記述します。
- 明快さのルール：明快さは賢さよりも優れています。
- 構成のルール：他のプログラムに接続するプログラムを設計します。
- 分離のルール：ポリシーをメカニズムから分離します。 エンジンからインターフェースを分離します。1)

1) http：//en.wikipedia.org/wiki/Unix_philosophy、2008 年 7 月 6 日に見つかりました

レイモンド氏はさらにいくつかのルールを提供しましたが、これら 4 つは LinuxCNC モーションコントロールシステムの本質的な特徴を説明しています。

モジュール性ルールは重要です。 これらのハンドブック全体を通して、通訳またはタスクプランナーまたはモーションまたは HAL の話を見つけることができます。 これらはそれぞれ、モジュールまたはモジュールのコレクションです。 マシンの実行に必要な部品だけを相互に接続できるモジュール性です。

クラリティルールは不可欠です。 LinuxCNC は進行中の作業であり、完成しておらず、今後も完成することはありません。 実行したいほとんどのマシンを実行するのに十分なほど完全です。 多くのユーザーとコード開発者が他のユーザーの作業を見て、彼らが行ったことに基づいて構築できるため、その進歩の多くが達成されます。

構成ルールを使用すると、モジュールを接続可能にすることで、利用可能な多くのモジュールから予測可能な制御システムを構築できます。 モジュールのセットへの標準インターフェースを設定し、それらの標準に従うことにより、接続性を実現します。

分離ルールでは、小さなことを行う別個のパーツを作成する必要があります。 関数を分離することにより、デバッグがはるかに簡単になり、交換モジュールをシステムにドロップして比較を簡単に行うことができます。

LinuxCNC のユーザーとしての Unix の方法は何を意味しますか。 これは、システムの使用方法を選択できることを意味します。 これらの選択の多くはマシン統合の一部ですが、多くはマシンの使用方法にも影響します。 あなたが読むとき、あなたはあなたが比較をする必要があるであろう多くの場所を見つけるでしょう。 最終的には、「それではなくこのインターフェースを使用する」または「パーツオフセットをその方法ではなくこの方法で書き込む」という選択を行います。 これらのハンドブック全体を通して、現在利用可能な能力の範囲について説明します。

LinuxCNC の使用を開始する際に、2 つの注意事項を示します。2)

2) http：//en.wikipedia.org/wiki/Unix_philosophy、2008 年 7 月 6 日に見つかりました

- UNIX での DougGwyn の言葉を言い換えると、「LinuxCNC は、ユーザーが愚かなことをするのを阻止するようには設計されていません。それは、ユーザーが賢いことをするのも阻止するからです。」
- 同様に、スティーブン・キングの言葉：「LinuxCNC はユーザーフレンドリーです。どのユーザーとフレンドリーであるかについては無差別ではありません。」

3.2 LinuxCNC ユーザー紹介

3.2.1 LinuxCNC のしくみ

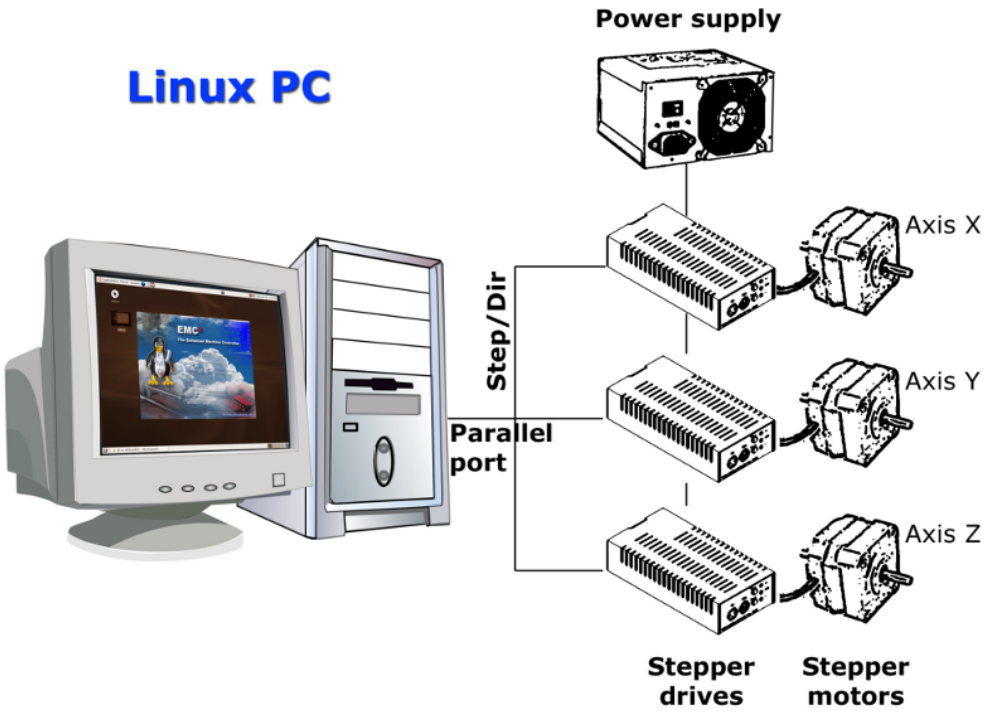
LinuxCNC は、コンピューター数値制御（CNC）ミルおよび旋盤、3D プリンター、ロボット、レーザーカッター、プラズマカッター、およびその他の自動化デバイスを制御するための高度にカスタマイズ可能なアプリケーションのスイートです。 最大 9 軸の移動を協調制御できます。

LinuxCNC は、本質的に、1 つの完全なシステムを形成するために統合されたいくつかの主要なコンポーネントで構成されています。

- グラフィカルユーザーインターフェイス（GUI）。これは、オペレーター、ソフトウェア、および CNC マシン自体の間の基本的なインターフェイスを形成します。
- LinuxCNC によって生成および受信されたさまざまな内部仮想信号すべてを外部とリンクする方法を提供するハードウェアアブストラクションレイヤー（HAL）。 そして、

- CNC マシンのモーション制御の生成と実行を調整する高レベルのコントローラー、つまりモーションコントローラー（EMCMOT）、ディスクリート入出力コントローラー（EMCIO）、タスクエグゼキューター（EMCTASK）。

次の図は、ステッピングモーターを備えた典型的な 3 軸 CNC ミルがどのように見えるかを示す簡単なブロック図です。



LinuxCNC を実行しているコンピュータは、パラレルポートを介して一連のパルスをステッピングドライブに送信します。各ドライブには、1 つのステッピングモーターが接続されています。 各ドライブは 2 つの独立した信号を受信します。 関連するステッピングモーターを時計回りまたは反時計回りの方向に動かすようにドライブに命令する 1 つの信号と、そのステッピングモーターが回転する速度を定義する 2 番目の信号。

パラレルポート制御下のステッピングモーターシステムが示されていますが、LinuxCNC システムは、速度と I / O 機能を向上させるためにさまざまな専用ハードウェアモーションコントロールインターフェイスを利用することもできます。 LinuxCNC でサポートされているインターフェイスの完全なリストは、Wiki の[サポートされているハードウェア]ページにあります。

ほとんどの場合、ユーザーは、Stepper ConfigurationWizard（コンピューターのパラレルポートを使用して動作する CNC システムの場合）または Mesa Hardware Wizard（Mesa Anything I / O PCI カードを使用するより高度なシステムの場合）のいずれかを使用して、ミルセットアップに固有の構成を作成します。 ）。 いずれかのウィザードを実行すると、その CNC マシンに固有のいくつかの構成ファイルを含むコンピューターのハードドライブ上にいくつかのフォルダーが作成され、LinuxCNC を簡単に起動できるようにデスクトップにアイコンが配置されます。

たとえば、ステッパー構成ウィザードを使用して、上記の My_CNC というタイトルの 3 軸 CNC ミルのセットアップを作成した場合、ウィザードによって作成されるフォルダーには通常、次のファイルが含まれます。

- フォルダー：My_CNC
  - My_CNC.ini
    - INI ファイルには、1 回転を完了するために各ステッピングモーターが回転する必要のあるステップ数、各ステッピングモーターが動作できる最大速度、各ステッピングモーターの移動制限など、CNC ミルの動作に関するすべての基本的なハードウェア情報が含まれています。 軸または各軸のリミットスイッチの構成と動作。
  - My_CNC.hal
    - この HAL ファイルには、内部仮想信号をコンピューター以外の物理接続にリンクする方法を LinuxCNC に指示する情報が含まれています。 たとえば、パラレルポートのピン 4 を指定して Z 軸のステップ方向信号を送信したり、パラレルポートのピン 13 でリミットスイッチがトリガーされたときに X 軸モーターの駆動を停止するように LinuxCNC に指示したりします。
  - custom.HAL
    - ウィザードの範囲を超えたミル構成のカスタマイズは、この HAL ファイルに LinuxCNC 内の他の仮想ポイントへのリンクを追加することで実行できます。 LinuxCNC セッションを開始すると、GUI がロードされる前に、このファイルが読み取られて処理されます。 例としては、GUI が表示される前に動作可能であることが確認されるように、スピンドルモーターへの Modbus 通信を開始することが含まれる場合があります。
  - custom_postgui.hal
    - custom_postgui HAL ファイルを使用すると、LinuxCNC をさらにカスタマイズできますが、GUI が表示された後に処理されるという点で custom.HAL とは異なります。 たとえば、custom.hal でスピンドルモーターへの Modbus 通信を確立した後、LinuxCNC は custom_postgui ファイルを使用して、モータードライブからのスピンドル速度の読み取り値を GUI に表示される棒グラフにリンクできます。
  - postgui_backup.hal
    - これは custom_postgui.hal ファイルのバックアップコピーとして提供され、ユーザーが以前に機能していた postguiHAL 構成をすばやく復元できるようにします。 これは、ミルの一部のパラメーターを変更するために、ユーザーが同じ My_CNC 名で構成ウィザードを再度実行する場合に特に役立ちます。 ウィザードでミル構成を保存すると、

postgui_backup ファイルはそのままで、既存の custom_postgui ファイルが上書きされます。

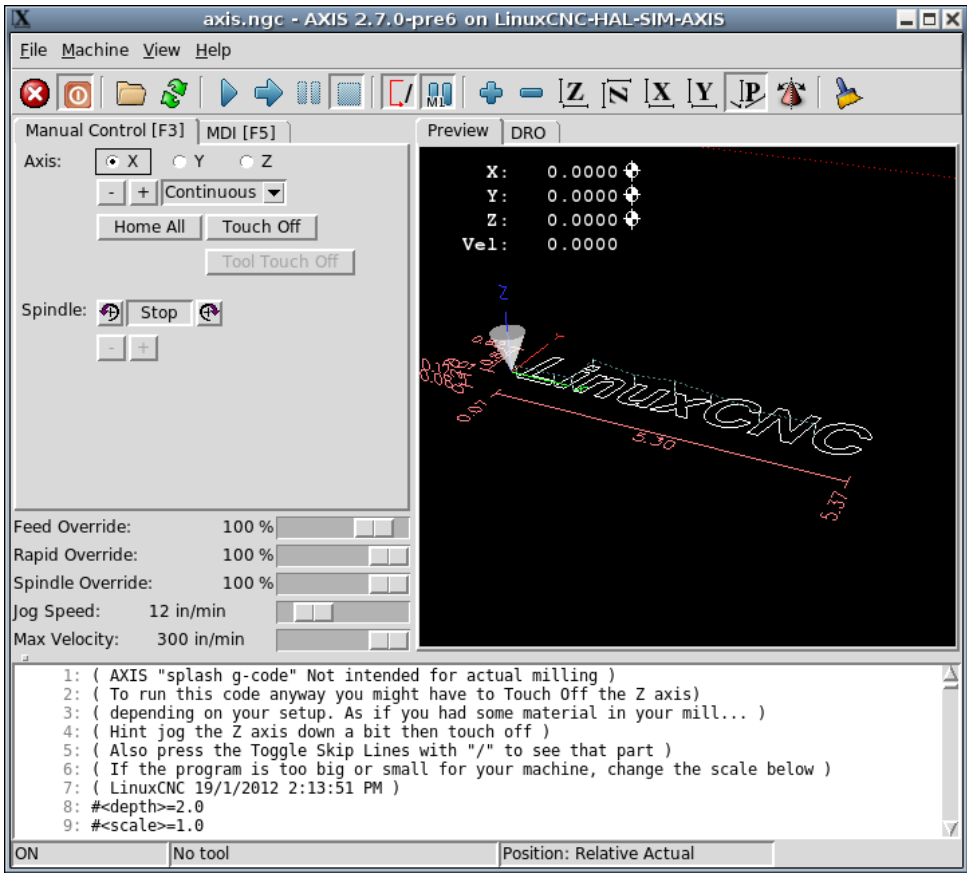
- tool.tbl
  - ツールテーブルファイルには、ミルで使用される切削工具のパラメータ化されたリストが含まれています。 これらのパラメータには、カッターの直径と長さを含めることができ、フライス加工操作内のさまざまなサイズの工具の動きを補正する方法を LinuxCNC に指示するデータのカタログを提供するために使用されます。

- フォルダー：nc_files
  - nc_files フォルダーは、ミルの駆動に使用される G コードプログラムを格納するためのデフォルトの場所として提供されます。 また、G コードの例を含むいくつかのサブフォルダーも含まれています。

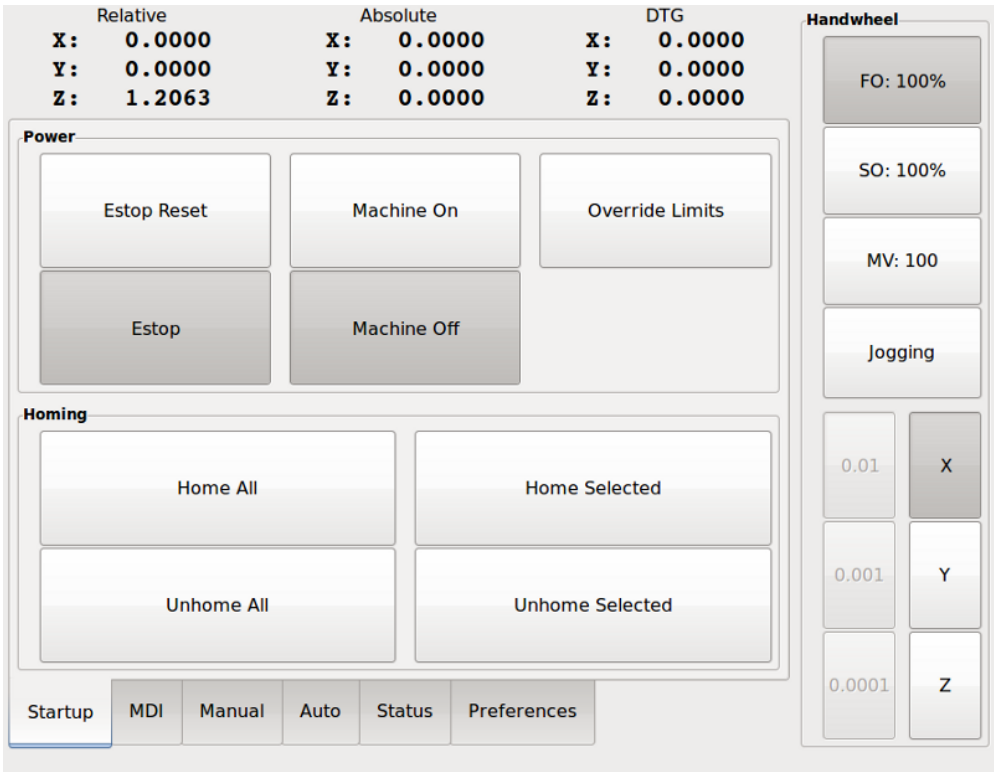
3.2.2 グラフィカルユーザーインターフェイス

グラフィカルユーザーインターフェイスは、工作機械のオペレーターが操作する LinuxCNC の一部です。 LinuxCNC には、INI ファイルに含まれる特定のフィールドを編集することで選択できるいくつかのタイプのユーザーインターフェイスが付属しています。

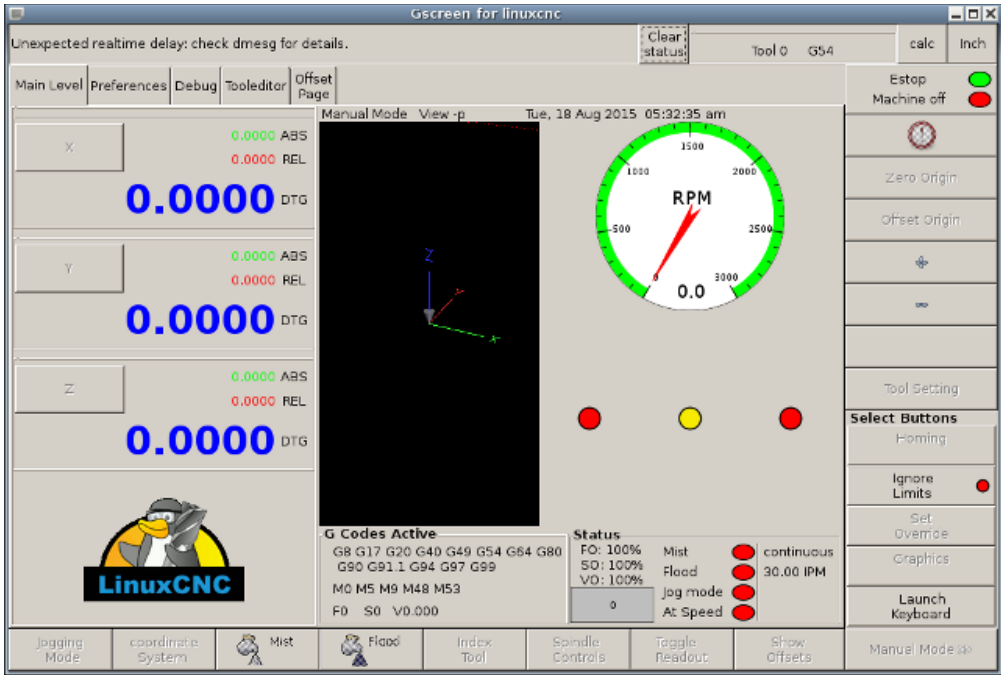
Axis、標準のキーボード GUI インターフェイス。 これは、構成ウィザードを使用してデスクトップアイコンランチャーを作成するときに起動されるデフォルトの GUI でもあります。



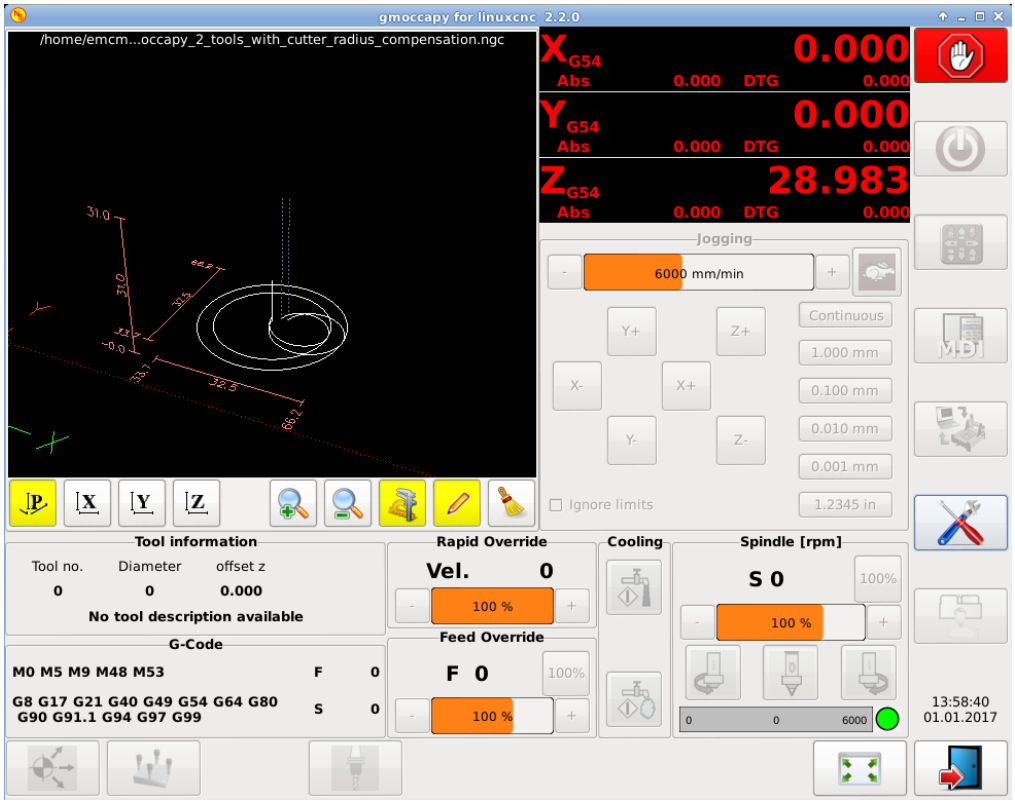
Touchy、タッチスクリーン GUI：



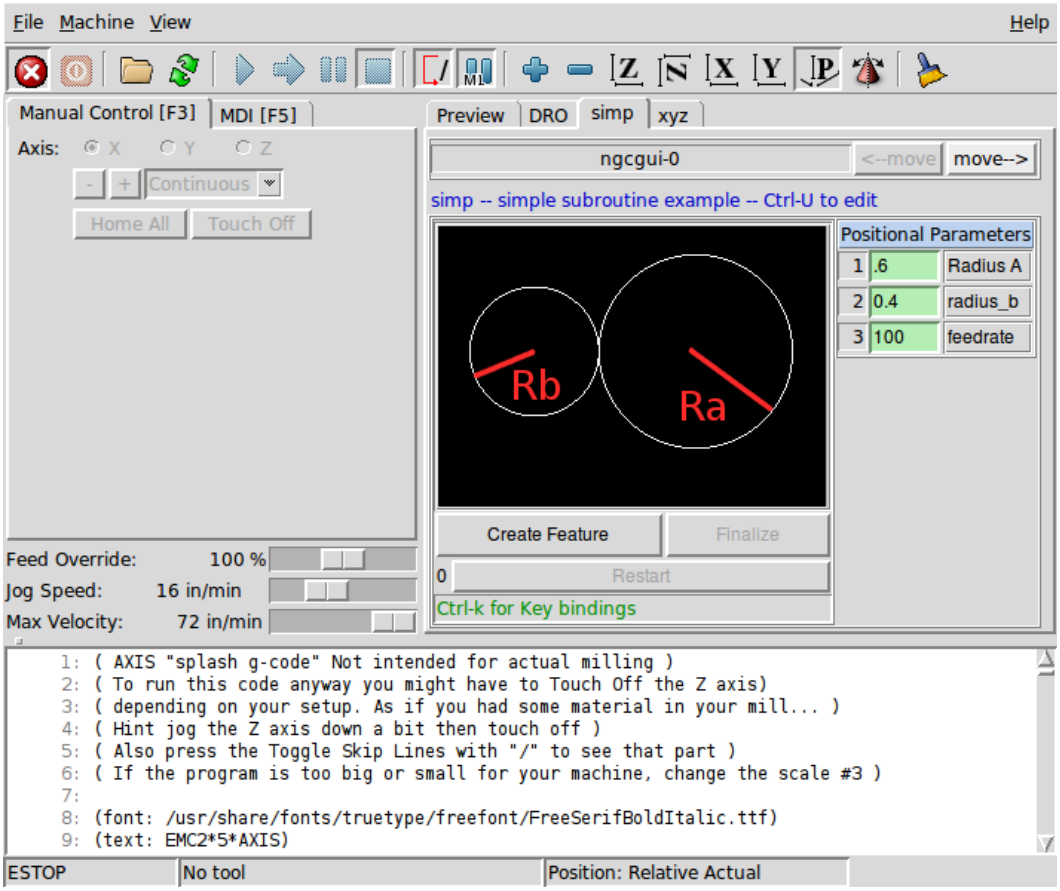
ユーザー設定可能なタッチスクリーン GUI である Gscreen：



GMOCCAPY、Gscreen に基づくタッチスクリーン GUI。 GMOCCAPY は、キーボードとマウスが GUI を制御するための好ましい方法であるアプリケーションでも同様に機能するように設計されています。



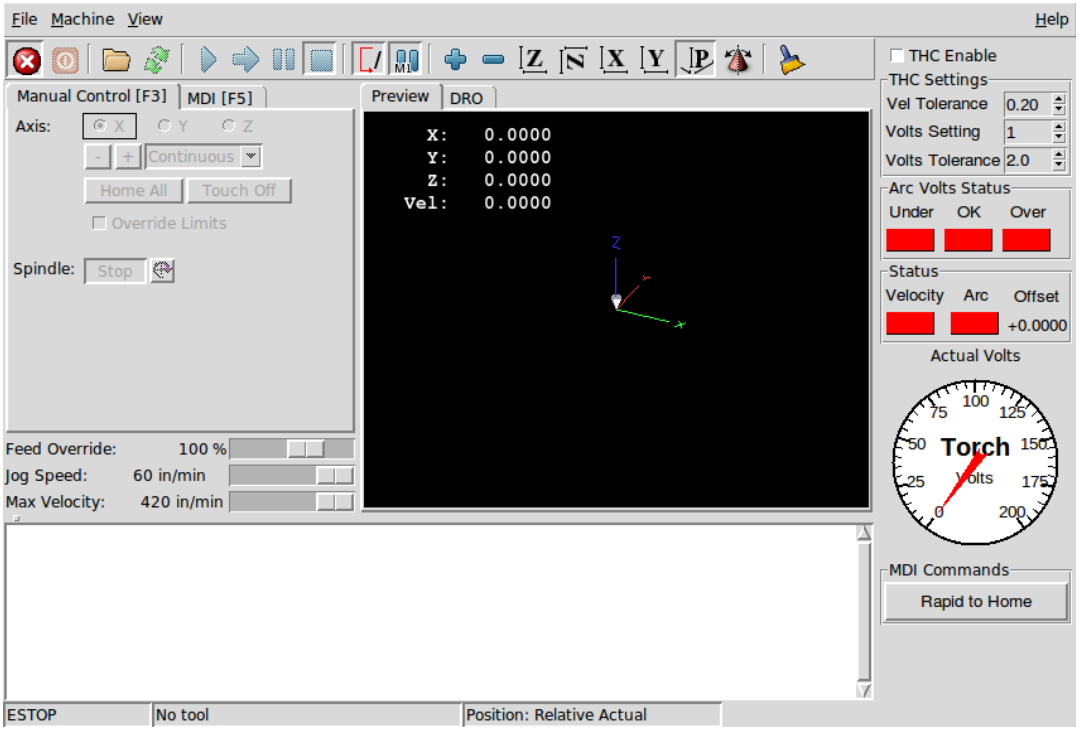
NGCGUI、G コードのウィザードスタイルのプログラミングを提供するサブルーチン GUI。 NGCGUI は、スタンドアロンプログラムとして実行することも、一連のタブとして別の GUI に埋め込むこともできます。 次のスクリーンショットは、Axis に埋め込まれた NGCGUI を示しています。



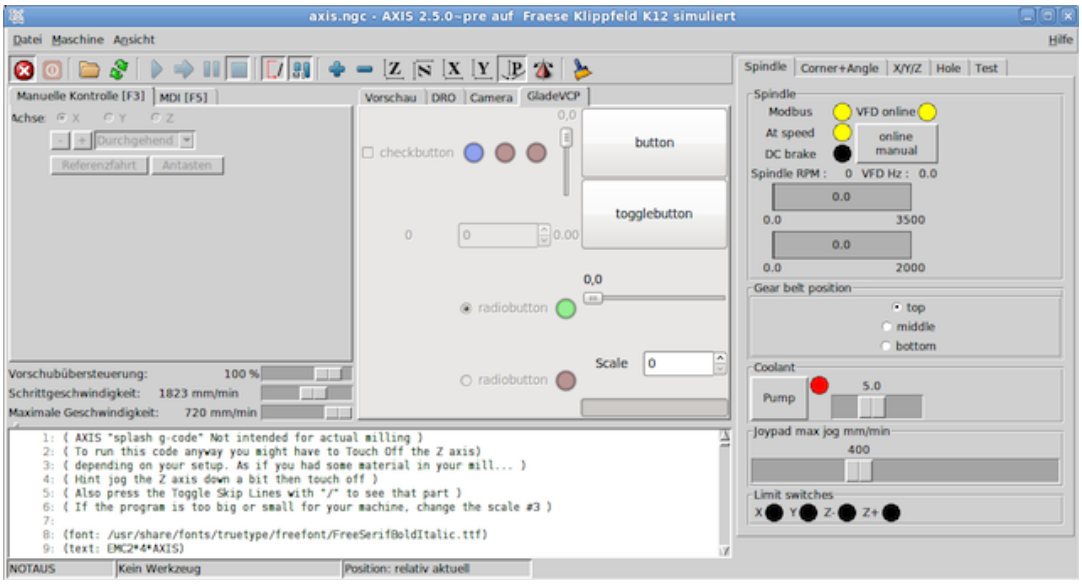
3.2.3 仮想コントロールパネル

上記のように、LinuxCNC の GUI の多くはユーザーがカスタマイズできます。 これは、柔軟性または機能性を向上させるために、GUI の 1 つの基本的な外観にインジケーター、読み出し、スイッチ、またはスライダーを追加するために行うことができます。 LinuxCNC では、2 つのスタイルの仮想コントロールパネルが提供されています。

PyVCP は、AxisGUI に追加できる Python ベースの仮想コントロールパネルです。 PyVCP は、スピンドルアットスピードインジケーターや緊急停止出力信号など、ハードウェアアブストラクションレイヤーに含まれる仮想信号のみを利用し、シンプルな飾り気のない外観を備えています。 これは、ユーザーが最小限の手間で仮想コントロールパネルを追加したい場合に最適です。



GladeVCP は、Axis または TouchyGUI に追加できる Glade ベースの仮想コントロールパネルです。 GladeVCP は、HAL 仮想信号の表示や制御に限定されず、ウィンドウやネットワークイベントなど、LinuxCNC の外部にある他の外部インターフェイスを含めることができるという点で PyVCP よりも優れています。 GladeVCP は、GUI に表示されるように構成する方法もより柔軟です。



3.2.4 言語

LinuxCNC は、翻訳ファイルを使用して、LinuxCNC ユーザーインターフェイスをフランス語、ドイツ語、イタリア語、フィンランド語、ロシア語、ルーマニア語、ポルトガル語、中国語などの多くの言語に翻訳します。 翻訳が作成されていると仮定すると、LinuxCNC は、Linux オペレーティングシステムの

起動時にログインに使用した母国語を自動的に使用します。 言語が翻訳されていない場合は、IRC、メーリングリスト、またはユーザーフォーラムの開発者に連絡してください。

3.2.5 動作モード

LinuxCNC の実行中、コマンドの入力に使用される 3 つの異なる主要モードがあります。 これらは、手動、自動、および手動データ入力（MDI）です。 あるモードから別のモードに変更すると、LinuxCNC コントロールの動作に大きな違いが生じます。 あるモードでは実行できるが、別のモードでは実行できない特定のことがあります。 オペレータは手動モードで軸を原点復帰できますが、自動モードまたは MDI モードでは原点復帰できません。 オペレーターは、自動モードでは G コードでいっぱいのファイル全体を実行することができますが、手動または MDI では実行できません。

手動モードでは、各コマンドは個別に入力されます。 人間の言葉で言えば、手動コマンドは、毎分 25 インチでクーラントをオンにするか X をジョグする場合があります。 これらは、スイッチを切り替えるか、軸のハンドホイールを回すのとほぼ同じです。 これらのコマンドは通常、マウスでボタンを押すか、キーボードのキーを押したままにすることで、グラフィカルインターフェイスの 1 つで処理されます。 自動モードでは、同様のボタンまたはキーを押すことで、ファイルに保存されている G コードのプログラム全体をロードまたは実行を開始できます。 MDI モードでは、オペレーターはコードのブロックを入力し、キーボードの<return>または<enter>キーを押してそれを実行するようにマシンに指示する場合があります。

一部のモーションコントロールコマンドは同時に使用でき、すべてのモードで同じモーションの変化を引き起こします。 これらには、中止、緊急停止、および送り速度のオーバーライドが含まれます。 このようなコマンドは自明である必要があります。

AXIS ユーザーインターフェイスは、ほとんどの場合自動コマンドを使用できるようにすることで、自動モードと他のモードの違いの一部を隠します。 また、タッチオフなどの一部の手動コマンドは実際には MDI コマンドを送信することによって実装されるため、手動と MDI の区別があいまいになります。 これは、ユーザーが要求したアクションに必要なモードに自動的に変更することによって行われます。

3.3 重要なユーザーコンセプト

この章では、g コードを使用して CNC マシンを実行する前に理解しておく必要のある重要なユーザー概念について説明します。

3.3.1 軌道制御

3.3.1.1 軌道計画

一般に、軌道計画は、LinuxCNC が、機械の制限内で動作しながら、G コードプログラムで指定されたパスをたどる手段です。

G コードプログラムに完全に従うことはできません。 たとえば、次の動きを 1 行のプログラムとして指定するとします。

```
G1 X1 F10 (G1 is linear move, X1 is the destination, F10 is the speed)
```

実際には、機械は停止から加速し、X = 1 に向かって移動し、減速して再び停止する必要があるため、F10 で全体の移動を行うことはできません。 移動の一部が F10 で行われることもありますが、多くの移動、特に短い移動では、指定された送り速度にまったく到達しません。 ナイーブカム検出器が G64Pn で使用されていない場合、G コードの動きが短いと、マシンの速度が遅くなり、動きが長くなる可能性があります。

上記の基本的な加速と減速は複雑ではなく、妥協する必要はありません。 INI ファイルでは、最大軸速度や軸加速度などの指定された機械の制約は、軌道プランナーが従う必要があります。

Trajectory Panner ini オプションの詳細については、INI の章の「TrajectorySection」を参照してください。

3.3.1.2 パスフォロー

それほど単純ではない問題は、パス追跡の問題です。 G コードでコーナーをプログラムすると、軌道プランナーはいくつかのことを実行できますが、すべてが正しい場合もあります。コーナーの座標で正確に停止するまで減速してから、新しい方向に加速することができます。 また、コーナーを通過する間、送り速度を維持する、いわゆるブレンディングを実行することもできます。これにより、マシンの制約に従うためにコーナーを丸める必要があります。 ここでトレードオフがあることがわかります。速度を落としてより良いパスフォローを取得するか、速度を上げてより悪いパスフォローを取得することができます。 特定のカット、素材、工具などに応じて、プログラマーは異なる方法で妥協したいと思うかもしれません。

急速な動きも現在の軌道制御に従います。 加速度が低く、パス許容値が指定されていないマシンで最大速度に達するのに十分な長さの移動を使用すると、かなり丸い角を得ることができます。

3.3.1.3 プランナーのプログラミング

軌道制御コマンドは次のとおりです。

- G61-（正確なパスモード）は、次のプログラムされたポイントに方向を変えるために一時的に完全に停止する可能性がある場合でも、プログラムされたポイントに正確にアクセスします。
- G61.1-（完全停止モード）は、プランナーにすべてのセグメントの終わりで正確に停止するように指示します。
- G64-（BlendWithout Tolerance Mode）G64 は、LinuxCNC を起動するときのデフォルト設定です。 G64 はブレンドしているだけで、ナイーブカム検出器は有効になっていません。 G64 および G64P0 は、送り速度を維持するために、パス追従精度を犠牲にするようにプランナーに指示します。 これは、正確な停止が有害である一部のタイプの材料またはツールに必要であり、プログラマーが

ツールのパスがプログラムで指定されているよりもいくらか曲がりくねっていることを覚えている限り、うまく機能します。 G64 で G0（高速）移動を使用する場合は、クリアランス移動に注意し、マシンの加速機能に基づいて障害物をクリアするのに十分な距離を確保してください。

- G64 P- Q-（公差付きブレンドモード）これにより、ナイーブカム検出器が有効になり、公差付きブレンドが可能になります。 G64 P0.05 をプログラムする場合は、連続送りが必要であることをプランナーに伝えますが、プログラムされたコーナーでは、ツールパスがプログラムされたパスの 0.05 ユーザー単位以内にとどまることができるように十分に減速する必要があります。 減速の正確な量は、プログラムされたコーナーの形状と機械の制約によって異なりますが、プログラマーが心配する必要があるのは公差だけです。 これにより、プログラマーは妥協後のパスを完全に制御できます。 ブレンド公差は、必要に応じてプログラム全体で変更できます。 G64 P0 の仕様は、G64 のみ（上記）と同じ効果があることに注意してください。これは、古い G コードプログラムの下位互換性のために必要です。 G コードの章の G64 セクションを参照してください。

- 許容範囲なしのブレンド-制御されたポイントは、指定された各動きに少なくとも 1 つのポイントで接触します。 マシンは、現在の移動（または、ブレンドがすでに開始されているときに一時停止した場合は次の移動）の終了時に正確に停止できないほどの速度で移動することはありません。 移動の終点からの距離は、最適な輪郭送りを維持するために必要なだけ大きくなります。

- ナイーブカム検出器-直線からの逸脱が Q-未満の XYZ 軸のみを含む連続する G1 移動は、単一の直線にマージされます。この統合されたムーブメントは、許容範囲とブレンドする目的で、個々の G1 ムーブメントを置き換えます。連続する動きの間に、制御されたポイントは、動きの実際のエンドポイントから P-を超えて通過しません。制御されたポイントは、各動きの少なくとも 1 つのポイントに接触します。マシンは、現在の移動（またはブレンドがすでに開始されているときに一時停止した場合は次の移動）の終了時に正確に停止できないほどの速度で移動することはありません。G2/3 では、G17（XY）平面で移動します。直線からの円弧の最大偏差が G64Q 許容値よりも小さい場合、円弧は 2 本の線に分割されます（円弧の始点から中点まで、および中点から終点まで）。これらのラインは、ラインのナイーブカムアルゴリズムの対象になります。したがって、ラインアーク、アークアーク、アークラインの場合、およびラインラインは、ナイーブカム検出器の恩恵を受けます。これにより、パスが単純化され、輪郭のパフォーマンスが向上します。

次の図では、青い線は実際の機械の速度を表しています。 赤い線は機械の加速能力です。 各プロットの下の水平線は、計画された移動です。 上のプロットは、次の移動の終わりに正確に停止できるように、機械の加速設定の制限内にとどまるために短い移動が発生したときに、軌道プランナーがどのように機械を減速するかを示しています。 下のプロットは、動きを組み合わせて速度を計画どおりに維持するためのナイーブカム検出器の効果を示しています。

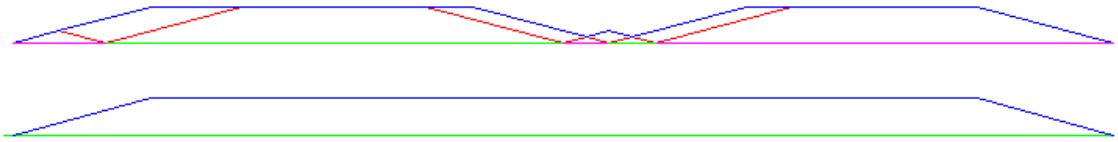


図 3-1

3.3.1.4 計画の動き

動きがあなたの機械/材料に合うのに十分長いことを確認してください。 主に、現在の移動の終了時に完全に停止できないような速度で機械が移動することはないという規則のため、機械が要求された送り速度を維持できる最小の移動長さがあります。 与えられた加速設定。

加速フェーズと減速フェーズはそれぞれ、ini ファイル MAX_ACCELERATION の半分を使用します。正確に反転するブレンドでは、これにより、軸の合計加速度が ini ファイル MAX_ACCELERATION と等しくなります。 その他の場合、実際のマシンの加速は ini ファイルの加速よりもいくらか小さくなります

////これは latexmath なしの以下の段落と重複する段落です。////

送り速度を維持するには、移動は、0 から目的の送り速度まで加速してから再び停止するのにかかる距離よりも長くする必要があります。 A を ini ファイル MAX_ACCELERATION の 1/2 として使用し、F を単位/秒単位の送り速度として使用すると、加速時間は ta = F / A であり、加速距離は da = F * ta / 2 です。 減速時間と距離は同じであり、臨界距離 d = da + dd = 2 * da = F2 / A になります。

たとえば、1 秒あたり 1 インチの送り速度と、10 インチ/秒 2 の加速度の場合、臨界距離は 12/10 = 1/10 = 0.1 インチです。

0.5 インチ/秒の送り速度の場合、臨界距離は 52/100 = 25/100 = 0.025 インチです。

3.3.2 G コード

1.1.1.1 デフォルト

LinuxCNC が最初に起動すると、デフォルトで多くの G および M コードがロードされます。 現在アクティブな G コードと M コードは、AXIS インターフェイスの[Active G-Codes：]ウィンドウの[MDI]タブで確認できます。 これらの G および M コードは、LinuxCNC の動作を定義し、LinuxCNC を実行する前にそれぞれが何をするかを理解することが重要です。 デフォルトは、G コードファイルを実行するときに変更して、LinuxCNC セッションを開始したときとは異なる状態のままにすることができます。 ベストプラクティスは、ジョブに必要なデフォルトを G コードファイルのプリアンブルに設定し、デフォルトが変更されていないと想定しないことです。 G コードクイックリファレンスページを印刷すると、それぞれが何であるかを思い出すのに役立ちます。

3.3.2.1 送り速度

送り速度がどのように適用されるかは、移動に関係する軸が回転軸であるかどうかによって異なります。 回転軸または旋盤を使用している場合は、送り速度のセクションを読んで理解してください。

3.3.2.2 工具半径オフセット

ツール半径オフセット（G41 / 42）では、2 つの隣接する動きを削ることなく、プログラムされた各動きに沿ってツールがどこかに触れることができる必要があります。 現在の工具径でそれが不可能な場合は、エラーが発生します。 直径の小さいツールは、同じパスでエラーなしで実行できます。 これは、カッターよりも狭いパスをエラーなしで通過するようにカッターをプログラムできることを意味します。 詳細については、カッター補正セクションを参照してください。

3.3.3 原点復帰

LinuxCNC を開始した後、プログラムを実行したり MDI コマンドを実行したりする前に、各軸をホームにする必要があります。

マシンにホームスイッチがない場合、各軸の一致マークは、マシンの座標を毎回同じ場所にホーミングするのに役立ちます。

ホームに戻ると、ini ファイルで設定されているソフト制限が使用されます。

デフォルトの動作から逸脱する場合、または Mini インターフェイスを使用する場合は、ini ファイルの [TRAJ]セクションでオプション NO_FORCE_HOMING = 1 を設定する必要があります。 ホーミングの詳細については、インテグレータマニュアルを参照してください。

3.3.4 工具交換

手動で工具を交換する場合、いくつかのオプションがあります。 これらのオプションの構成については、[EMCIO]セクションを参照してください。 G コードの章の G28 および G30 セクションも参照してください。

3.3.5 座標系

座標系は最初は混乱する可能性があります。 CNC マシンを実行する前に、LinuxCNC で使用される座標系の基本を理解する必要があります。 LinuxCNC 座標系の詳細については、このマニュアルの「座標系」セクションを参照してください。

1.1.1.1 機械座標

LinuxCNC をホームにするときは、ホームする軸ごとに G53 マシン座標系を 0 に設定します。

- 他の座標系や工具オフセットは、原点復帰によって変更されません。

G53 マシンの座標系で移動するのは、移動と同じ行に G53 をプログラムするときだけです。 通常、G54 座標系を使用しています。

3.3.5.1 G54-59.3 ユーザー座標

通常、G54 座標系を使用します。 オフセットが現在のユーザー座標系に適用されると、DRO が位置：軸の相対実績を表示しているときに、線の付いた小さな青いボールがマシンの原点になります。 オフセットが一時的なものである場合は、[マシン]メニューのゼロ座標系を使用するか、G コードファイルの最後にある G10 L2 P1 X0 Y0Z0 をプログラムします。 でオフセットをクリアしたい座標系に合うように P 番号を変更します。

- ユーザー座標系に保存されているオフセットは、LinuxCNC がシャットダウンされても保持されます。
- Axis の[タッチオフ]ボタンを使用すると、選択したユーザー座標系のオフセットが設定されます。

3.3.5.2 あなたが失われたとき

必要だと思うときに DRO で 0,0,0 を取得するのに問題がある場合は、いくつかのオフセットがプログラムされている可能性があり、それらを削除する必要があります。

- G53 G0 X0 Y0Z0 で機械原点に移動します
- G92.1 で G92 オフセットをクリアします
- G54 で G54 座標系を使用する
- G54 座標系を G10L2 P1 X0 Y0 Z0R0 の機械座標系と同じに設定します
- G49 で工具オフセットをオフにします
- メニューから相対座標表示をオンにします

これで、マシンの原点 X0 Y0 Z0 にいるはずであり、相対座標系はマシンの座標系と同じである必要があります。

3.3.6 マシン構成

次の図は、ツールの進行方向とミルテーブルおよびリミットスイッチを示す典型的なミルを示しています。 ミルテーブルが、ツール方向の画像で示されているデカルト座標系の矢印と反対方向にどのように移動するかに注目してください。 これにより、ツールがマテリアルに対して正しい方向に移動します。

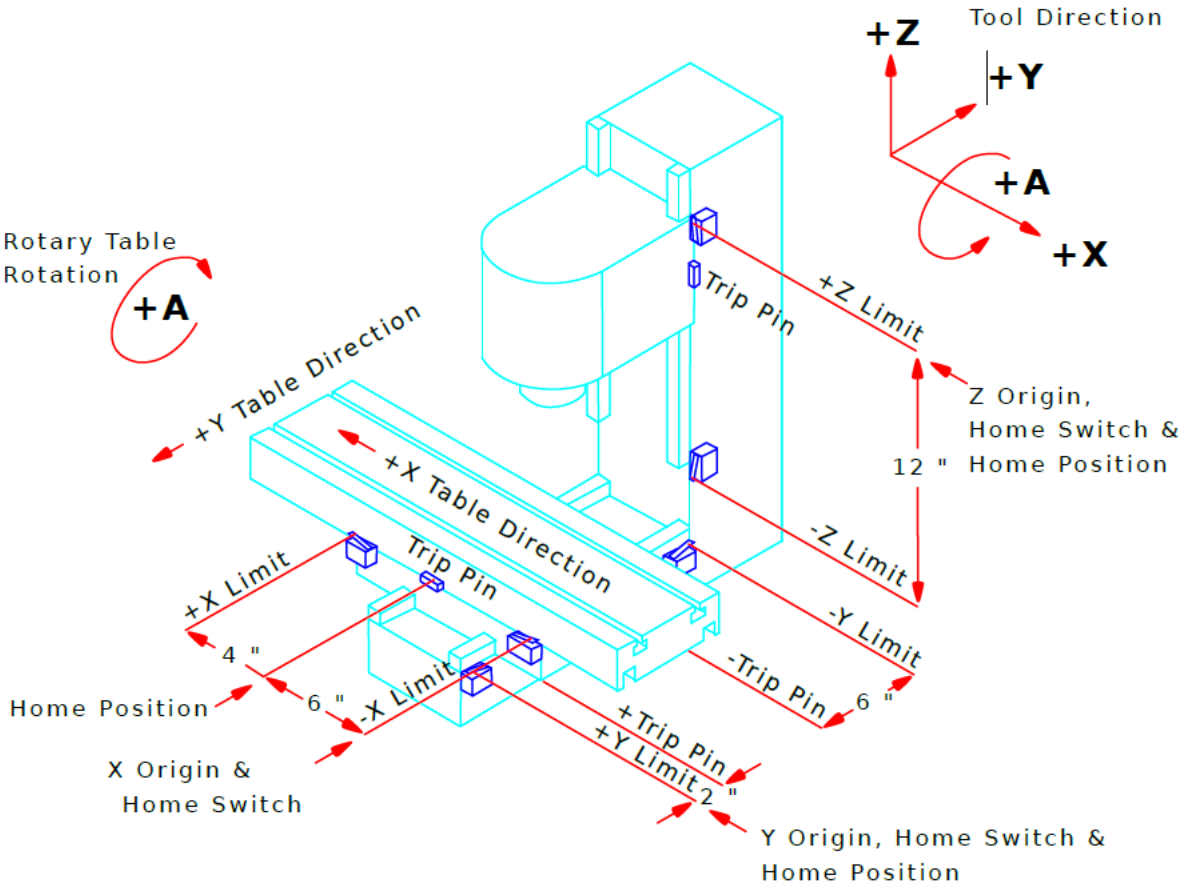


図 3-2

次の図は、工具とリミットスイッチの進行方向を示す典型的な旋盤を示しています。

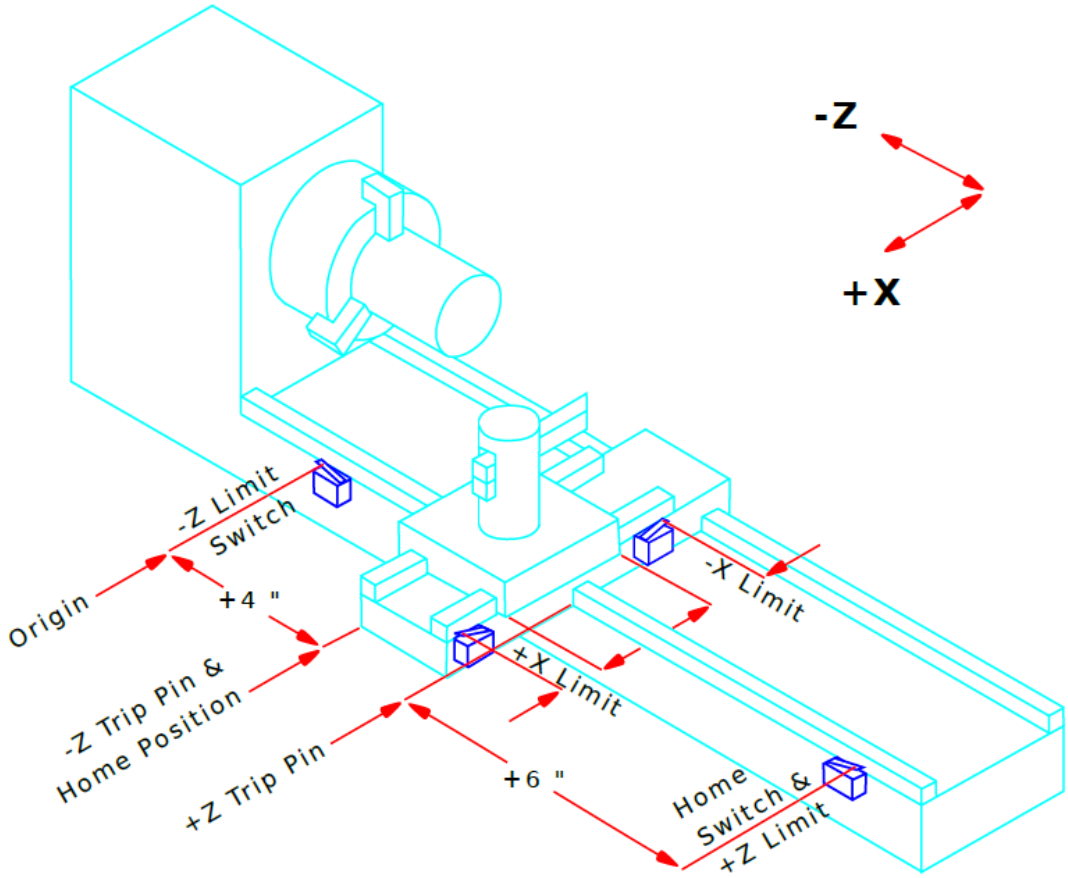


図 3-3

3.4 LinuxCNC の開始

3.4.1 LinuxCNC の実行

LinuxCNC は、スクリプトファイル linuxcnc で起動されます。

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

LINUXCNC スクリプトオプション

- -v = verbose-動作時に情報を出力します
- -d =デバッグ用の画面にスクリプトコマンドをエコーします

linuxcnc スクリプトに ini ファイルが渡されると、ini ファイルが読み取られ、LinuxCNC が起動します。 ini ファイル[HAL]セクションは、複数のファイルが使用されている場合に HAL ファイルをロードする順序を指定します。 HAL = xxx.hal ファイルがロードされると、GUI がロードされ、POSTGUI = .xxx.hal ファイルがロードされます。 HAL ピンを使用して PyVCP または GladeVCP オブジェクトを作成する場

合は、postguiHAL ファイルを使用してこれらのピンに接続する必要があります。 詳細については、INI 構成の[HAL]セクションを参照してください。

3.4.1.1 構成セレクター

linuxcnc スクリプトに ini ファイルが渡されない場合は、構成セレクターが読み込まれるため、サンプル構成を選択して保存できます。 サンプル構成を保存したら、アプリケーションに合わせて変更できます。 設定ファイルは linuxcnc / configs ディレクトリに保存されます。

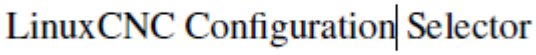


図 3-4

3.5 CNC マシンの概要

このセクションでは、インタープリターの入力端と出力端から CNC マシンをどのように見るかについて簡単に説明します。

3.5.1 機械部品

CNC マシンには、制御される可能性のある、または制御の実行方法に影響を与える可能性のある多くの機械部品があります。 このセクションでは、インタープリターと相互作用するコンポーネントのサブセットについて説明します。 ジョグボタンなど、インタープリターと直接相互作用しない機械部品は、制御に影響を与える場合でも、ここでは説明しません。

1.1.1.1 軸

CNC マシンには 1 つ以上の軸があります。 異なるタイプの CNC マシンには、異なる組み合わせがあります。 たとえば、4 軸フライス盤には XYZA 軸または XYZB 軸がある場合があります。 旋盤には通常 XZ 軸があります。 フォームカッティングマシンは XYUV 軸を備えている場合があります。 LinuxCNC では、1 つの軸に 2 つのモーターを備えた XYYZ ガントリーマシンの場合は、2 番目の線形軸よりも運動学によってより適切に処理されます。 3)

一次線形軸軸一次線形一次線形 X、Y、および Z 軸は、相互に直交する 3 つの方向に線形運動を生成します。

二次線形軸軸二次線形二次線形 U、V、およびワックスは、3 つの相互に直交する方向に線形運動を生成します。 通常、X と U は平行、Y と V は平行、Z と W は平行です。

回転軸軸回転回転 A、B、C 軸は角運動（回転）を生成します。 通常、A は X に平行な線を中心に回転し、B は Y に平行な線を中心に回転し、C は Z に平行な線を中心に回転します。

3)ヘキサポッドマシンのように機械部品の動きが独立していない場合でも、RS274 / NGC 言語と標準的な機械加工機能は、同じ相対速度を生成するために実際のメカニズムを制御する方法を下位レベルの制御が知っている限り、引き続き使用できます。 独立した軸によって生成されるツールとワークピースの動き。これはキネマティクスと呼ばれます。

3.5.1.1 スピンドル

CNC マシンには通常、1 つの切削工具、プローブ、または旋盤の場合は材料を保持するスピンドルがあります。 スピンドルは、CNC ソフトウェアによって制御される場合とされない場合があります。 LinuxCNC は、最大 8 つのスピンドルをサポートします。これらのスピンドルは、個別に制御でき、さまざまな速度でさまざまな方向に同時に実行できます。

3.5.1.2 クーラント

CNC マシンにミストクーラントおよび/またはフラッドクーラントを提供するコンポーネントがある場合、それらは G コードで制御できます。

3.5.1.3 フィードと速度のオーバーライド

CNC マシンには、個別の送り制御と速度オーバーライド制御を設定できます。これにより、オペレータは、プログラムされた速度の一定の割合で加工に使用される実際の送り速度またはスピンドル速度を指定できます。

3.5.1.4 ブロック削除スイッチ

CNC マシンはブロック削除スイッチを持つことができます。 ブロック削除セクションを参照してください。

3.5.1.5 オプションのプログラム停止スイッチ

CNC マシンには、オプションのプログラム停止スイッチを付けることができます。 オプションのプログラム停止セクションを参照してください。

3.5.2 制御およびデータコンポーネント

1.1.1.1 直線軸

X、Y、および Z 軸は、直交する線形軸の標準的な右手座標系を形成します。 3 つの線形運動メカニズムの位置は、 これらの軸上の座標を使用して表されます。

U、V、およびワックスも、標準の右手座標系を形成します。 X と U は平行、Y と V は平行、Z と W は平行です（A、B、C がゼロに回転した場合）。

3.5.2.1 回転軸

回転軸は、対応する X、Y、または Z 軸の正の端から見て、正の回転の方向が反時計回りであるラップされた直線軸として度単位で測定されます。 ラップされた直線軸とは、軸が反時計回りに回転すると角位置が無制限に増加し（プラス無限大に向かう）、軸が時計回りに回転すると角度位置が無制限に減少

する（マイナス無限大に向かう）軸を意味します。 回転に機械的な制限があるかどうかに関係なく、ラップされた直線軸が使用されます。

時計回りまたは反時計回りは、ワークピースの観点からです。 ワークピースが回転軸を回転するターンテーブルに固定されている場合、ワークピースの観点から反時計回りに回転するには、ターンテーブルを（最も一般的な機械構成の場合）時計回りに回転させます。 機械の隣に立っている人の 4)

---

4)並列処理の要件に違反した場合、システムビルダーは時計回りと反時計回りを区別する方法を説明する必要があります。

3.5.2.2 制御点

値）、これはスピンドル軸上のポイント（ゲージポイントと呼ばれることが多い）であり、スピンドルの端を超えて一定の距離にあり、通常はスピンドルに適合するツールホルダーの端の近くにあります。 工具長オフセットに正の量を指定することにより、制御点の位置を主軸軸に沿って移動できます。 この量は通常、使用中の切削工具の長さであるため、制御点は切削工具の端にあります。 旋盤では、工具長オフセットを X 軸と Z 軸に指定でき、制御点は工具先端またはその少し外側（工具の前面と側面が接触する軸に沿った垂直線が交差する場所）にあります。 。

3.5.2.3 協調線形運動

指定されたパスに沿って工具を駆動するには、マシニングセンターが複数の軸の動きを調整する必要があります。 協調線形運動という用語を使用して、名目上、各軸が一定の速度で移動し、すべての軸が開始位置から終了位置に同時に移動する状況を説明します。 X、Y、および Z 軸（またはそれらのいずれか 1 つまたは 2 つ）のみが移動する場合、これは直線の動きを生成します。したがって、この用語では線形という言葉が使用されます。 実際のモーションでは、モーションの開始時や終了時に加速または減速が必要になるため、一定の速度を維持できないことがよくあります。 ただし、各軸が他の軸と同じ必要な動作の一部を常に完了するように軸を制御することは可能です。 これにより、ツールが同じパスに沿って移動します。この種のモーション協調線形モーションとも呼ばれます。

調整された線形運動は、一般的な送り速度またはトラバース速度のいずれかで実行できます。または、スピンドルの回転に同期させることもできます。 軸速度の物理的な制限により目的の速度が得られない場合は、目的のパスを維持するためにすべての軸の速度が低下します。

3.5.2.4 送り速度

制御点が移動する速度は、名目上、ユーザーが設定できる一定の速度です。 インタープリターでは、送り速度は次のように解釈されます（逆時間送りまたは 1 回転あたりの送りモードが使用されている場合を除きます。その場合はセクション G93-G94-G95-モードを参照してください）。

1. XYZ のいずれかが移動している場合、F は XYZ デカルトシステムで 1 分あたりの単位であり、他のすべての軸（ABCUVW）は、協調して開始および停止するように移動します。

2. それ以外の場合、UVW のいずれかが移動している場合、F は UVW デカルトシステムで 1 分あたりの単位であり、他のすべての軸（ABC）は、調整された方法で開始および停止するように移動します。
3. それ以外の場合、移動は純粋な回転運動であり、F ワードは ABC 疑似デカルトシステムの回転単位です。

3.5.2.5 クーラント

フラッドクーラントとミストクーラントはそれぞれ個別にオンにすることができます。 RS274 / NGC 言語は、それらを一緒にオフにします。セクション M7 M8M9 を参照してください。

3.5.2.6 ドエル

マシニングセンターは、特定の時間滞留するように（つまり、すべての軸を動かさないように）命令することができます。 ドウェルの最も一般的な用途は、切りくずを壊して取り除くことです。そのため、通常、ドウェル中にスピンドルが回転します。 パス制御モード（「パス制御」のセクションを参照）に関係なく、マシンは、正確なパスモードであるかのように、前にプログラムされた移動の最後に正確に停止します。

3.5.2.7 単位

X、Y、および Z 軸に沿った距離に使用される単位は、ミリメートルまたはインチで測定できます。 機械制御に関係する他のすべての数量の単位は変更できません。 数量が異なれば、特定の単位も異なります。 スピンドル速度は、1 分あたりの回転数で測定されます。 回転軸の位置は度で測定されます。 セクション G93G94 G95 で説明されているように、送り速度は、現在の長さの単位/分、度/分、またはスピンドルの回転あたりの長さの単位で表されます。

3.5.2.8 現在位置

制御点は常に現在位置と呼ばれる場所にあり、コントローラーはそれがどこにあるかを常に認識しています。 いくつかのイベントのいずれかが発生した場合、軸の動きがない場合は、現在の位置を表す数値を調整する必要があります。

1. 長さの単位が変更されます。
2. 工具長オフセットが変更されました
3. 座標系のオフセットが変更されます。

3.5.2.9 選択された平面

常に選択された平面があります。これは、マシニングセンターの XY 平面、YZ 平面、または XZ 平面である必要があります。 もちろん、Z 軸は XY 平面に垂直であり、X 軸は YZ 平面に垂直であり、Y 軸は XZ 平面に垂直です。

3.5.2.10 ツールカルーセル

ツールカルーセルの各スロットには、ゼロまたは 1 つのツールが割り当てられます。

3.5.2.11 ツール交換

マシニングセンターは工具交換を命じられる場合があります。

3.5.2.12 パレットシャトル

2 つのパレットはコマンドで交換できます。

3.5.2.13 パス制御モード

マシニングセンターは、（1）正確な停止モード、（2）正確なパスモード、または（3）オプションの許容誤差のある連続モードの 3 つのパス制御モードのいずれかに設定できます。 正確な停止モードでは、プログラムされた各移動の終了時にマシンが一時的に停止します。 正確なパスモードでは、マシンはプログラムされたパスを可能な限り正確にたどり、必要に応じてパスの鋭い角で減速または停止します。 連続モードでは、送り速度を維持できるように、パスの鋭い角をわずかに丸めることができます（ただし、指定されている場合は、許容誤差を超えないようにします）。 セクション G61 および G64 を参照してください。

3.5.3 スイッチとのインタプリタの相互作用

インタープリターはいくつかのスイッチと相互作用します。 このセクションでは、相互作用について詳しく説明します。 いかなる場合でも、通訳者はこれらのスイッチの設定が何であるかを知りません。

1.1.1.1 フィードおよび速度オーバーライドスイッチ

インタプリタは、フィードおよび速度オーバーライドスイッチの M48 を有効または無効にする RS274 / NGC コマンドを解釈します。 スレッドサイクル中のスレッドの終わりからのトラバースなどの特定の移動では、スイッチは自動的に無効になります。

LinuxCNC は、これらのスイッチが有効になっている場合、速度とフィードオーバーライドの設定に反応します。

詳細については、「M48M49 オーバーライド」セクションを参照してください。

3.5.3.1 ブロック削除スイッチ

ブロック削除スイッチがオンの場合、スラッシュ（ブロック削除文字）で始まる G コードの行は解釈されません。 スイッチがオフの場合、そのような行が解釈されます。 通常、NGC プログラムを開始する前に、ブロック削除スイッチを設定する必要があります。

3.5.3.2 オプションのプログラム停止スイッチ

このスイッチがオンで、M1 コードが検出されると、プログラムの実行が一時停止されます。

3.5.4 ツールテーブル

インタープリターを使用するには、ツールテーブルが必要です。 このファイルには、どのツールがどのツールチェンジャースロットにあるか、および各ツールのサイズとタイプが示されています。 ツールテーブルの名前は、ini ファイルで定義されています。

```
[EMCIO]
# tool table file
TOOL_TABLE = tooltable.tbl
```

デフォルトのファイル名はおそらく上記のようになりますが、ini ファイルと同じ名前を使用し、拡張子を tbl にして、マシンに独自のツールテーブルを付けることをお勧めします。

```
TOOL_TABLE = acme_300.tbl
```

また

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

ツールテーブル形式の詳細については、ツールテーブル形式のセクションを参照してください。

3.5.5 パラメーター

RS274 / NGC 言語ビューでは、マシニングセンターは、システム定義（RS274NGC_MAX_PARAMETERS）によって定義された数値パラメータの配列を維持します。 それらの多くは、特に座標系の定義に特定の用途があります。 開発により新しいパラメータのサポートが追加されると、数値パラメータの数が増える可能性があります。 マシニングセンターの電源がオフになっていても、パラメータ配列は時間の経過とともに持続します。 LinuxCNC は、パラメーターファイルを使用して永続性を確保し、インタープリターにファイルを維持する責任を与えます。 インタプリタは、起動時にファイルを読み取り、終了時にファイルを書き込みます。

すべてのパラメータは、G コードプログラムで使用できます。

パラメータファイルの形式を次の表に示します。 このファイルは、任意の数のヘッダー行、1つの空白行、および任意の数のデータ行で構成されます。 インタープリターはヘッダー行をスキップします。データの前に空白行が1つだけあることが重要です（スペースやタブもありません）。 次の表に示すヘッダー行はデータ列を説明しているため、その行を常にヘッダーに含めることをお勧めします（必須ではありません）。

インタプリタは、テーブルの最初の2列のみを読み取ります。 3番目の列であるCommentは、インタプリタによって読み取られません。

ファイルの各行には、最初の列にパラメーターのインデックス番号が含まれ、2番目の列にそのパラメーターを設定する値が含まれています。値は、インタープリター内で倍精度浮動小数点数として表されますが、ファイルに小数点は必要ありません。次の表に示すすべてのパラメータは必須パラメータであり、未使用軸の回転軸値を表すパラメータを省略できる場合を除いて、パラメータファイルに含める必要があります。必要なパラメータが欠落している場合は、エラーが通知されます。パラメータファイルには、その番号が1〜5400の範囲にある限り、他のパラメータを含めることができます。パラメータ番号は昇順で並べる必要があります。そうでない場合は、エラーが通知されます。インタプリタによって読み取られたファイルに含まれるパラメータはすべて、インタプリタが終了するときに書き込むファイルに含まれます。元のファイルは、新しいファイルが書き込まれるときにバックアップファイルとして保存されます。ファイルの書き込み時にコメントは保持されません。

表3.1：パラメータファイル形式

パラメータ番号	パラメータ値	コメント
5161	0.0	G28 Home X
5162	0.0	G28 Home Y

詳細については、「パラメーター」セクションを参照してください。

3.6 LinuxCNCの実行

3.6.1 LinuxCNCの呼び出し

インストール後、LinuxCNCは他のLinuxプログラムと同じように起動します。コマンドlinuxcncを発行してターミナルから実行するか、[アプリケーション]-[CNC]メニューで選択します。

3.6.2 構成ランチャー

[CNC]メニューまたはiniファイルを指定せずにコマンドラインからLinuxCNCを起動すると、[構成セレクタ]ダイアログが起動します。

[構成セレクタ]ダイアログを使用すると、ユーザーは既存の構成（[マイ構成]）の 1 つを選択するか、（サンプル構成から）新しい構成を選択してホームディレクトリにコピーできます。 コピーされた構成は、構成セレクターを次に呼び出したときに[マイ構成]の下に表示されます。

構成セレクターは、編成された構成の選択を提供します。

- 私の構成-〜 / linuxcnc / configs にあるユーザー構成
- サンプル構成-選択すると、サンプル構成が〜/ linuxcnc / configs にコピーされます。 ランチャーを使用する場合は、サンプル構成をコピーしたら、[マイ構成]から選択します
  - sim-シミュレートされたハードウェアを含む構成。 これらは、LinuxCNC がどのように機能するかをテストまたは学習するために使用できます。
  - by_interface-GUI によって編成された構成。
  - by_machine-マシンごとに編成された構成。
  - アプリ-linuxcnc を起動する必要はありませんが、PyVCP や GladeVCP などのアプリケーションのテストや試行に役立つ場合があります。
  - attic-廃止された構成または履歴構成。

sim 構成は、多くの場合、新規ユーザーにとって最も有用な開始点であり、サポートされている GUI を中心に構成されています。

- 軸-キーボードとマウスの GUI
- gmoccapy-タッチスクリーン GUI
- gscreen-タッチスクリーン GUI
- low_graphics-キーボード GUI
- tklinuxcnc-キーボードとマウスの GUI（保守されなくなりました）
- touchy-タッチスクリーン GUI

GUI 構成ディレクトリには、特別な状況や他のアプリケーションの埋め込みを示す構成のサブディレクトリが含まれている場合があります。

by_interface 構成は、次のような一般的なサポートされているインターフェイスを中心に構成されています。

- 一般的なメカトロニクス
- Mesa
- Parport
- Pico
- Pluto

- Servotogo
- Vigilant
- vitalsystems

これらの構成をシステムの開始点として使用するには、関連するハードウェアが必要になる場合があります。

by_machine 構成は、次のような完全な既知のシステムを中心に構成されています。

- boss
- cooltool
- sherline
- smithy
- tormach

これらの構成を使用するには、完全なシステムが必要になる場合があります

アプリのアイテムは通常、1）linuxcnc を起動する必要のないユーティリティ、または 2）linuxcnc で使用できるアプリケーションのデモンストレーションです。

- info-問題の診断に役立つ可能性のあるシステム情報を含むファイルを作成します。
- gladevcp-gladevcp アプリケーションの例。
- halrun-ターミナルで halrun を開始します。
- レイテンシー-レイテンシーを調査するアプリケーション
  - レイテンシーテスト-標準テスト
  - レイテンシープロット-ストリップチャート
  - レイテンシ-ヒストグラム-ヒストグラム
- parport-parport をテストするためのアプリケーション
- pyvcp-pyvcp アプリケーションの例。
- xhc-hb04-xhc-hb04USB ワイヤレス MPG をテストするためのアプリケーション

Note

Apps ディレクトリでは、ユーザーが便利に変更したアプリケーションのみがユーザーのディレクトリにコピーできます。

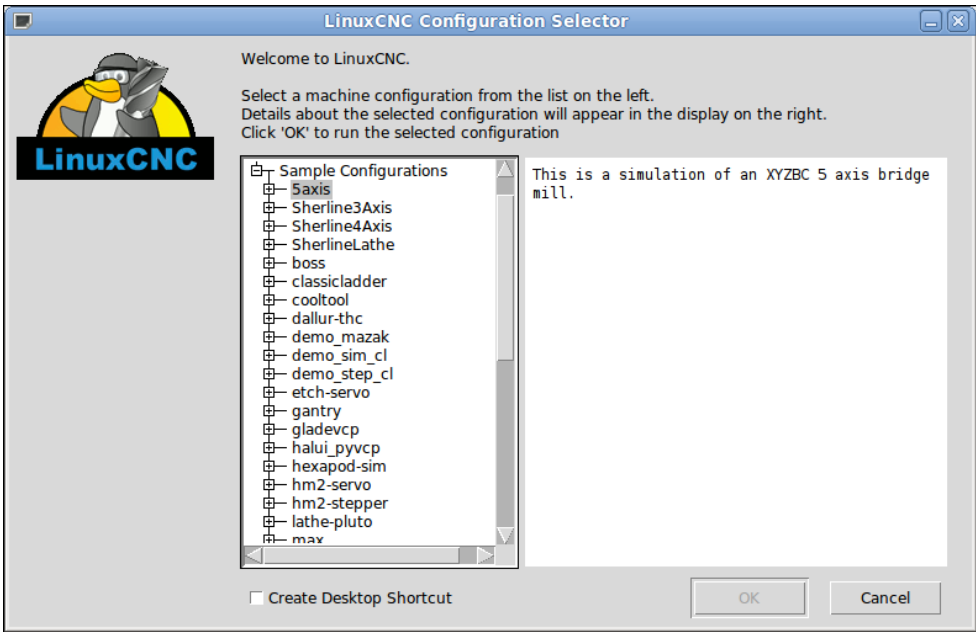


図 3-5

リストされている構成のいずれかをクリックして、それに関する特定の情報を表示します。 構成をダブルクリックするか、[OK]をクリックして構成を開始します。 [デスクトップショートカットの作成]を選択し、[OK]をクリックして Ubuntu デスクトップにアイコンを追加し、[構成セレクター]画面を表示せずにこの構成を直接起動します。

「サンプル構成」セクションから構成を選択すると、その構成のコピーが linuxcnc / configs ディレクトリーに自動的に配置されます。

3.6.3 構成の次のステップ

マシンと同じインターフェイスハードウェア（またはシミュレーター構成）を使用するサンプル構成を見つけ、コピーをホームディレクトリに保存した後、マシンの詳細に応じてカスタマイズできます。 構成のトピックについては、インテグレーターのマニュアルを参照してください。

3.6.4 シミュレーター構成

サンプル構成/ sim の下にリストされているすべての構成は、任意のコンピューターで実行することを目的としています。 特定のハードウェアは必要なく、リアルタイムのサポートも必要ありません。

これらの構成は、個々の機能またはオプションを調査するのに役立ちます。 sim 構成は、デモンストレーションで使用されたグラフィカルユーザーインターフェイスに従って編成されています。 axis のディレクトリには、最もテストされた GUI であるため、ほとんどの選択肢とサブディレクトリが含まれています。 特定の GUI で示される機能は、他の GUI でも使用できる場合があります。

3.6.5 構成リソース

構成セレクターは、構成に必要なすべてのファイルを〜/ linuxcnc / configs の新しいサブディレクトリー（同等に：/ home / username / linuxcnc / configs）にコピーします。 作成された各ディレクトリには、特定の構成を記述するために使用される少なくとも 1 つの ini ファイル（iniflename.ini）が含まれます。

コピーされた directroy 内のファイルリソースには、通常、関連する構成用の 1 つ以上の ini ファイル（filename.ini）とツールテーブルファイル（toolfilename.tbl）が含まれます。 さらに、リソースには、ハーフイル（filename.hal、filename.tcl）、ディレクトリを説明するための README ファイル、および特定の構成にちなんで名付けられたテキストファイル（inifilename.txt）の構成固有の情報が含まれる場合があります。 後者の 2 つのファイルは、構成セレクターを使用すると表示されます。

提供されているサンプル構成では、システムの Halfile ライブラリにあるため、コピーされたディレクトリに存在しない HALFILE が構成 ini ファイルに指定されている場合があります。 これらのファイルは、ユーザー構成ディレクトリにコピーして、変更またはテストのためにユーザーの必要に応じて変更できます。 Halfiles を見つけるときに最初にユーザー構成ディレクトリが検索されるため、ローカルでの変更が優先されます。

構成セレクターは、システムの Halfile ライブラリーを指すシンボリックリンクをユーザー構成ディレクトリ（hallib という名前）に作成します。 このリンクにより、ライブラリファイルのコピーが簡単になります。 たとえば、ローカルで変更を加えるためにライブラリ core_sim.hal ファイルをコピーするには、次のようにします。

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

3.7 ステッパー構成ウィザード

3.7.1 序章

LinuxCNC は、さまざまなハードウェアインターフェイスを使用して、さまざまな機械を制御できます。

Stepconf は、特定のクラスの CNC マシン用の LinuxCNC の構成ファイルを生成するプログラムです。これらのファイルは、標準のパラレルポートを介して制御され、step＆direction タイプの信号によって制御されます。

Stepconf は、LinuxCNC のインストール時にインストールされ、CNC メニューにあります。

Stepconf は、作成した各構成の選択肢を保存するファイルを linuxcnc / config ディレクトリに配置します。 何かを変更するときは、構成名に一致するファイルを選択する必要があります。 ファイル拡張子は.stepconf です。

Stepconf Wizard は、少なくとも 800 x600 の画面解像度で最適に動作します。

3.7.2 スタートページ

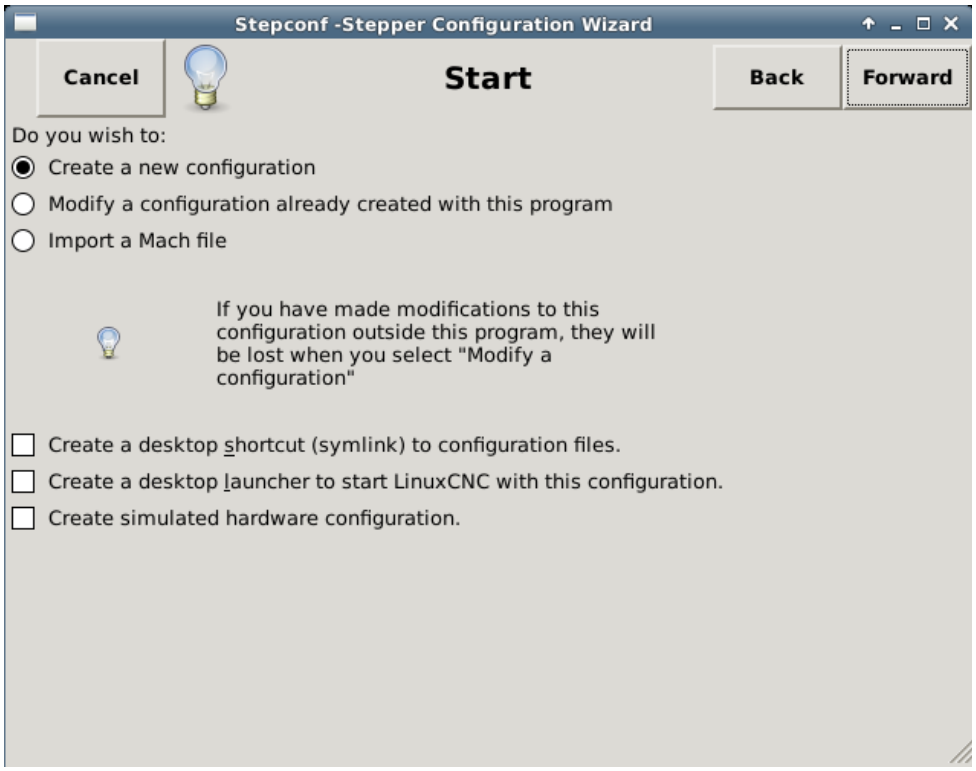


図 3-6

- 新規作成-新しい構成を作成します。
- 変更-既存の構成を変更します。 これを選択すると、ファイルピッカーがポップアップ表示されるので、変更する.stepconf ファイルを選択できます。 メインの.hal または.ini ファイルに変更を加えた場合、これらは失われます。 custom.hal および custom_postgui.hal への変更は、Stepconf ウィザードによって変更されません。 Stepconf は、ビルドされた lastconf を強調表示します。
- インポート-Mach 構成ファイルをインポートし、それを linuxcnc 構成ファイルに変換してみます。 インポート後、stepconf のページに移動して、エントリを確認/変更します。 元の machxml ファイルは変更されません。

これらの次のオプションは、Stepconf の次の実行のために設定ファイルに記録されます。

- デスクトップショートカットの作成-これにより、デスクトップにファイルへのリンクが配置されます。
- デスクトップランチャーの作成-これにより、デスクトップにランチャーが配置され、アプリケーションが起動します。
- シミュレートされたハードウェアの作成-これにより、実際のハードウェアがない場合でも、テスト用の構成を構築できます。

3.7.3 基本情報

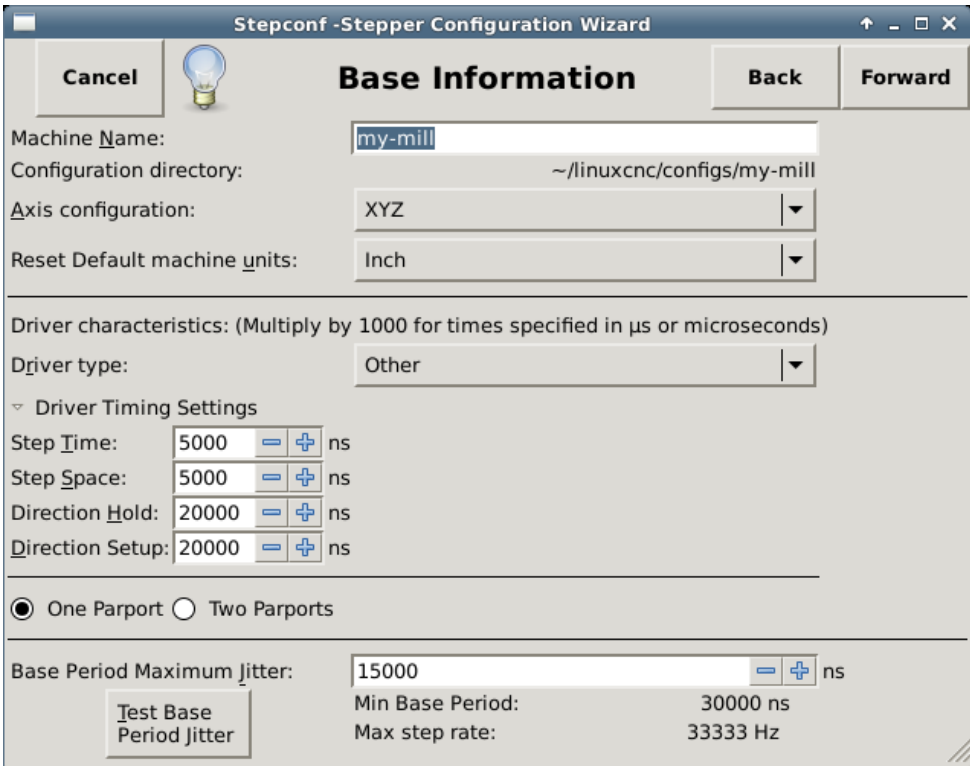


図 3-7

- マシン名-マシンの名前を選択します。 大文字、小文字、数字、-および_のみを使用してください。
- 軸構成-XYZ（ミル）、XYZA（4 軸ミル）、または XZ（旋盤）を選択します。
- 機械単位-インチまたは mm を選択します。 以降のすべてのエントリは、選択した単位になります。 これを変更すると、Axes セクションのデフォルト値も変更されます。 いずれかの軸セクションで値を選択した後でこれを変更すると、選択した単位のデフォルト値で上書きされます。
- ドライバーの種類-プルダウンボックスにステッパードライバーの 1 つが表示されている場合は、それを選択します。 それ以外の場合は、[その他]を選択し、ドライバーのデータシートでタイミング値を見つけて、[ドライバータイミング設定]にナノ秒として入力します。 データシートにマイクロ秒単位の値が記載されている場合は、1000 を掛けます。たとえば、4.5us を 4500ns と入力します。

いくつかの人気のあるドライブのリストとそのタイミング値は、LinuxCNC.orgWiki の StepperDriveTiming にあります。

ブレークアウトボード上のオプトカプラーや RC フィルターなどの追加の信号調整または分離は、ドライバーのタイミング制約に加えて、独自のタイミング制約を課す可能性があります。 これを可能にするために、ドライブ要件に時間を追加する必要がある場合があります。

LinuxCNC 構成セレクターには、Sherline の構成が既に構成されています。

- ステップ時間-ステップパルスがオンになっている時間（ナノ秒単位）。 この設定についてよくわからない場合は、20,000 の値がほとんどのドライブで機能します。

- ステップスペース-ナノ秒単位のステップパルス間の最小時間。 この設定についてよくわからない場合は、20,000 の値がほとんどのドライブで機能します
- 方向保持-方向を変更した後、方向ピンが保持される時間（ナノ秒単位）。 この設定についてよくわからない場合は、20,000 の値がほとんどのドライブで機能します。
- 方向の設定-ナノ秒単位の最後のステップパルスの後、方向が変わるまでの時間。 この設定についてよくわからない場合は、20,000 の値がほとんどのドライブで機能します。
- 1/2 パーポート-構成するパラレルポートの数を選択します。
- 基本期間の最大ジッター-レイテンシーテストの結果をここに入力します。 レイテンシーテストを実行するには、Test Base PeriodJitter ボタンを押します。 詳細については、レイテンシテストのセクションを参照してください。
- 最大ステップレート-Stepconf は、入力されたドライバーの特性と遅延テストの結果に基づいて、最大ステップレートを自動的に計算します。
- 最小基準期間-Stepconf は、入力されたドライバーの特性と遅延テストの結果に基づいて、最小基準期間を自動的に決定します。

3.7.4 レイテンシーテスト

テストの実行中は、コンピューターを悪用する必要があります。 画面上でウィンドウを移動します。 ウェブをサーフィンします。 ディスク上のいくつかの大きなファイルをコピーします。 音楽を再生します。 glxgears などの OpenGL プログラムを実行します。 アイデアは、レイテンシーテストが最悪のケースの数が何であるかを確認する間、PC をそのペースに乗せることです。 少なくとも数分テストを実行します。 テストを実行する時間が長いほど、頻度の低い間隔で発生する可能性のあるイベントをより適切にキャッチできます。 これはコンピュータ専用のテストであるため、テストを実行するためにハードウェアを接続する必要はありません。

警告

遅延テストの実行中は、LinuxCNC の実行を試みないでください。

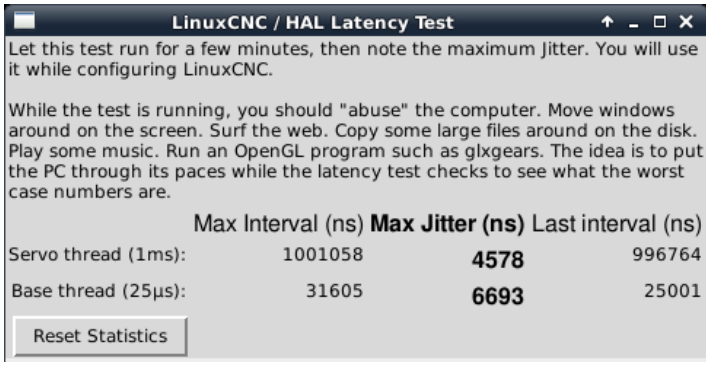


図 3-8

レイテンシーとは、PC が実行中の処理を停止し、外部要求に応答するのにかかる時間です。 この場合、要求はステップパルスのタイミング基準として機能する周期的なハートビートです。 レイテンシーが低いほど、ハートビートをより速く実行でき、ステップパルスがより速くよりスムーズになります。

レイテンシーは CPU 速度よりもはるかに重要です。 レイテンシを決定する要因は CPU だけではありません。マザーボード、ビデオカード、USB ポート、SMI の問題、およびその他の多くのものが遅延を損なう可能性があります。

SMI 問題のトラブルシューティング（LinuxCNC.orgWiki）

Ubuntu の SMI によって引き起こされるリアルタイムの問題を修正する

http://wiki.linuxcnc.org/cgi-bin/wiki.pl?FixingSMIIssues

重要な数値は最大ジッターです。 上記の例では、9075 ナノ秒、つまり 9.075 マイクロ秒が最大のジッターです。この数値を記録し、[基本期間の最大ジッター]ボックスに入力します。

最大ジッタ数が約 15～20 マイクロ秒（15000～20000 ナノ秒）未満の場合、コンピュータはソフトウェアステッピングで非常に優れた結果をもたらすはずです。 最大レイテンシが 30～50 マイクロ秒のような場合でも、良好な結果を得ることができますが、特にマイクロステッピングを使用している場合や非常に細いピッチの親ねじがある場合は、最大ステップレートが少し期待外れになる可能性があります。 数値が 100us 以上（100,000 ナノ秒）の場合、PC はソフトウェアステッピングの候補としては適していません。 1 ミリ秒（1,000,000 ナノ秒）を超える数値は、ソフトウェアステッピングを使用するかどうかに関係なく、PC が LinuxCNC の適切な候補ではないことを意味します。

3.7.5 パラレルポートの設定

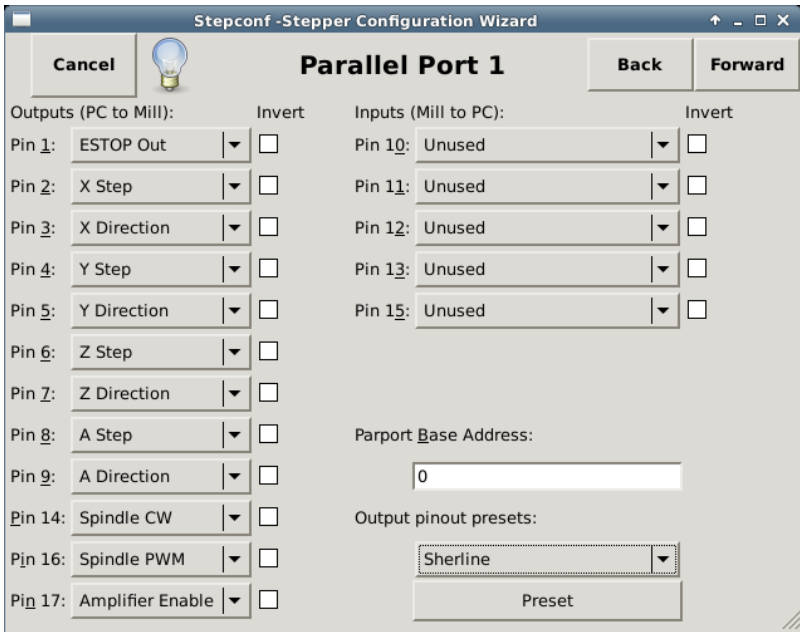


図 3-9

アドレスは、16 進数（多くの場合 0x378）または Linux のデフォルトのポート番号（おそらく 0）として指定できます。

各ピンについて、パラレルポートのピン配置に一致する信号を選択します。 信号が反転している場合は、反転チェックボックスをオンにします（true /アクティブの場合は 0V、false /非アクティブの場合は 5V）。

- 出力ピン配置プリセット-Sherline 標準（ピン 2、4、6、8 の方向）または Xylotex 標準（ピン 3、5、7、9 の方向）に従ってピン 2 から 9 を自動的に設定します。
- 入力と出力-入力または出力が使用されていない場合は、オプションを未使用に設定します。
- 外部非常停止-これは、入力ピンのドロップダウンボックスから選択できます。 一般的な非常停止チェーンは、通常は閉じているすべての接点を使用します。
- ホーミングおよびリミットスイッチ-これらは、ほとんどの構成の入力ピンドロップダウンボックスから選択できます。
- チャージポンプ-ドライバボードでチャージポンプ信号が必要な場合は、チャージポンプ入力に接続する出力ピンのドロップダウンリストから[チャージポンプ]を選択します。 チャージポンプ出力は、Stepconf によってベーススレッドに接続されます。 チャージポンプの出力は、[基本的なマシン構成]ページに表示されている最大ステップレートの約 1/2 になります。

3.7.6 パラレルポート 2 のセットアップ

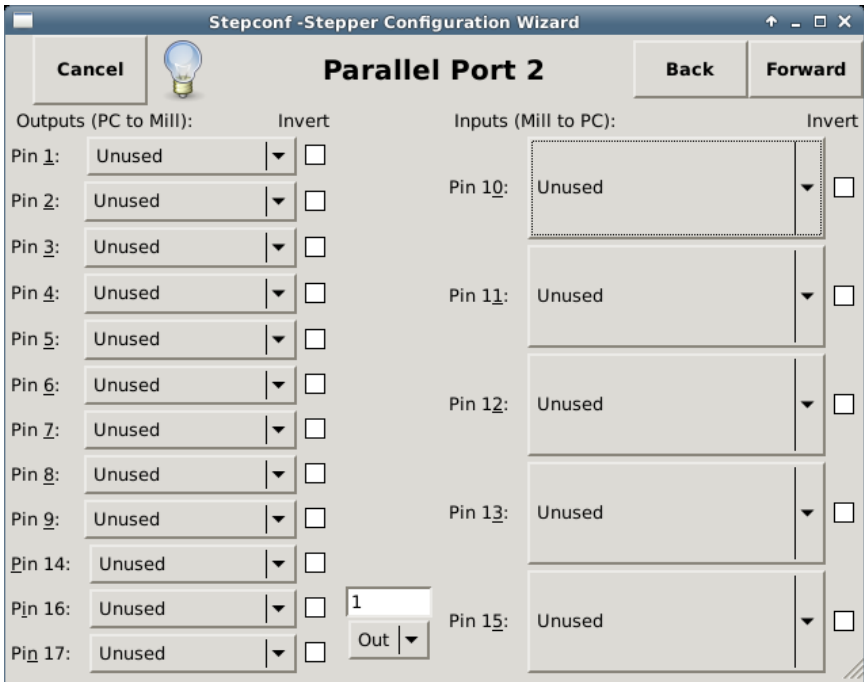


図 3-10

2 番目のパラレルポート（選択されている場合）を構成して、このページでピンを割り当てることができます。

ステップ信号と方向信号は選択できません。

使用可能な入力/出力ピンの数を最大化するために、入力または出力を選択できます。

アドレスは、16 進数（多くの場合 0x378）または Linux のデフォルトのポート番号（おそらく 1）として指定できます。

3.7.7 軸構成

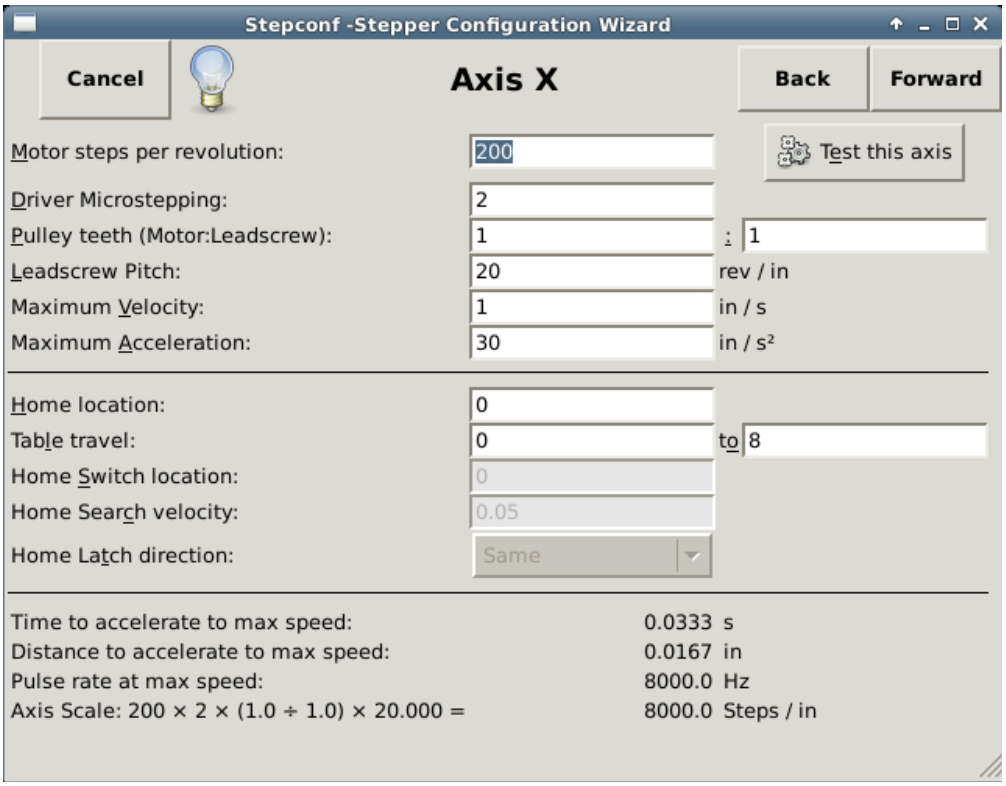


図 3-11

- 1 回転あたりのモーターステップ数-モーター 1 回転あたりのフルステップ数。 モーターが 1 ステップあたりの度数（1.8 度など）であることがわかっている場合は、360 を 1 ステップあたりの度数で割って、モーターの 1 回転あたりのステップ数を求めます。
- ドライバーのマイクロステッピング-ドライバーによって実行されるマイクロステッピングの量。 ハーフステップの場合は 2 を入力します。
- プーリー比-マシンのモーターと親ねじの間にプーリーがある場合は、ここに比率を入力します。 そうでない場合は、1：1 と入力します。
- 親ねじピッチ-ここに親ねじのピッチを入力します。 インチ単位を選択した場合は、1 インチあたりのねじ山の数を入力します。mm 単位を選択した場合は、1 回転あたりのミリメートル数を入力します（たとえば、2mm / rev の場合は 2 を入力します）。 機械が間違った方向に移動する場合は、ここに正の数ではなく負の数を入力するか、軸の方向ピンを反転させます。
- 最大速度-軸の最大速度を 1 秒あたりの単位で入力します。

- 最大加速度-これらの項目の正しい値は、実験によってのみ決定できます。 速度を設定するには「最大速度の検索」を、加速度を設定するには「最大加速度の検索」を参照してください。
- ホームロケーション-この軸のホーミング手順を完了した後、マシンが移動する位置。 ホームスイッチのないマシンの場合、これは、ホームボタンを押す前にオペレーターが手動でマシンを移動する場所です。 ホームスイッチとリミットスイッチを組み合わせる場合は、スイッチからホームポジションに移動する必要があります。そうしないと、ジョイントリミットエラーが発生します。
- テーブル移動-機械の原点に基づくその軸の移動範囲。 ホームロケーションは、テーブルトラベル内にあり、テーブルトラベル値の 1 つと等しくない必要があります。
- ホームスイッチの場所-マシンの原点を基準にして、ホームスイッチがトリップまたは解放される場所。 この項目と以下の 2 つは、パラレルポートのピン配置でホームスイッチが選択された場合にのみ表示されます。ホームスイッチとリミットスイッチを組み合わせる場合、ホームスイッチの位置をホーム位置と同じにすることはできません。そうしないと、ジョイントリミットエラーが発生します。
- ホーム検索速度-ホームスイッチを検索するときに使用する速度。 スイッチが移動の終わりに近い場合、この速度は、移動の終わりに達する前に軸が減速して停止できるように選択する必要があります。 スイッチが（トリップポイントから移動の一端まで閉じるのではなく）短い移動範囲でのみ閉じられる場合は、スイッチが再び開く前に軸が減速して停止し、ホーミングできるように、この速度を選択する必要があります。 常にスイッチの同じ側から開始する必要があります。 原点復帰手順の開始時にマシンが間違った方向に移動した場合は、ホームサーチ速度の値を無効にします。
- ホームラッチの方向-[同じ]を選択して軸をスイッチから外し、非常に低速で再度アプローチします。 2 回目にスイッチを閉じると、ホームポジションが設定されます。 [反対]を選択すると、軸がスイッチから外れ、スイッチが開くとホームポジションが設定されます。
- 最大速度まで加速する時間-最大加速と最大速度から計算された最大速度に到達する時間。
- 最高速度まで加速する距離-停止状態から最高速度に到達する距離。
- 最大速度での脈拍数-上記で入力した値に基づいて計算された情報。 最大速度での最大脈拍数が BASE_PERIOD を決定します。 20000Hz を超える値は、応答時間が遅くなったり、ロックアップしたりする可能性があります（使用可能な最速のパルスレートはコンピューターによって異なります）
- AxisSCALE-ini ファイルの[SCALE]設定で使用される番号。 これは、ユーザー単位あたりのステップ数です。
- この軸をテストする-これにより、各軸のテストを可能にするウィンドウが開きます。 これは、この軸のすべての情報を入力した後に使用できます。

1.1.1.1 この軸をテストする

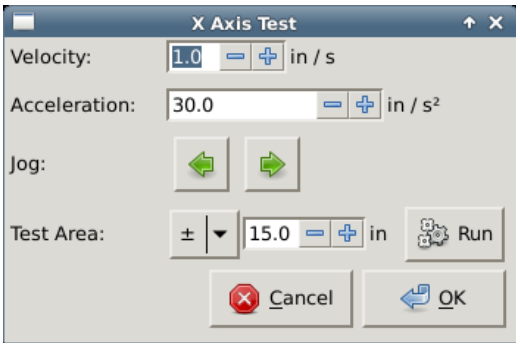


図 3-12

この軸のテストは、加速度と速度の異なる値を試すためにステップ信号と方向信号のみを出力する基本的なテスターです。

重要

この軸のテストを使用するには、必要に応じて軸を手動で有効にする必要があります。 ドライバーがチャージポンプを持っている場合は、それをバイパスする必要があります。 この軸が反応してスイッチ入力を制限しないことをテストします。 注意して使用してください。

最大速度を見つける

低い加速度（たとえば、2 インチ/ s2 または 50mm / s2）と、達成したい速度から始めます。 付属のボタンを使用して、移動の中心近くまで軸をジョグします。 加速度の値が低いと、軸が減速して停止するまでに驚くほどの距離がかかる可能性があるため、注意してください。

利用可能な移動量を測定した後、テストエリアに安全な距離を入力します。ストール後、モーターが次に予期しない方向に動き始める可能性があることに注意してください。 次に、[実行]をクリックします。 機械はこの軸に沿って前後に動き始めます。 このテストでは、加速とテストエリアの組み合わせにより、マシンが選択した速度に到達し、少なくとも短い距離を巡航できるようにすることが重要です。距離が長いほど、このテストは優れています。 式 d = 0.5 * v * v / a は、指定された加速度で指定された速度に到達するために必要な最小距離を示します。 便利で安全な場合は、テーブルを運動方向に押し付けて、切削抵抗をシミュレートします。 マシンが停止した場合は、速度を下げてテストを再開してください。

マシンが明らかにストールしなかった場合は、[実行]ボタンをクリックしてオフにします。 これで、軸は開始位置に戻ります。 位置が正しくない場合、テスト中に軸が停止するか、ステップが失われます。 速度を下げて、テストを再開します。

Velocity をどれだけ低くしても、マシンが動かなかったり、停止したり、ステップを失ったりしない場合は、次のことを確認してください。

- ステップ波形のタイミングを修正する

- ステップピンの反転を含む正しいピン配置
- 正しく、十分にシールドされたケーブル
- モーター、モーターカップリング、親ねじなどの物理的な問題。

このテスト手順中に軸が停止したりステップを失ったりしない速度を見つけたら、それを 10%減らし、それを軸の最大速度として使用します。

最大加速度の検出前の手順で検出した最大速度を使用して、テストする加速度の値を入力します。 上記と同じ手順で、必要に応じて加速度の値を上下に調整します。 このテストでは、加速とテスト領域の組み合わせにより、マシンが選択した速度に到達できるようにすることが重要です。 このテスト手順中に軸がストールしたりステップを失ったりしない値を見つけたら、それを 10%減らし、それを軸の最大加速度として使用します。

3.7.8 スピンドル構成

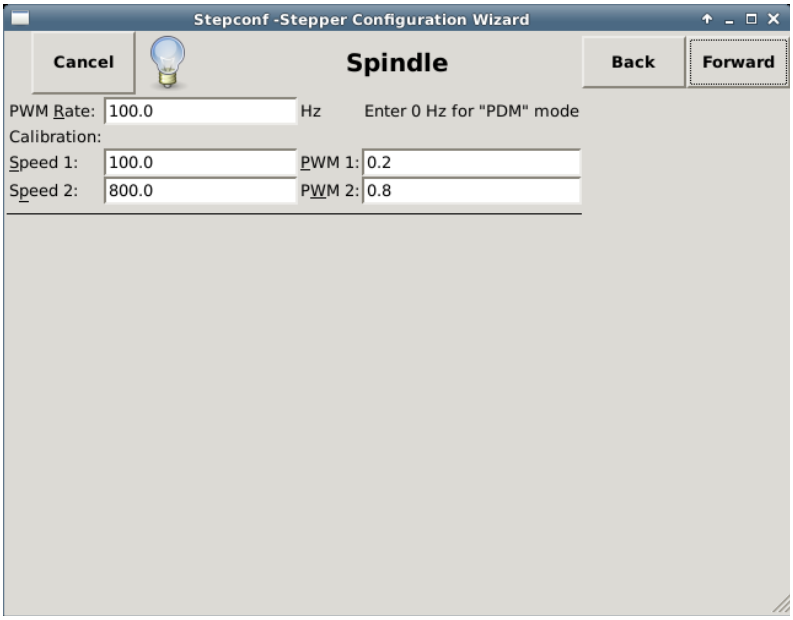


図 3-13

このページは、出力の 1 つに対して[パラレルポートのピン配置]ページでスピンドル PWM が選択されている場合にのみ表示されます。

1.1.1.1 スピンドル速度制御

ピン配列にスピンドル PWM が表示されている場合は、次の情報を入力する必要があります。

- PWM レート-スピンドルへの PWM 信号の搬送周波数。 PDM モードの場合は 0 を入力します。これは、アナログ制御電圧を生成するのに役立ちます。 適切な値については、スピンドルコントローラのドキュメントを参照してください。

- 速度 1 および 2、PWM 1 および 2-生成された構成ファイルは、単純な線形関係を使用して、特定の RPM 値の PWM 値を決定します。 値がわからない場合は、決定できます。 詳細については、スピンドルキャリブレーションの決定を参照してください。

3.7.8.1 スピンドル同期モーション

スピンドルエンコーダからの適切な信号が HAL を介して LinuxCNC に接続されている場合、LinuxCNC は旋盤ねじ切りをサポートします。 これらの信号は次のとおりです。

- スピンドルインデックス-スピンドルの 1 回転ごとに 1 回発生するパルスです。
- スピンドルフェーズ A-これは、スピンドルが回転するときに複数の等間隔の場所で発生するパルスです。
- スピンドルフェーズ B（オプション）-これは発生する 2 番目のパルスですが、スピンドルフェーズ A からのオフセットがあります。A と B の両方を使用する利点は、方向検出、ノイズ耐性の向上、および分解能の向上です。

スピンドルフェーズ A とスピンドルインデックスがピン配置に表示されている場合は、次の情報を入力する必要があります。

- Spindle-At-Speed を使用する-エンコーダフィードバックを使用すると、フィードが移動する前に linuxcnc がスピンドルがコマンド速度に達するのを待機するように選択できます。 このオプションを選択し、十分に近いスケールを設定します。
- 速度表示フィルターゲイン-視覚的なスピンドル速度表示の安定性を調整するための設定。
- 1 回転あたりのサイクル数-スピンドルの 1 回転中のスピンドル A 信号のサイクル数。 このオプションは、入力がスピンドルフェーズ A に設定されている場合にのみ有効になります
- スレッドの最大速度-スレッドで使用される最大スピンドル速度。 高スピンドル RPM または高解像度のスピンドルエンコーダの場合、BASE_PERIOD の値を低くする必要があります。

3.7.8.2 スピンドルキャリブレーションの決定

[スピンドル構成]ページに次の値を入力します。

Speed1:	0	`WM1:	0
Speed2:	1000	PWM2:	1

構成プロセスの残りのステップを完了してから、構成を使用して LinuxCNC を起動します。 マシンの電源を入れ、[MDI]タブを選択します。 M3 S100 と入力して、スピンドルの回転を開始します。 別の S 番号 S800 を入力して、スピンドル速度を変更します。 有効な数値（この時点で）の範囲は 1～1000 です。

2 つの異なる S 番号について、実際のスピンドル速度を RPM で測定します。 S 番号と実際のスピンドル速度を記録します。 Stepconf を再度実行します。 [速度]に測定速度を入力し、PWM に S 番号を 1000 で割った値を入力します。

ほとんどのスピンドルドライバーは、応答曲線がやや非線形であるため、次のことを行うのが最適です。

- 2 つのキャリブレーション速度が RPM で近すぎないことを確認してください
- 2 つのキャリブレーション速度が、フライス盤で通常使用する速度の範囲内にあることを確認してください

たとえば、スピンドルが 0RPM から 8000RPM になるが、通常は 400 RPM（10%）から 4000 RPM（100%）の速度を使用する場合、1600 RPM（40%）と 2800 を与える PWM 値を見つけます。 RPM（70%）。

3.7.9 オプション

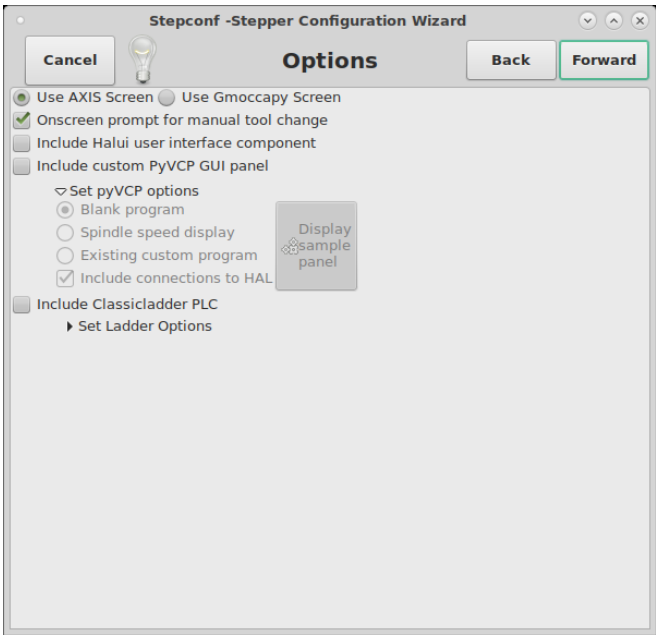


図 3-14

- Halui を含める-これにより、Halui ユーザーインターフェイスコンポーネントが追加されます。 詳細については、HALUI の章を参照してください。
- pyVCP を含める-このオプションは、作業する pyVCP パネルベースファイルまたはサンプルファイルを追加します。 詳細については、PyVCP の章を参照してください。
- ClassicLadder PLC を含める-このオプションは、ClassicLadder PLC（プログラマブルロジックコントローラー）を追加します。 詳細については、Classicladder の章を参照してください。
- ツール変更の画面上のプロンプト-このボックスがチェックされている場合、LinuxCNC は一時停止し、M6 が検出されたときにツールを変更するように促します。 この機能は通常、プリセット可能なツールがある場合にのみ役立ちます。

3.7.10 マシン構成が完了しました

[適用]をクリックして、構成ファイルを書き込みます。 後で、このプログラムを再実行して、前に入力した設定を微調整できます。

3.7.11 アクシストラベルアンドホーム

各軸について、移動範囲は限られています。 旅行の物理的な終わりは、ハードストップと呼ばれます。

ハードストップの前にリミットスイッチがあります。 通常の動作中にリミットスイッチが発生すると、LinuxCNC はモーターアンプをシャットダウンします。 ハードストップとリミットスイッチの間の距離は、動力のないモーターが惰性で停止するのに十分な長さでなければなりません。

リミットスイッチの前にソフトリミットがあります。 これは、ホーミング後にソフトウェアで適用される制限です。 MDI コマンドまたは g コードプログラムがソフト制限を通過する場合、それは実行されません。 ジョグがソフト制限を超えると、ソフト制限で終了します。

ホームスイッチは、移動中のどこにでも配置できます（ハードストップ間）。 リミットスイッチに達したときに外部ハードウェアがモーターアンプを非アクティブにしない限り、リミットスイッチの 1 つをホームスイッチとして使用できます。

ゼロ位置は、機械座標系で 0 である軸上の位置です。 通常、ゼロ位置はソフト制限内にあります。 旋盤では、一定の表面速度モードでは、工具オフセットが有効になっていないときに、マシン X = 0 がスピンドル回転の中心に対応する必要があります。

ホームポジションは、ホーミングシーケンスの最後に軸が移動する移動内の位置です。 この値はソフト制限内である必要があります。 特に、ホームポジションがソフトリミットと正確に等しくなることはありません。

1.1.1.1 リミットスイッチなしで動作

リミットスイッチなしで機械を操作できます。 この場合、マシンがハードストップに到達するのを阻止するのはソフトリミットのみです。 ソフトリミットは、マシンがホームに戻された後にのみ機能します。

3.7.11.1 ホームスイッチなしで操作

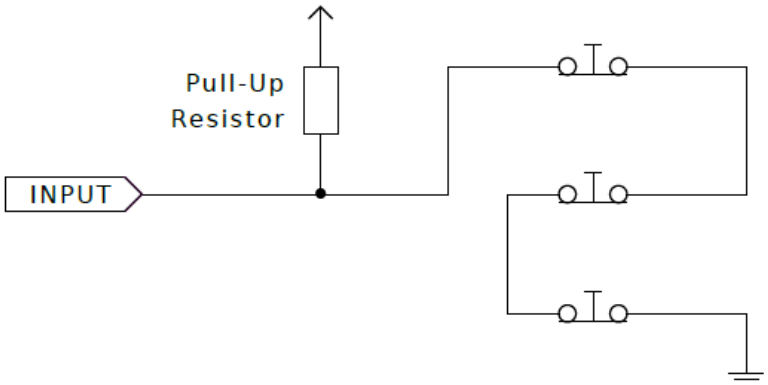
ホームスイッチなしで機械を操作できます。 マシンにリミットスイッチがあり、ホームスイッチがない場合は、ホームスイッチとしてリミットスイッチを使用するのが最適です（たとえば、ピン配置で[最小制限] + [ホーム X] を選択します）。 マシンにスイッチがまったくない場合、または別の理由でリミットスイッチをホームスイッチとして使用できない場合は、目またはマッチマークを使用してマシンをホームにする必要があります。 目によるホーミングは、スイッチへのホーミングほど再現性はありませんが、それでもソフトリミットを使用できます。

3.7.11.2 ホームおよびリミットスイッチの配線オプション

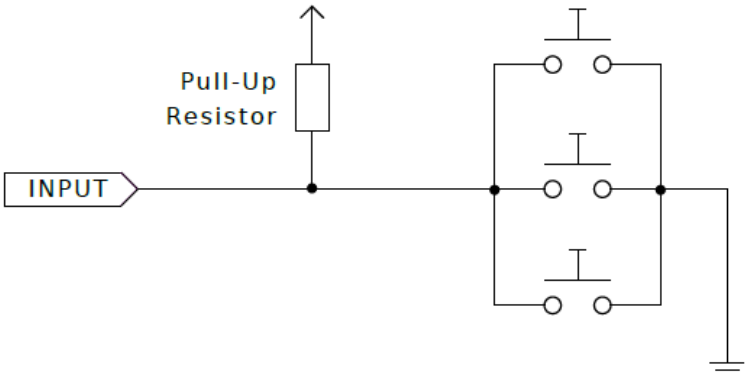
外部スイッチの理想的な配線は、スイッチごとに 1 つの入力です。 ただし、PC のパラレルポートは合計 5 つの入力しか提供しませんが、3 軸マシンには最大 9 つのスイッチがあります。 代わりに、複数のスイッチがさまざまな方法で相互に配線されているため、必要な入力の数は少なくなります。

次の図は、複数のスイッチを 1 つの入力ピンに配線する一般的な考え方を示しています。 いずれの場合も、1 つのスイッチが作動すると、INPUT に表示される値は論理 HIGH から LOW になります。 ただし、LinuxCNC は、スイッチが閉じているときに TRUE 値を期待するため、ピン配置構成ページで対応する[反転]ボックスをオンにする必要があります。 図に示されているプルアップ抵抗は、グランドへの接続が行われるまで入力をハイにプルし、その後、入力がローになります。 そうしないと、回路が開いているときに入力がオンとオフの間でフロートする可能性があります。 通常、パラレルポートの場合は 47k を使用します。

ノーマルクローズスイッチ N / C スイッチを直列に配線（簡略図）



ノーマルオープンスイッチ N / O スイッチを並列に配線（簡略図）



Stepconf では、次のスイッチの組み合わせが許可されています。

- すべての軸のホームスイッチを組み合わせる
- すべての軸のホームスイッチを組み合わせる
- 1 つの軸に両方のリミットスイッチを組み合わせる
- 1 つの軸に両方のリミットスイッチとホームスイッチを組み合わせる
- 1 つのリミットスイッチと 1 つの軸のホームスイッチを組み合わせる

3.8 Mesa 構成ウィザード

PNCconf は、特定の Mesa Anything I / O 製品を利用する構成の構築を支援するために作成されています。

閉ループサーボシステムまたはハードウェアステッパーシステムを構成できます。 Stepconf と同様のウィザードアプローチを使用します（ソフトウェアステッピング、パラレルポート駆動システムに使用されます）。

PNCconf はまだ開発段階（ベータ版）であるため、いくつかのバグがあり、機能が不足しています。 バグや提案を LinuxCNC フォーラムページまたはメールリストに報告してください。

PNCconf を使用する場合、2 つの考えがあります。

1 つは、PNCconf を使用して常にシステムを構成することです。オプションを変更する場合は、PNCconf をリロードして、新しいオプションを構成できるようにします。 これは、マシンがかなり標準的であり、カスタムファイルを使用して非標準機能を追加できる場合にうまく機能します。 PNCconf は、この点であなたと協力しようとします。

もう 1 つは、PNCconf を使用して目的に近い構成を構築し、すべてを手作業で編集してニーズに合わせて調整することです。 これは、PNCconf の範囲を超えて大幅な変更が必要な場合、または LinuxCNC をいじくり回したり、学習したりする場合に選択されます。

進むボタン、戻るボタン、キャンセルボタンを使用してウィザードページをナビゲートします。また、ページ、図、および出力ページに関するヘルプ情報を提供するヘルプボタンもあります。

ヒント

PNCconf のヘルプページには、最新の情報と追加の詳細が記載されている必要があります。

3.8.1 ステップバイステップの説明

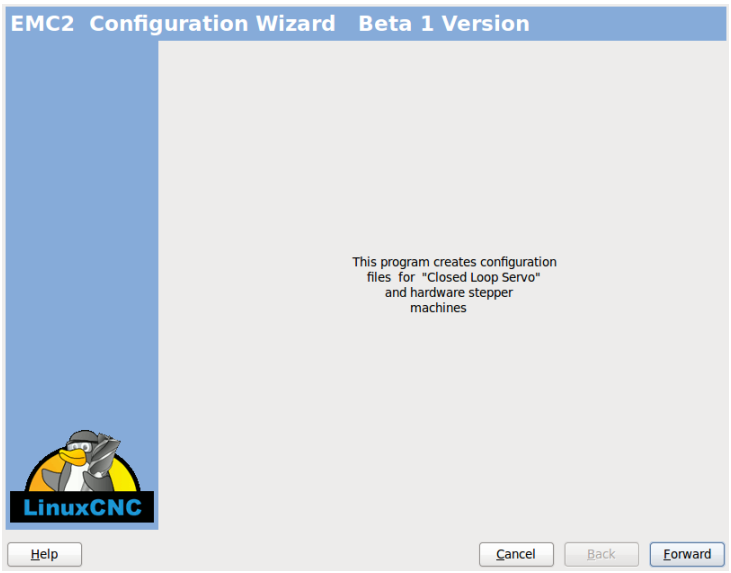


図 3-15

3.8.2 作成または編集

これにより、以前に保存した構成を選択したり、新しい構成を作成したりできます。 [構成の変更]を選択してから[次へ]を押すと、ファイル選択ボックスが表示されます。 Pncconf は、最後に保存したファイルを事前に選択します。 編集する構成を選択します。 メインの hal ファイルまたは ini ファイルに変更を加えた場合、Pncconf はそれらのファイルを上書きし、それらの変更は失われます。 一部のファイルは上書きされず、Pncconf はそれらのファイルにメモを置きます。 また、デスクトップショートカット/ランチャーオプションを選択することもできます。 デスクトップショートカットは、新しい構成ファイルを指すフォルダーアイコンをデスクトップに配置します。 それ以外の場合は、linuxcnc / configs の下のホームフォルダを調べる必要があります。

デスクトップランチャーは、構成を直接開始するためのアイコンをデスクトップに追加します。 CNC メニューにある ConfigurationSelector LinuxCNC を使用し、構成名を選択して、メインメニューから起動することもできます。

3.8.3 基本的な機械情報

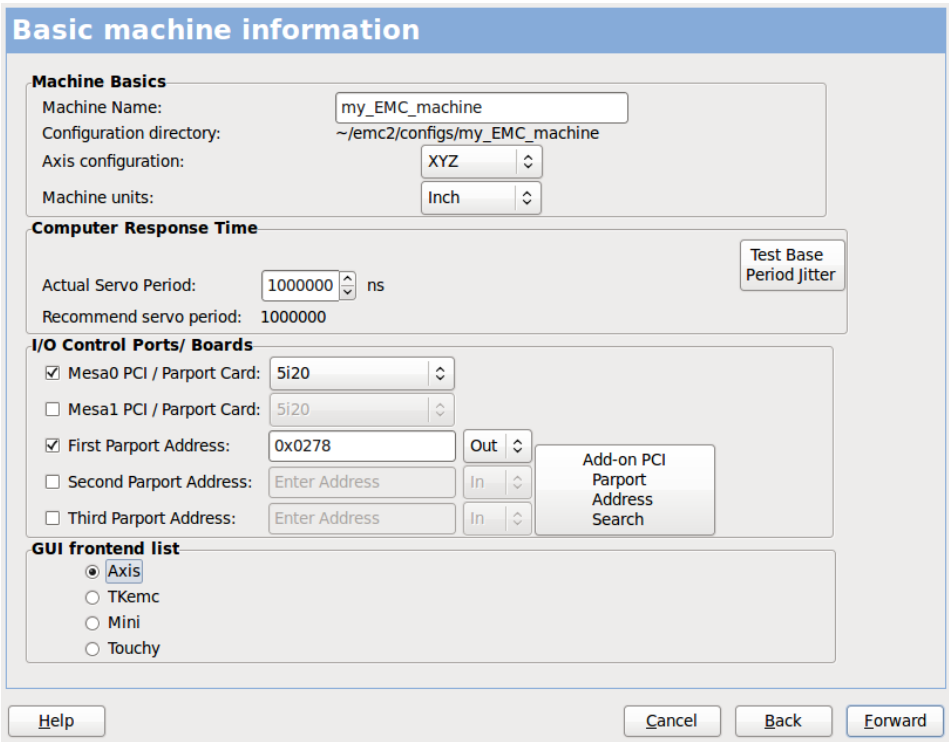


図 3-16

機械の基本

スペースを含む名前を使用する場合、PNCconf はスペースをアンダースコアに置き換えます（大まかなルールとして、Linux は名前のスペースを好みません）軸構成を選択します-これにより、構築するマシンのタイプと使用可能な軸が選択されます。 機械単位セレクターを使用すると、次のページでメートル法またはインペリアル単位のデータを入力できます。

ヒント

メトリックを使用する場合、デフォルトは変換されないため、それらが正しい値であることを確認してください。

コンピューターの応答時間

サーボ周期は、システムの心拍数を設定します。 待ち時間とは、コンピューターがその期間より長くなる可能性がある時間を指します。 鉄道と同じように、LinuxCNC は非常にタイトで一貫したタイムラインですべてを必要とします。そうしないと、悪いことが起こります。 LinuxCNC は、リアルタイムオペレーティングシステムを必要とし、使用します。つまり、LinuxCNC が計算を必要とする場合、応答時間は低く、LinuxCNC の計算を行う場合、優先度の低い要求（画面ボタンへのユーザー入力や 描画など）。

レイテンシーのテストは非常に重要であり、早期にチェックすることが重要です。幸いなことに、Mesa カードを使用して最速の応答時間（エンコーダーカウントと PWM 生成）を必要とする作業を行うことで、これらのことにパラレルポートを使用した場合よりもはるかに長いレイテンシーに耐えることができます。 LinuxCNC の標準テストでは、BASE 期間のレイテンシをチェックしています（基本期間を使用していなくても）。テストベース期間のジッターボタンを押すと、レイテンシーテストウィンドウが起動します（これをアプリケーション/ cnc パネルから直接ロードすることもできます）。テストでは、数分間実行することに言及していますが、長いほど良いです。 15 分は最低限、一晩はさらに良いと考えてください。この時点で、コンピューターを使用して物をロードし、ネットを使用し、USB を使用するなど、最悪の場合のレイテンシーを知り、特定のアクティビティがレイテンシーを損なうかどうかを調べます。基本周期のジッターを調べる必要があります。 20000 未満のものはすべて優れています。マシンで高速のソフトウェアステッピングを実行することもできます。20000～50000 は、ソフトウェアステッピングには適していますが、私たちにとっては問題ありません。 50000～100000 は実際にはそれほど優れていませんが、高速応答を行うハードウェアカードで使用できます。したがって、100000 未満のものはすべて使用できます。待ち時間が期待外れであるか、定期的にひどいしゃっくりが発生する場合でも、それを改善できる可能性があります。

ヒント

LinuxCNC wiki で取得された機器とレイテンシーのユーザーコンパイル済みリストがあります：
http：//wiki.linuxcnc.org/cgi-bin/-wiki.pl？Latency-Test リストに情報を追加することを検討してください。
また、そのページには、いくつかの遅延の問題を修正するための情報へのリンクがあります。

これで、レイテンシーに満足し、サーボ期間を選択する必要があります。 速度（電圧）ではなくトルク（電流）をより速い速度で制御する閉ループサーボシステムを構築している場合、ほとんどの場合、1000000 ns のサーボ周期で問題ありません（1 kHz のサーボ計算速度-1 秒あたり 1000 回の計算が得られます）。 200000（5 kHz の計算レート）のようなものの方が良いでしょう。 サーボレートを下げることの問題は、コンピュータが LinuxCNC の計算以外のことを行うために利用できる時間が少なくなることです。 通常、ディスプレイ（GUI）

の応答性が低下します。 バランスを決める必要があります。 閉ループサーボシステムを調整してからサーボ周期を変更する場合は、おそらくそれらを再度調整する必要があることに注意してください。

I / O 制御ポート/ボード

PNCconf は、最大 2 つの Mesa ボードと 3 つのパラレルポートを備えたマシンを構成できます。 パラレルポートは、単純な低速（サーボレート）I / O にのみ使用できます。

Mesa

PNCconf は、エンコーダをカウントしたり、ステップ信号や PWM 信号を出力したりするようにパラレルポートを構成しないため、少なくとも 1 つの Mesa ボードを選択する必要があります。 選択ボックスで使用可能なメサカードは、PNCconf がシステム上のファームウェアに対して検出したものに基づいています。 設定ファイルを使用して、カスタムファームウェアを追加したり、一部のファームウェアまたはボードをブラックリストに登録（無視）したりするオプションがあります。 ファームウェアが見つからない場合、PNCconf は警告を表示し、内部サンプルファームウェアを使用します-テストはできません。 注意すべき点の 1 つは、2 枚の PCI Mesa カードを選択した場合、現在、どちらのカードが 0 でどちらが 1 であるかを予測する方法がないことです。テストする必要があります。カードを移動すると、順序が変わる可能性があります。 2 枚のカードで設定する場合、テストを機能させるには両方のカードを取り付ける必要があります。

パラレルポート

最大 3 つのパラレルポート（パーポートと呼ばれる）を単純な I / O として使用できます。パーポートのアドレスを設定する必要があります。 Linux のパラレルポート番号システム（0、1、または 2）を入力するか、実際のアドレスを入力することができます。オンボードパーポートのアドレスは、多くの場合 0x0278 または 0x0378（16 進数で記述）ですが、BIOS ページで確認できます。 BIOS ページは、コンピューターを最初に起動したときに見つかります。キーを押して入力する必要があります（F2 など）。 BIOS ページで、パラレルポートアドレスを見つけて、一部のコンピューターで SPP、EPP などのモードを設定できます。この情報は、起動中に数秒間表示されます。 PCI パラレルポートカードの場合、アドレスはパーポートアドレス検索ボタンを押すことで見つけることができます。これにより、検索可能なすべての PCI デバイスのリストを含むヘルプ出力ページがポップアップ表示されます。アドレスのリストを含むパラレルポートデバイスへの参照があるはずです。それらのアドレスの 1 つが機能するはずです。すべての PCI パラレルポートが正しく機能するわけではありません。 in（入力ピンの最大数）または out（出力ピンの最大数）のいずれかのタイプを選択できます。

GUI フロントエンドリスト

これは、LinuxCNC が使用するグラフィカル表示画面を指定します。 それぞれに異なるオプションがあります。

AXIS

- 旋盤を完全にサポートします。
- 最も開発され、使用されているフロントエンドです
- 最も開発され、使用されているフロントエンドです
- tkinter ベースなので、PYVCP（Python ベースの仮想コントロールパネル）を自然に統合します。
- 3D グラフィカルウィンドウがあります。
- 側面または中央のタブに VCP を統合できます

TOUCHY

- Touchy は、タッチスクリーン、いくつかの最小限の物理スイッチ、および MPG ホイールで使用するように設計されています。
- サイクルスタート、アボート、およびシングルステップの信号とボタンが必要です
- また、共有軸 MPG ジョギングを選択する必要があります。
- GTK ベースなので、GLADE VCP（仮想コントロールパネル）を自然に統合します。
- GTK ベースなので、GLADE VCP（仮想コントロールパネル）を自然に統合します。
- グラフィカルウィンドウはありません
- カスタムテーマで外観を変更できます

MINI

- OEMSherline マシンの標準
- Estop を使用しません
- VCP 統合なし

TkLinuxCNC

- コントラストの明るいブルースクリーン
- 別のグラフィックウィンドウ
- VCP 統合なし

3.8.4 外部構成

このページでは、ジョギングやオーバーライドなどの外部コントロールを選択できます。

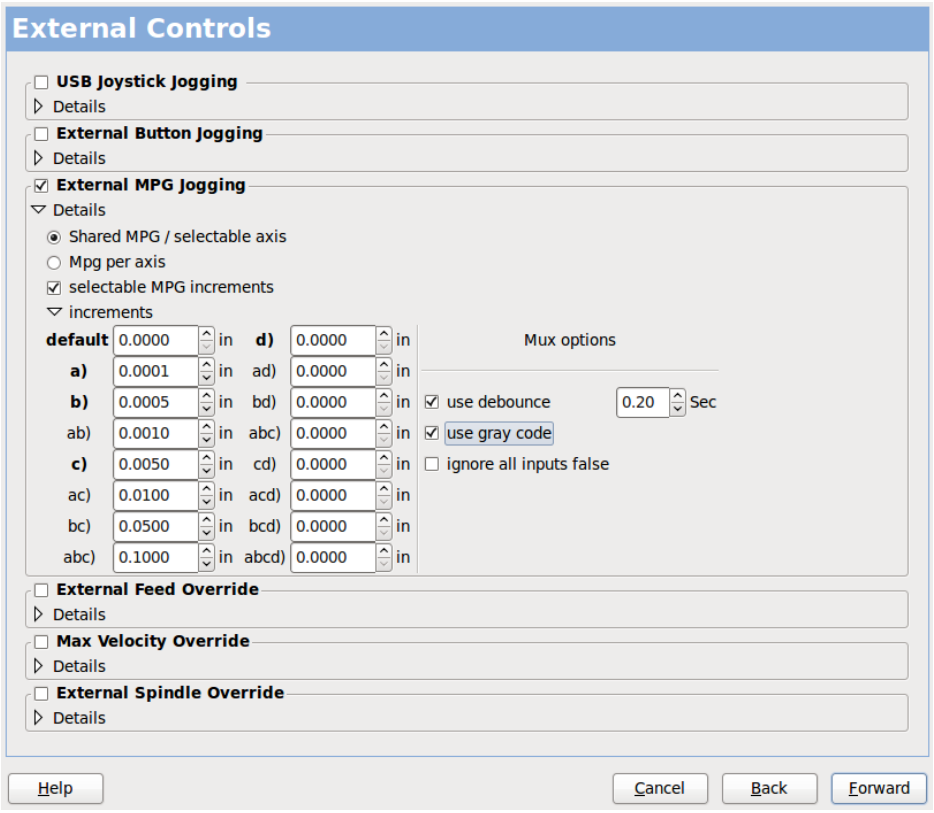


図 3-17

ジョギング用にジョイスティックを選択した場合、LinuxCNC をロードするには常にジョイスティックを接続する必要があります。 アナログスティックを便利なジョギングに使用するには、カスタム HAL コードを追加する必要があります。 MPG ジョギングには、MESA エンコーダカウンターに接続されたパルスジェネレーターが必要です。 オーバーライドコントロールは、パルスジェネレーター（MPG）またはスイッチ（回転式ダイヤルなど）のいずれかを使用できます。 外部ボタンは、スイッチベースの OEM ジョイスティックで使用できます。

ジョイスティックジョギング

カスタムデバイスルールをシステムにインストールする必要があります。 これは、LinuxCNC が LINUX のデバイスリストに接続するために使用するファイルです。 PNCconf は、このファイルの作成に役立ちます。

デバイスルールの検索は、システムでルールを検索します。これを使用して、PNCconf で既に構築したデバイスの名前を見つけることができます。

デバイスルールを追加すると、プロンプトに従って新しいデバイスを構成できます。 デバイスが利用可能である必要があります。 テストデバイスを使用すると、デバイスをロードし、そのピン名を確認し、halmeter でその機能を確認できます。

ジョイスティックのジョギングは、HALUI および hal_input コンポーネントを使用します。

外部ボタン

指定されたジョギング速度で簡単なボタンを使用して軸をジョギングできます。 おそらく急速なジョギングに最適です。

MPG ジョギング

手動パルスジェネレータを使用して、マシンの軸をジョグできます。

MPG は、商用グレードのマシンでよく見られます。 これらは、MESA エンコーダカウンタでカウントできる直交パルスを出力します。 PNCconf では、軸ごとに MPG を使用することも、すべての軸で 1 つの MPG を共有することもできます。 スイッチまたはシングルスピードを使用してジョグ速度を選択できます。

選択可能な増分オプションは、mux16 コンポーネントを使用します。 このコンポーネントには、生のスイッチ入力のフィルタリングに役立つデバウンスやグレイコードなどのオプションがあります。

オーバーライド

PNCconf では、パルスジェネレーター（MPG）またはスイッチ（ロータリーなど）を使用して、送り速度やスピンドル速度をオーバーライドできます。

3.8.5 GUI 構成

ここでは、表示画面のデフォルトを設定したり、仮想コントロールパネル（VCP）を追加したり、いくつかの LinuxCNC オプションを設定したりできます。

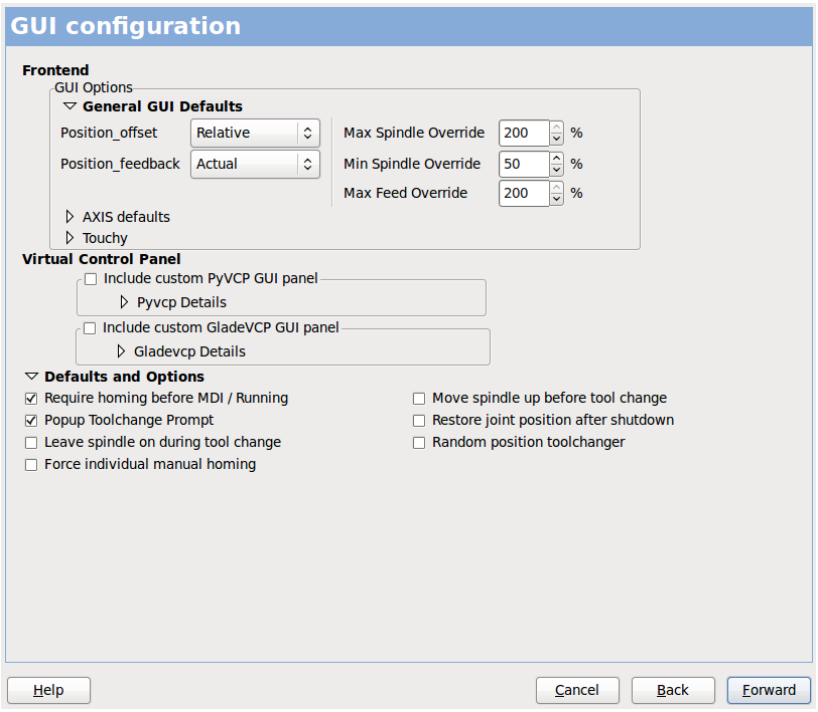


図 3-18

フロントエンド GUI オプション

デフォルトオプションを使用すると、任意の表示画面に対して一般的なデフォルトを選択できます。

AXIS のデフォルトは、AXIS に固有のオプションです。 サイズ、位置、または強制最大化オプションを選択した場合、PNCconf は設定ファイル（.axisrc）を上書きしてもよいかどうかを尋ねます。 このファイルに手動でコマンドを追加していない限り、許可しても問題ありません。 システムに対応している場合は、位置と最大力を使用して、AXIS を 2 番目のモニターに移動できます。

Touchy のデフォルトは、Touchy に固有のオプションです。 Touchy のオプションのほとんどは、設定ページを使用して Touchy の実行中に変更できます。 Touchy は GTK を使用して画面を描画し、GTK はテーマをサポートしています。 テーマは、プログラムの基本的なルックアンドフィールを制御します。 テーマはネットからダウンロードするか、自分で編集することができます。 あなたが選ぶことができるコンピュータ上の現在のテーマのリストがあります。 一部のテキストを目立たせるために、PNCconf ではテーマのデフォルトを上書きできます。 システムに対応している場合は、位置と最大力のオプションを使用して、Touchy を 2 番目のモニターに移動できます。

VCP オプション

仮想コントロールパネルを使用すると、カスタムコントロールと表示を画面に追加できます。 AXIS と Touchy は、これらのコントロールを画面内の指定された位置に統合できます。 VCP には、Tkinter を使用して画面を描画する pyVCP と、GTK を使用して画面を描画する GLADEVCP の 2 種類があります。

PyVCP

PyVCP 画面の XML ファイルは手動でのみ作成できます。 PyVCP は両方とも TKinter を使用するため、AXIS に自然に適合します。

HAL ピンは、ユーザーがカスタム HAL ファイル内に接続するために作成されます。 ユーザーがそのまま使用するか、上に構築するためのサンプルスピンドルディスプレイパネルがあります。 後でコントロールウィジェットを追加できる空のファイルを選択するか、スピンドル速度を表示し、スピンドルが要求された速度であるかどうかを示すスピンドル表示サンプルを選択できます。

PNCconf は、適切なスピンドルディスプレイ HAL ピンを接続します。 AXIS を使用している場合、パネルは右側に統合されます。 AXIS を使用していない場合、パネルはフロントエンド画面とは別のスタンドアロンになります。

ジオメトリオプションを使用して、パネルのサイズと移動を行うことができます。たとえば、システムに対応している場合は、パネルを 2 番目の画面に移動できます。 [サンプルパネルの表示]ボタンを押すと、サイズと配置のオプションが適用されます。

GLADE VCP

GLADE VCP は、どちらも GTK を使用して描画するため、TOUCHY 画面内に自然に収まりますが、GLADE VCP のテーマを変更することで、AXIS で非常にうまくブレンドすることができます。（レドモンドを試してください）

グラフィカルエディタを使用して XML ファイルを作成します。 HAL ピンは、ユーザーがカスタム HAL ファイル内に接続するために作成されます。

GLADE VCP は、PNCconf が現在活用していない、はるかに洗練された（そして複雑な）プログラミングインタラクションも可能にします。（マニュアルの GLADE VCP を参照してください）

PNCconf には、ユーザーがそのまま使用するか、上に構築するためのサンプルパネルがあります。 GLADE VCP PNCconf を使用すると、サンプルディスプレイでさまざまなオプションを選択できます。

サンプルオプションの下で、必要なものを選択します。 ゼロボタンは、後で HALUI セクションで編集できる HALUI コマンドを使用します。

自動 Z タッチオフには、従来のラダータッチオフプログラムとプローブ入力を選択する必要もあります。 導電性タッチオフプレートと接地された導電性ツールが必要です。 それがどのように機能するかについてのアイデアについては、以下を参照してください。

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

[表示オプション]で、サイズ、位置、および最大力をスタンドアロンパネルで使用して、システムに対応している場合は 2 台目のモニターに画面を配置することができます。

パネルの基本的なルックアンドフィールを設定する GTK テーマを選択できます。 通常、これをフロントエンド画面と一致させる必要があります。 これらのオプションは、[サンプルの表示]ボタンを押すと使用されます。 フロントエンド画面に応じて GLADEVCP を使用すると、パネルを表示する場所を選択できます。

強制的にスタンドアロンにするか、AXIS を使用して中央または右側に配置し、Touchy を使用して中央に配置することができます。

デフォルトとオプション

- MDI /実行前にホーミングが必要
  - 原点復帰前にミシンを移動できるようにする場合は、このチェックボックスをオフにします。
- ポップアップツールプロンプト
  - ツール変更の画面プロンプトを選択するか、ユーザー提供のカスタムツールチェンジャー Hal ファイルの標準信号名をエクスポートします
- 工具交換中はスピンドルをオンのままにします。
  - 旋盤に使用
- 個別の手動ホーミングを強制する

- 工具交換前にスピンドルを上に動かしてください
- シャットダウン後に関節の位置を復元する

➢ 重要なキネマティクスマシンに使用されます

- ランダムポジションツールチェンジャー

➢ ツールを同じポケットに戻さないツールチェンジャーに使用されます。 ツールチェンジャーをサポートするには、カスタム HAL コードを追加する必要があります。

3.8.6 Mesa 構成

Mesa 構成ページでは、さまざまなファームウェアを利用できます。 ここで Mesa カードを選択した基本ページで、使用可能なファームウェアを選択し、使用可能なコンポーネントの数と数を選択します。

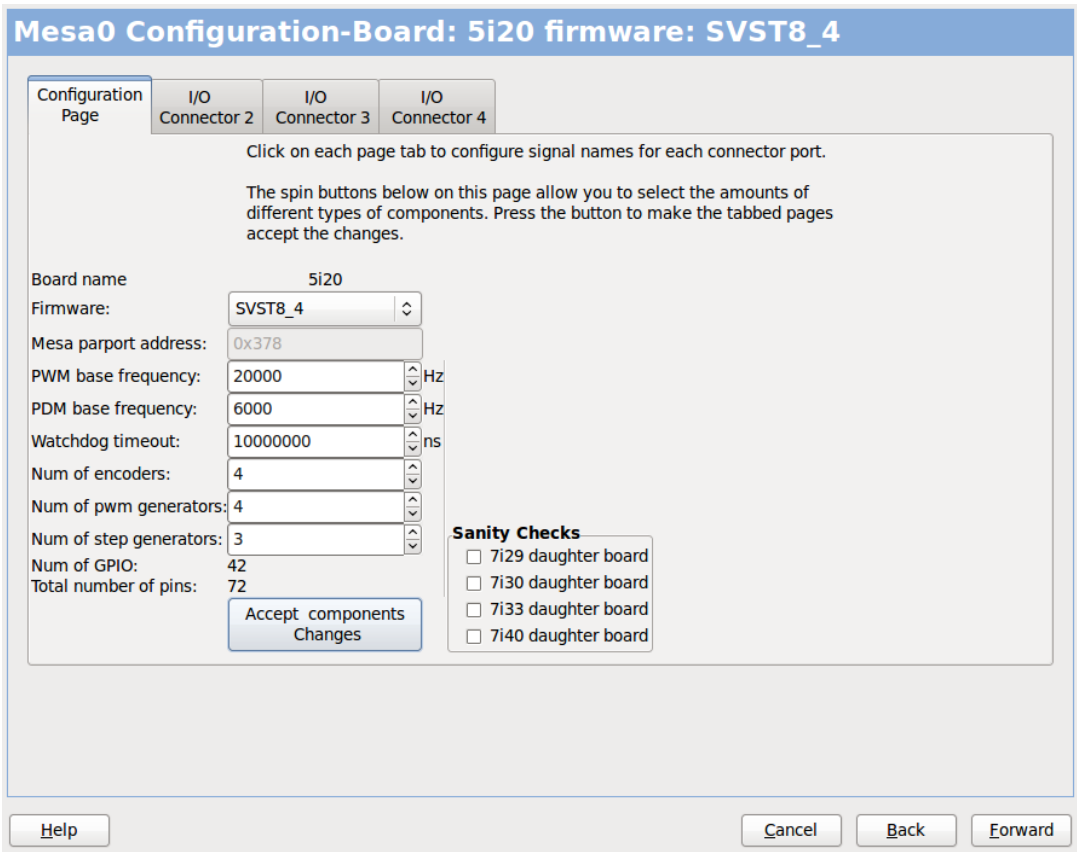


図 3-19

パーポートアドレスは、Mesa パーポートカード 7i43 でのみ使用されます。 オンボードのパラレルポートは通常 0x278 または 0x378 を使用しますが、BIOS ページからアドレスを見つけることができるはずです。 7i43 では、EPP モードを使用するためにパラレルポートが必要です。これも BIOS ページで設定されています。 PCI パラレルポートを使用している場合は、基本ページの検索ボタンを使用してアドレスを検索できます。

Note

多くの PCI カードは、EPP プロトコルを適切にサポートしていません。

PDM PWM と 3PWM の基本周波数は、リップルと直線性のバランスを設定します。 Mesa ドーターボードを使用している場合は、ボードのドキュメントに推奨事項を記載する必要があります

重要

損傷を避け、最高のパフォーマンスを得るには、これらに従うことが重要です。

The 7i33 requires PDM and a PDM base frequency of 6 mHz The 7i29 requires PWM and a PWM base frequency of 20 Khz The 7i30 requires PWM and a PWM base frequency of 20 Khz The 7i40 requires PWM and a PWM base frequency of 50 Khz The 7i48 requires UDM and a PWM base frequency of 24 Khz

ウォッチドッグタイムアウトは、コンピュータからの通信が中断された場合に、MESA ボードが出力を強制終了するまで待機する時間を設定するために使用されます。 Mesa はアクティブロー出力を使用することを覚えておいてください。つまり、出力ピンがオンのときはロー（約 0 ボルト）で、オフの場合はハイ（約 5 ボルト）で出力がオフのときに機器が安全であることを確認します（ウォッチドッグが噛まれます）。） 州。

未使用のコンポーネントの選択を解除することで、使用可能なコンポーネントの数を選択できます。 すべてのコンポーネントタイプがすべてのファームウェアで使用できるわけではありません。

コンポーネントの最大数より少ない数を選択すると、より多くの GPIO ピンを取得できます。 ドーターボードを使用する場合は、カードが使用するピンの選択を解除しないでください。 たとえば、一部のファームウェアは 2 枚の 7i33 カードをサポートしています。1 枚しかない場合は、2 番目の 7i33 をサポートしたコネクタを利用するのに十分なコンポーネントの選択を解除できます。 コンポーネントは、最初に最大の番号で数値的に選択解除され、次に番号をスキップせずに下に選択されます。 これを行うことにより、コンポーネントが必要な場所にない場合は、別のファームウェアを使用する必要があります。 ファームウェアは、コンポーネントの場所、内容、および最大量を決定します。 カスタムファームウェアが可能です。LinuxCNC 開発者と Mesa に連絡するときはよく聞いてください。 PNCconf でカスタムファームウェアを使用するには、特別な手順が必要であり、常に可能であるとは限りません。ただし、PNCconf を可能な限り柔軟にするようにしています。

これらすべてのオプションを選択した後、[コンポーネントの変更を受け入れる]ボタンを押すと、PNCconf が I / O セットアップページを更新します。 Mesa ボードに応じて、使用可能なコネクタの I / O タブのみが表示されます。

3.8.7 Mesa I / O セットアップ

タブは、Mesa ボードの入力ピンと出力ピンを構成するために使用されます。 PNCconf を使用すると、カスタム HAL ファイルで使用するカスタム信号名を作成できます。

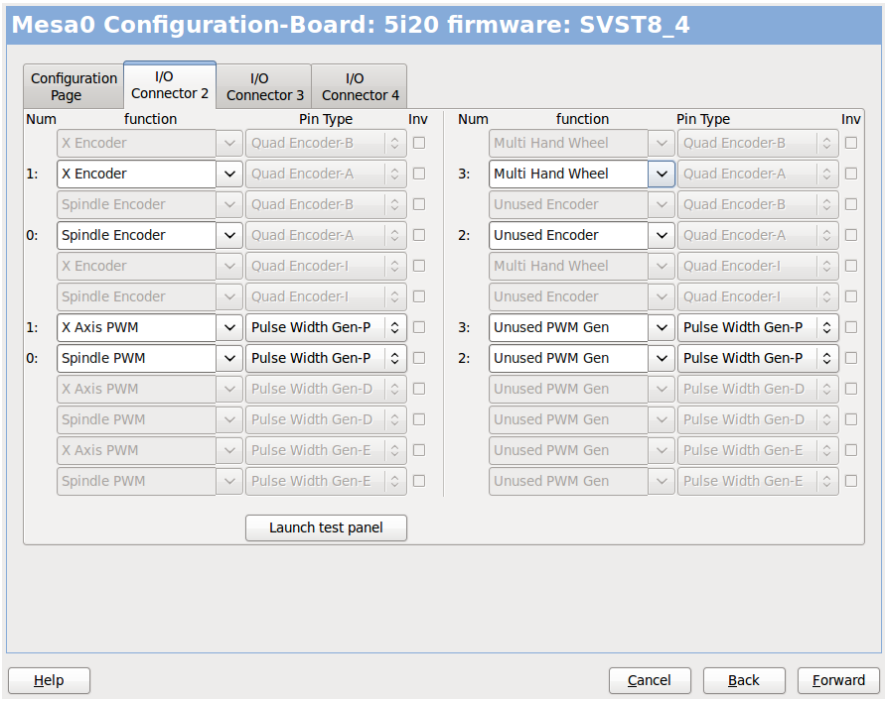


図 3-20

このファームウェアのこのタブでは、コンポーネントは 7i33 ドーターボード用にセットアップされており、通常は閉ループサーボで使用されます。 エンコーダカウンタと PWM ドライバのコンポーネント番号は番号順になっていないことに注意してください。 これはドーターボードの要件に従います。

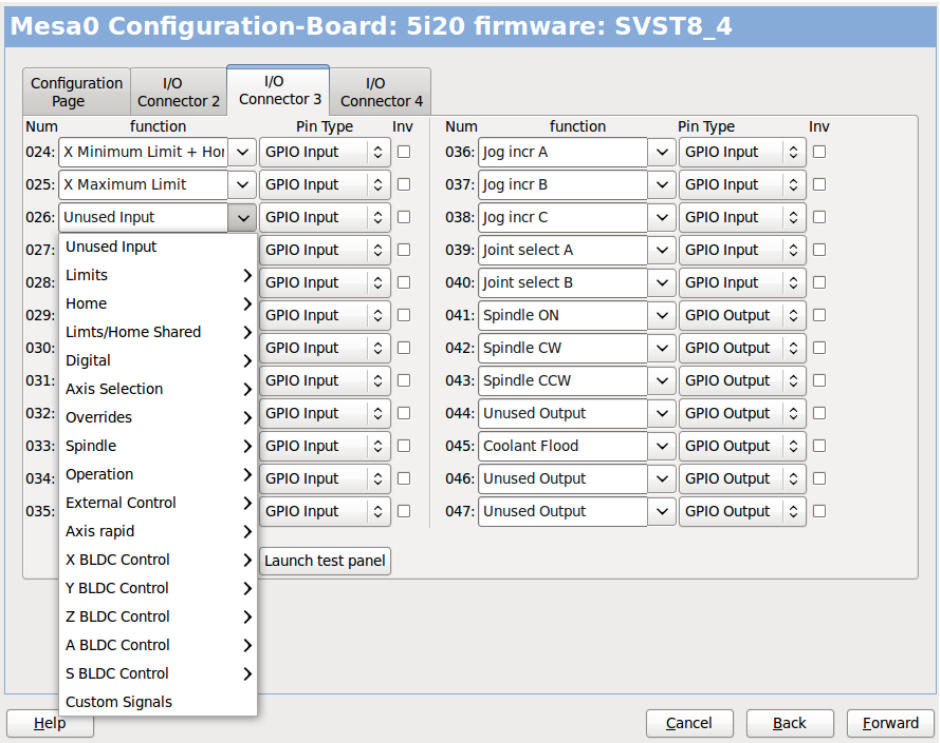


図 3-21

このタブでは、すべてのピンが GPIO です。 3 桁の数字に注意してください-それらは HAL ピン番号と一致します。 GPIO ピンは入力または出力として選択でき、反転することができます。

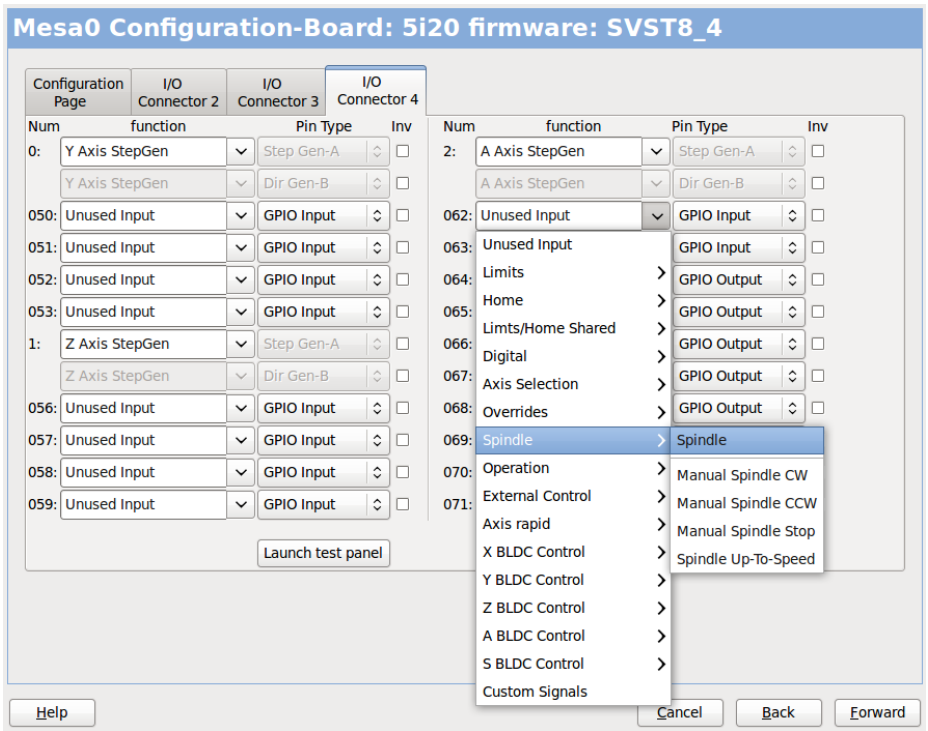
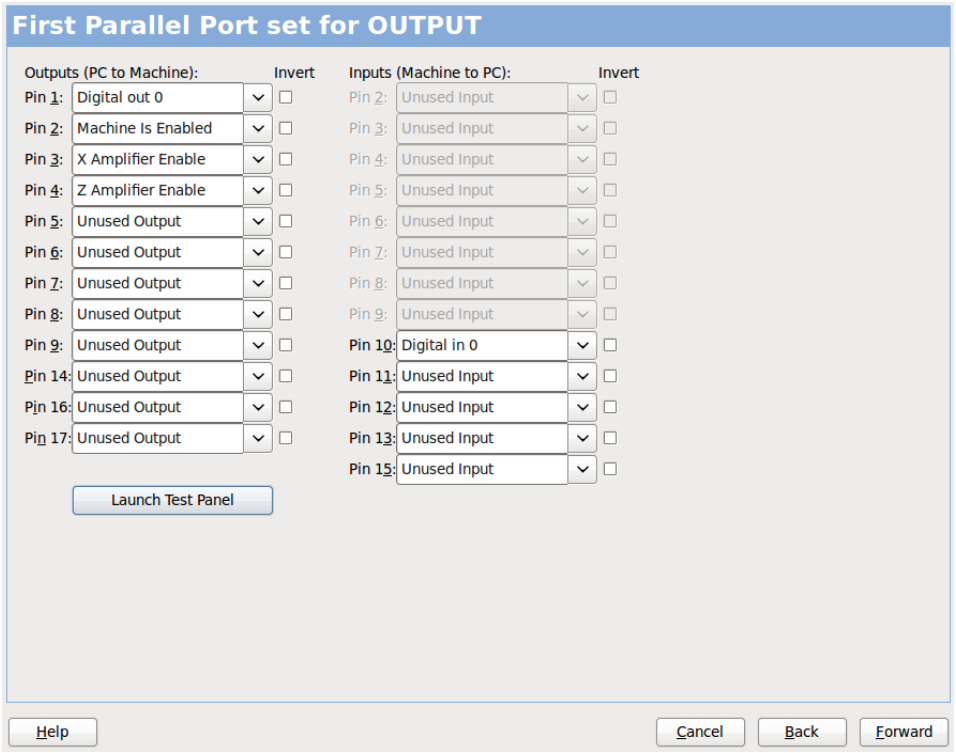


図 3-22

このタブには、ステップジェネレーターと GPIO が混在しています。 ステップジェネレータの出力ピンと方向ピンを逆にすることができます。 ステップ Gen-A ピン（ステップ出力ピン）を反転すると、ステップタイミングが変わることに注意してください。 コントローラが期待するものと一致する必要があります。

3.8.8 パーポート構成



パラレルポートは、Mesa の GPIO ピンと同様の単純な I / O に使用できます。

3.8.9 軸構成

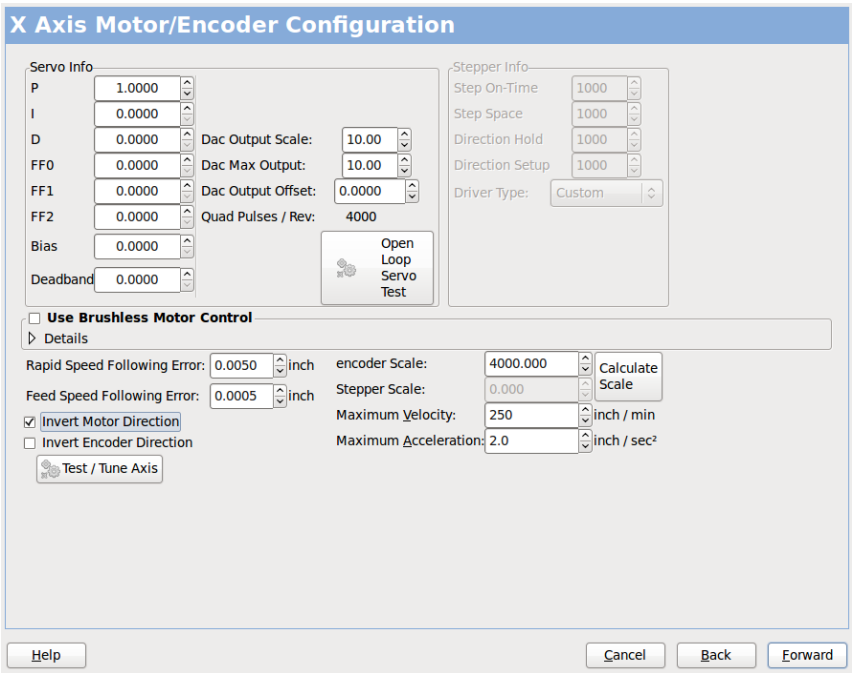


図 3-23

このページでは、モーターおよび/またはエンコーダーの組み合わせの構成とテストを行うことができます。サーボモーターを使用する場合は開ループテストを使用でき、ステッパーを使用する場合はチューニングテストを使用できます。

開ループテスト

開ループテストは、モーターとエンコーダーの方向を確認するために重要です。 正のボタンを押すと、モーターが軸を正の方向に移動し、エンコーダーも正の方向にカウントする必要があります。 軸の動きは、Machinery's Handbook[5] の標準に従う必要があります。そうでない場合、AXIS のグラフィック表示はあまり意味がありません。 うまくいけば、ヘルプページと図がこれを理解するのに役立つでしょう。 軸の方向は、テーブルの動きではなく、ツールの動きに基づいていることに注意してください。 開ループテストでは加速ランプがないため、DAC 番号を小さくして開始します。 軸を既知の距離だけ移動することで、エンコーダのスケーリングを確認できます。 エンコーダーは、エンコーダーへの電力供給方法に応じて、アンプが有効になっていない場合でもカウントする必要があります。

5) インダストリアルプレスが発行した「機械のハンドブック」の「数値制御」の章の「軸の命名法」。

警告

モーターとエンコーダーが方向のカウントに同意しない場合、PID 制御を使用するとサーボが暴走します。

現時点では、PNCconf で PID 設定をテストできないため、構成を再編集するときの設定が実際に使用されます。テストした PID 設定を入力してください。

DAC スケーリング、最大出力、およびオフセットは、DAC 出力を調整するために使用されます。

コンピュート DAC

これらの 2 つの値は、モーターアンプに出力される軸のスケールとオフセット係数です。 2 番目の値（オフセット）は、計算された出力（ボルト単位）から減算され、最初の値（スケール係数）で除算されてから、D / A コンバーターに書き込まれます。 スケール値の単位は、DAC 出力ボルトあたりの真のボルトです。 オフセット値の単位はボルトです。 これらは、DAC を線形化するために使用できます。

具体的には、出力を書き込む場合、LinuxCNC は最初に、準 SI 単位の目的の出力を、アンプ DAC のボルトなどの生のアクチュエータ値に変換します。 このスケーリングは次のようになります。スケールの値は、単位分析を実行することによって分析的に取得できます。つまり、単位は[出力 SI 単位] / [アクチュエータ単位]です。たとえば、1 ボルトが 250 mm /秒の速度になるような速度モード増幅器を備えたマシンでは、オフセットの単位はマシン単位、たとえば mm /秒であり、センサーから事前に差し引かれていることに注意してください。読み。 このオフセットの値は、アクチュエータ出力に対して 0.0 を生成する出力の値を見つけることによって取得されます。 DAC が線形化されている場合、このオフセットは通常 0.0 です。

スケールとオフセットを使用して DAC を線形化することもでき、アンプゲイン、DAC の非線形性、DAC ユニットなどの複合効果を反映する値になります。これを行うには、次の手順に従います。

- 出力のキャリブレーションテーブルを作成し、DAC を目的の電圧で駆動し、結果を測定します。

表 3.2：出力電圧の測定

Raw	測定値
-10	-9.93
-9	-8.83
0	-0.96
1	-0.03
9	9.87
10	10.07

- 最小二乗線形フィットを実行して、meas = a * raw + b となるような係数 a、b を取得します。
- 測定結果がコマンド出力と同じになるように生の出力が必要であることに注意してください。 これの意味は
  - cmd = a * raw + b
  - cmd = a * raw + b

- その結果、線形フィットからの a 係数と b 係数を、コントローラーのスケールとオフセットとして直接使用できます。

MAX OUTPUT：モーターアンプに書き込まれる PID 補正の出力の最大値（ボルト単位）。 計算された出力値は、この制限に固定されます。 制限は、生の出力ユニットにスケーリングする前に適用されます。 値はプラス側とマイナス側の両方に対称的に適用されます。

チューニングテスト残念ながら、チューニングテストはステッパーベースのシステムでのみ機能します。 軸の方向が正しいことを再度確認します。 次に、軸を前後に動かしてシステムをテストします。加速または最高速度が高すぎると、ステップが失われます。 ジョギング中は、加速度の低い軸が停止するまで時間がかかる場合がありますのでご注意ください。 このテスト中、リミットスイッチは機能しません。 テスト動作の両端が来るように一時停止時間を設定できます。 これにより、ダイヤルゲージを設定して読み取り、手順を失っていないかどうかを確認できます。

ステッパーのタイミングステッパーのタイミングは、ステップコントローラーの要件に合わせて調整する必要があります。 Pncconf は、いくつかのデフォルトのコントローラータイミングを提供するか、カスタムタイミング設定を許可します。 いくつかの既知のタイミング番号については、
http：//wiki.linuxcnc.org/cgi-bin/wiki.pl？Stepper_Drive_Timing を参照してください（把握したものを自由に追加してください）。 疑わしい場合は、5000 などの大きな数値を使用すると、最大速度のみが制限されます。

ブラシレスモーター制御これらのオプションは、特別なファームウェアとドーターボードを使用してブラシレスモーターの低レベル制御を可能にするために使用されます。 また、あるメーカーから別のメーカーへの HALL センサーの変換も可能です。 部分的にのみサポートされており、HAL 接続を完了するために 1 つ必要になります。 詳細については、メールリストまたはフォーラムにお問い合わせください。

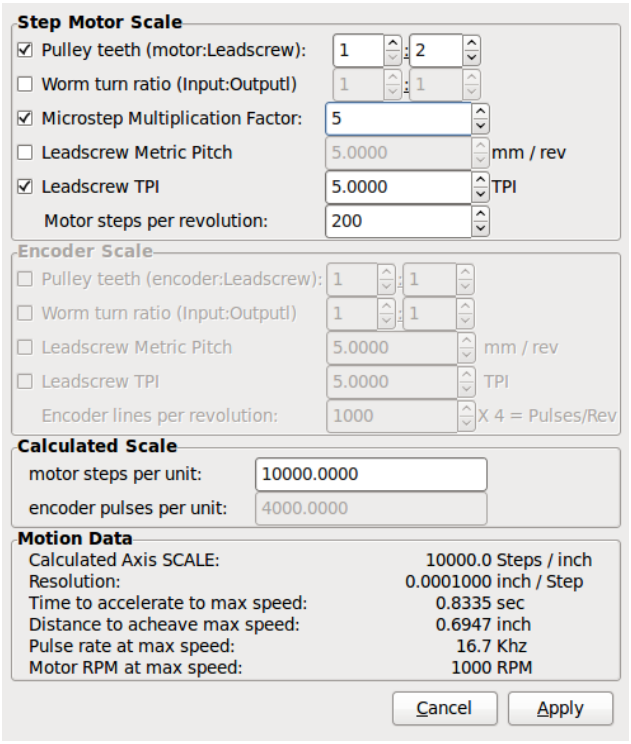


図 3-24

スケール設定は直接入力することも、スケール計算ボタンを使用して支援することもできます。 チェックボックスを使用して、適切な計算を選択します。 プーリーの歯には、ギア比ではなく歯の数が必要であることに注意してください。 ウォームターン比は、ギア比が必要な場合とは正反対です。 スケールプレスに満足している場合は適用し、そうでない場合はキャンセルを押してスケールを直接入力します。

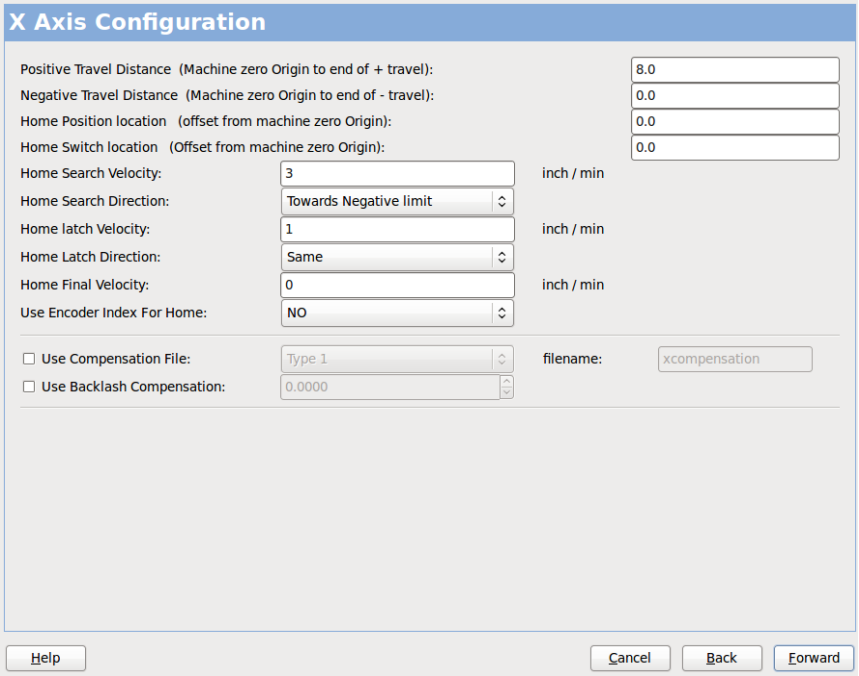


図 3-25

ホームスイッチとリミットスイッチの 2 つの例については、図のタブも参照してください。 これらは、ホーミングと制限を設定するためのさまざまな方法の 2 つの例です。

重要

軸を正しい方向に動かすことから始めることは非常に重要です。さもないと、正しくホーミングするのは非常に困難です。

正と負の方向は、マシニストハンドブックの表ではなくツールを参照していることを忘れないでください。

典型的な膝またはベッドミル

- TABLE が移動すると、それは正の Y 方向になります
- TABLE が左に移動すると、それは正の X 方向になります
- TABLE が下に移動すると、それは正の Z 方向になります
- HEAD が上に移動すると、それは正の Z 方向になります

典型的な旋盤で

- ツールがチャックから離れて右に移動するとき
- それは正の Z 方向です
- ツールがオペレーターに向かって移動するとき
- それは正の X 方向です。 一部の旋盤には X があります
- 反対側（例：裏側のツール）、それは問題なく動作しますが
- これを反映するように AXIS のグラフィック表示を行うことはできません。

ホーミングおよび/またはリミットスイッチを使用する場合、LinuxCNC は、スイッチが押されている/トリップされているときに HAL 信号が真であると想定します。 リミットスイッチの信号が間違っている場合、LinuxCNC はマシンが常にリミットの終わりにあると見なします。 ホームスイッチの検索ロジックが間違っていると、LinuxCNC は間違った方向にホームしているように見えます。 それが実際に行っているのは、ホームスイッチをバックオフしようとしていることです。

リミットスイッチの位置を決定します。

リミットスイッチは、電気的な問題が発生した場合のソフトウェア制限のバックアップです。 サーボ暴走。機械が軸の動きの物理的な端に当たらないように、リミットスイッチを配置する必要があります。 高速で移動すると、軸が接触点を超えて惰性走行することを忘れないでください。 リミットスイッチは、マシンのローでアクティブにする必要があります。 例えば。 電力は常にスイッチを通過します-電力の損失（スイッチが開いている）が作動します。 逆に配線することもできますが、これはフェイルセーフです。 LinuxCNC の HAL 信号がアクティブハイになるように、これを反転する必要がある場合があります。TRUE は、スイッチがトリップしたことを意味します。 LinuxCNC を起動するときに、制限内の警告が表示され、軸がスイッチをトリップしていない場合は、信号を反転することがおそらく解決策です。 （HALMETER を使用して、対応する HAL 信号を確認します（例：joint.0.pos-lim-sw-in X 軸の正のリミットスイッチ））

ホームスイッチの場所を決定します。

リミットスイッチを使用している場合ホームスイッチとして使用することもできます。 別のホームスイッチは、使用中の長軸が通常リミットスイッチから遠く離れている場合、または軸を端に移動すると材料との干渉の問題が発生する場合に役立ちます。 たとえば、旋盤のシャフトが長いと、工具がシャフトに当たらないと限界に戻るのが難しくなるため、中央に近い別のホームスイッチの方が適している場合があります。 インデックス付きのエンコーダを使用している場合、ホームスイッチはコースホームとして機能し、インデックスが実際のホームロケーションになります。

MACHINEORIGIN の位置を決定します。

MACHINE ORIGIN は、LinuxCNC がすべてのユーザー座標系を参照するために使用するものです。 特定の場所に配置する必要がある理由はほとんど考えられません。 MACHINE COORDINATE システムにアクセスできる G コードはごくわずかです。（G53、G30、G28）G30 での工具交換オプションを使用する場合は、工具

交換位置に原点があると便利です。 慣例により、ホームスイッチに ORIGIN を配置するのが最も簡単な場合があります。

（最終的な）ホームポジションを決定します。

これは、LinuxCNC が ORIGIN の場所を特定した後、キャリッジを一貫した便利な位置に配置するだけです。

正/負の軸の移動距離を測定/計算します。

軸を原点に移動します。 可動スライドに基準をマークし、可動でないサポート（それらが一列に並ぶように）が機械を限界の終わりまで動かします。 移動距離の 1 つであるマーク間の測定。 テーブルを移動の反対側に移動します。 マークを再度測定します。 それがもう 1 つの移動距離です。 ORIGIN が限界の 1 つにある場合、その移動距離はゼロになります。

（機械）ORIGIN

原点はマシンのゼロ点です。（カッター/マテリアルを設定したゼロ点ではありません）。 LinuxCNC は、このポイントを使用して、他のすべてを参照します。 ソフトウェアの制限内にある必要があります。 LinuxCNC は、ホームスイッチの位置を使用して原点位置を計算します（ホームスイッチを使用する場合、またはホームスイッチを使用しない場合は手動で設定する必要があります。

移動距離

これは、軸が各方向に移動できる最大距離です。 これは、原点からリミットスイッチまで直接測定できる場合とできない場合があります。 正と負の移動距離は、合計移動距離になります。

正の移動距離

これは、軸が原点から正の移動距離まで移動する距離、または合計移動から負の移動距離を引いた距離です。 原点が正の限界にある場合は、これをゼロに設定します。 は常にゼロまたは正の数になります。

負の移動距離

これは、軸が原点から負の移動距離まで移動する距離です。 または、総移動量から正の移動距離を引いたもの。 原点が負の限界にある場合は、これをゼロに設定します。 これは常にゼロまたは負の数になります。 この負の PNCconf を作成するのを忘れた場合、内部でそれを行います。

（最終）ホームポジション

これは、ホームシーケンスが終了する位置です。 これは原点から参照されるため、原点のどちら側にあるかに応じて負または正になります。 （最終的な）ホームポジションで、原点に到達するために正の方向に移動する必要がある場合、数値は負になります。

ホームスイッチの場所

これは、ホームスイッチから原点までの距離です。 原点のどちら側にあるかによって、負または正になる可能性があります。 ホームスイッチの場所にいるときに、原点に到達するために正の方向に移動する必要がある場合、数値は負になります。 これをゼロに設定すると、原点はリミットスイッチの位置になります（さらに、使用されている場合はインデックスを見つけるための距離）

ホーム検索速度

1 分あたりの単位でのコースホーム検索速度。

ホーム検索方向

ホームスイッチの検索方向を負（つまり、負のリミットスイッチに向かって）または正（つまり、正のリミットスイッチに向かって）に設定します。

ホームラッチ速度

1 分あたりの単位でのファインホーム検索速度

ホーム最終速度

ラッチ位置から（最終）ホーム位置までの速度（単位/分）。 最大高速の場合は 0 に設定します

ホームラッチ方向

ラッチ方向を検索方向と同じまたは反対に設定できます。

ホームにエンコーダインデックスを使用

LinuxCNC は、ホーミングのラッチ段階でエンコーダインデックスパルスを検索します。

補償ファイルを使用する

Comp ファイル名とタイプを指定できます。 高度な補正を可能にします。 INI の章の AXIS セクションを参照してください。

バックラッシュ補正を使用する

簡単なバックラッシュ補正を設定できます。 補正ファイルとの併用はできません。 INI の章の AXIS セクションを参照してください。

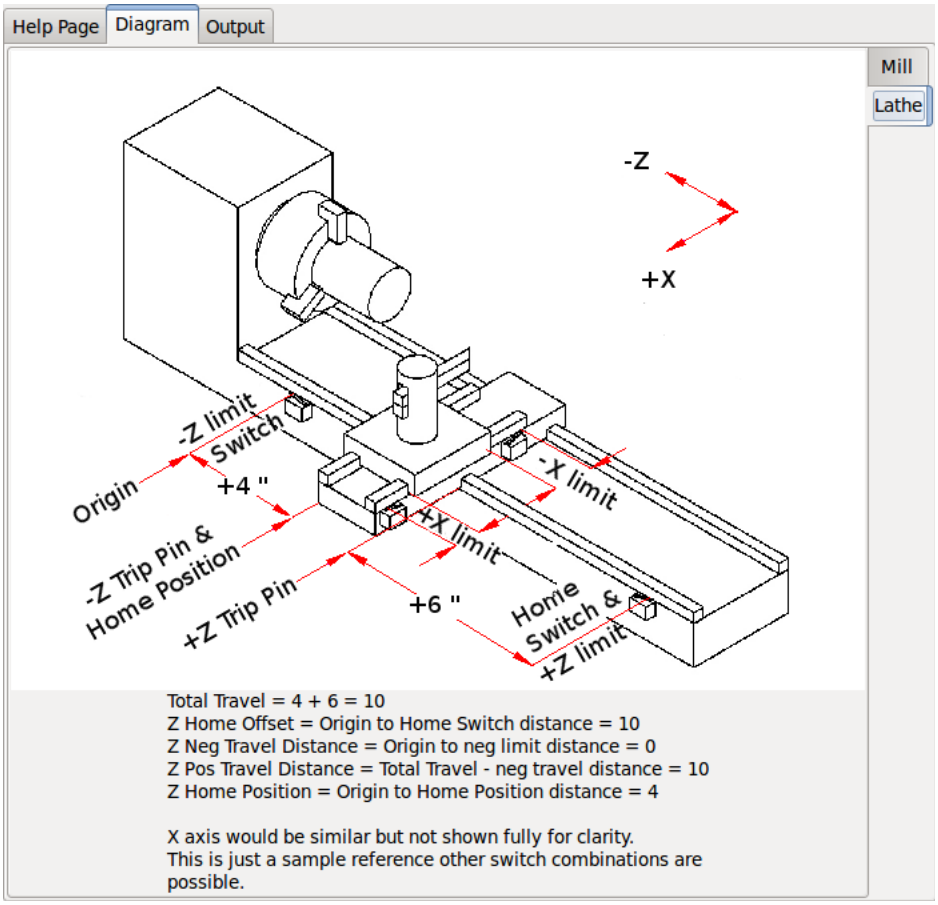


図 3-26

これらの図は、リミットスイッチと標準軸の移動方向の例を示すのに役立ちます。 この例では、Z 軸は 2 つのリミットスイッチであり、正のスイッチはホームスイッチとして共有されています。 MACHINE ORIGIN（ゼロ点）は負の限界にあります。 キャリッジの左端は負のトリップピンで、右端は正のトリップピンです。 FINAL HOME POSITION は、正の側で ORIGIN から 4 インチ離れていることを望みます。 キャリッジが正の限界に移動した場合、負の限界と負のトリップピンの間を 10 インチ測定します。

3.8.10 スピンドル構成

スピンドル信号を選択した場合、このページでスピンドル制御を構成できます。

ヒント

このページのオプションの多くは、前のページで適切なオプションが選択されていない限り表示されません。

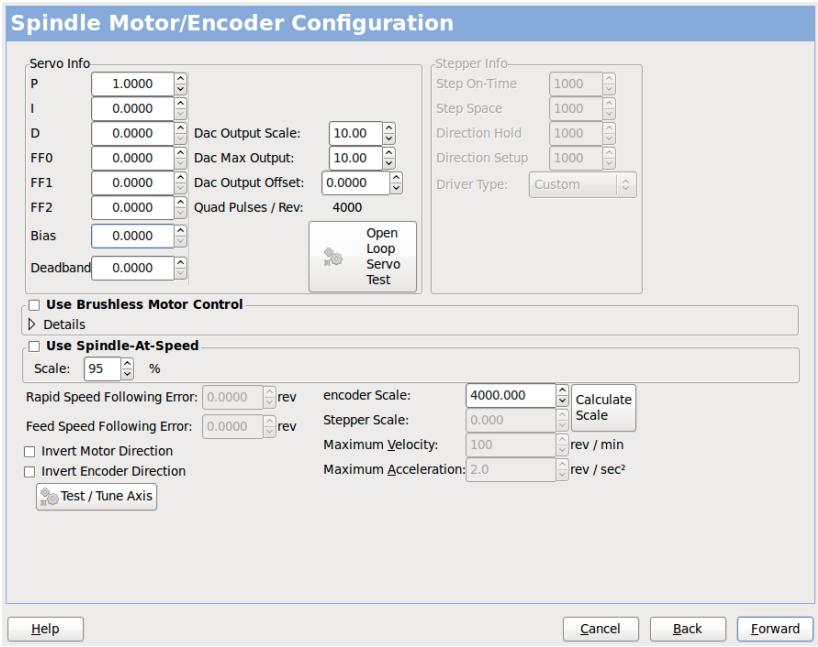


図 3-27

このページは、軸モーター構成ページに似ています。

いくつかの違いがあります：

- ステッパー駆動のスピンドルを選択しない限り、加速や速度の制限はありません。
- ギアチェンジやレンジのサポートはありません。
- VCP スピンドル表示オプションを選択した場合は、スピンドルの速度スケールとフィルター設定が表示される場合があります。
- Spindle-at-speed を使用すると、LinuxCNC は、スピンドルが要求された速度になるまで待機してから、軸を移動できます。 これは、一定の表面送りと大きな速度直径変化を伴う旋盤で特に便利です。 エンコーダフィードバックまたは通常 VFD ドライブに接続されているデジタルスピンドルアットスピード信号のいずれかが必要です。
- エンコーダフィードバックを使用する場合は、実際の速度を要求された速度にどれだけ近づけて速度と見なすかを指定する、スピンドル速度スケール設定を選択できます。
- エンコーダフィードバックを使用している場合、VCP 速度の表示が不安定になる可能性があります。フィルタ設定を使用して表示をスムーズにすることができます。 エンコーダスケールは、使用するエンコーダカウント/ギアに合わせて設定する必要があります。
- スピンドルエンコーダーに単一の入力を使用している場合は、setp hm2_7i43.0.encoder.00.counter-mode 1（ボード名とエンコーダー番号を要件に合わせて変更）という行をカスタム HAL ファイルに追加する必要

があります。 カウンターモードの詳細については、Hostmot2 のエンコーダーセクションを参照してください。

3.8.11 高度なオプション

これにより、HALUI コマンドの設定と、クラシックラダーおよびサンプルラダープログラムのロードが可能になります。 軸のゼロ調整などの GLADEVCP オプションを選択した場合は、コマンドが表示されます。 カスタム halcmd の使用の詳細については、HALUI の章を参照してください。 いくつかのラダープログラムオプションがあります。 Estop プログラムを使用すると、外部 ESTOP スイッチまたは GUI フロントエンドが Estop をスローできます。 また、時限潤滑ポンプ信号があります。 Z 自動タッチオフには、タッチオフプレート、GLADE VCP タッチオフボタン、および現在のユーザー原点をゼロに設定してすばやくクリアするための特別な HALUI コマンドがあります。 シリアル modbus プログラムは、基本的に、シリアル modbus 用のクラシックラダーをセットアップする空白のテンプレートプログラムです。 マニュアルの Classicladder の章を参照してください。

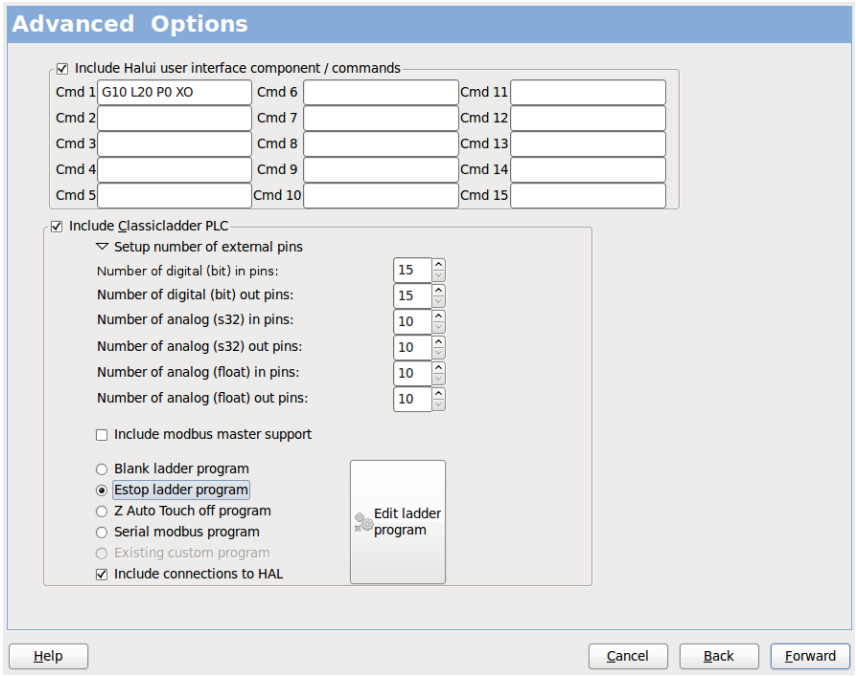


図 3-28

3.8.12 HAL コンポーネント

このページでは、カスタム HAL ファイルに必要な HAL コンポーネントを追加できます。 このようにして、ユーザーが必要とするコンポーネントを許可しながら、メインの HAL ファイルを手動で編集する必要はありません。

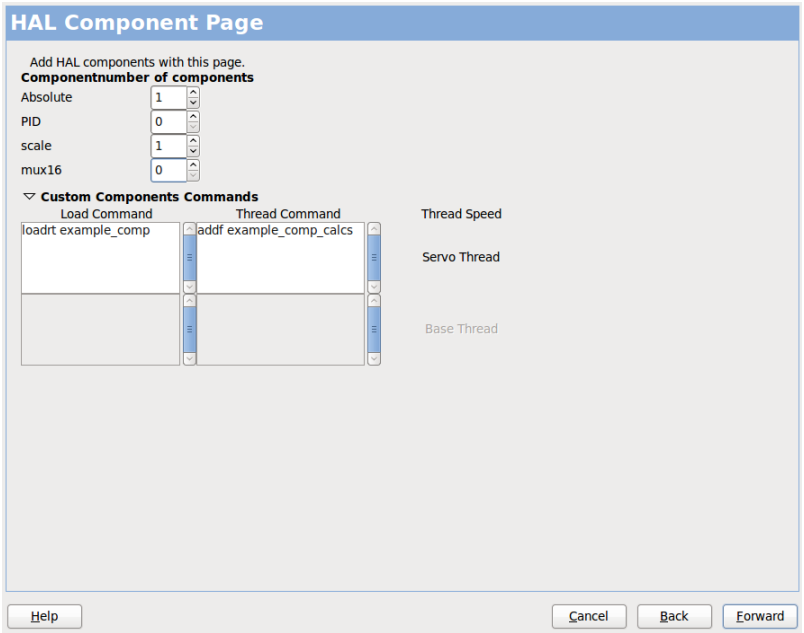


図 3-29

最初の選択は、pncconf が内部で使用するコンポーネントです。 カスタム HAL ファイルのコンポーネントの追加インスタンスをロードするように pncconf を構成できます。

カスタムファイルに必要なインスタンスの数を選択します。pncconf はそれらの後に必要なものを追加します。

つまり、2 つ必要で、pncconf に 1 つ必要な場合、pncconf は 3 つのインスタンスをロードし、最後のインスタンスを使用します。

カスタムコンポーネントコマンド

この選択により、pncconf が使用しない HAL コンポーネントをロードできるようになります。 見出しの loading コマンドの下に loadrt または loadusr コマンドを追加します。見出しの Thread コマンドの下に addf コマンドを追加します。 コンポーネントは、thread コマンドで書き込んだ順序で、入力の読み取りと出力の書き込みの間にスレッドに追加されます。

3.8.13 PNCconf の高度な使用法

PNCconf は、ユーザーによる柔軟なカスタマイズを可能にするために最善を尽くします。 PNCconf は、カスタム信号名、コンポーネントのカスタムロード、カスタム HAL ファイル、およびカスタムファームウェアをサポートしています。

ユーザーのカスタム HAL ファイルに対して、選択したオプションに関係なく PNCconf が常に提供する信号名もあります。いくつかの考えでは、後で PNCconf で別のオプションを選択しても、ほとんどのカスタマイズは機能するはずです。

最終的に、カスタマイズが PNCconf のフレームワークの範囲を超えている場合は、PNCconf を使用して基本構成を構築するか、LinuxCNC のサンプル構成の 1 つを使用して、必要に応じて手動で編集できます。

カスタム信号名

コンポーネントをカスタム HAL ファイル内の何かに接続する場合は、コンボ入力ボックスに一意の信号名を入力します。 特定のコンポーネントは、カスタム信号名に末尾を追加します。

エンコーダーは<カスタム名> +を追加します：

- 位置
- カウント
- 速度
- ndex-enable
- リセット

ステッパーは追加します：

- 有効
- カウント
- position-cmd
- position-fb
- velocity-fb

PWM 追加：

- 有効
- 価値

GPIO ピンには、入力された信号名が接続されているだけです。

このようにして、カスタム HAL ファイルでこれらの信号に接続し、後でそれらを移動するオプションを使用できます。

カスタム信号名

[Hal コンポーネント]ページを使用して、ユーザーがカスタマイズに必要なコンポーネントをロードできます。

カスタムファームウェアのロード

PNCconf は、システム上のファームウェアを検索してから、理解できるものに変換できる XML ファイルを探します。 これらの XML ファイルは、LinuxCNC チームから公式にリリースされたファームウェアに対してのみ提供されます。 カスタムファームウェアを利用するには、PNCconf が理解できるアレイに変換し、そのファイルパスを PNCconf の設定ファイルに追加する必要があります。 デフォルトでは、このパスはデスクトップで custom_firmware という名前のフォルダーと firmware.py という名前のファイルを検索します。

非表示の設定ファイルはユーザーのホームファイルにあり、.pncconf-preferences という名前で、表示および編集するには[非表示のファイルを表示]を選択する必要があります。 このファイルの内容は、PNCconf を最初にロードしたときに確認できます。ヘルプボタンを押して、出力ページを確認してください。

カスタムファームウェアの変換に関する情報については、LinuxCNC メールリストまたはフォーラムで質問してください。 すべてのファームウェアを PNCconf で利用できるわけではありません。

カスタム HAL ファイル

HAL コマンドを追加するために使用できる 4 つのカスタムファイルがあります。

- custom.hal は、GUI フロントエンドのロード後に実行する必要のない HAL コマンド用です。 これは、configurationnamedHAL ファイルの後に実行されます。
- custom_postgui.hal は、AXIS のロードまたはスタンドアロンの PYVCP ディスプレイのロード後に実行する必要があるコマンド用です。
- custom_gvcp.hal は、gladeVCP がロードされた後に実行する必要があるコマンド用です。
- shutdown.hal は、LinuxCNC が制御された方法でシャットダウンしたときに実行するコマンド用です。

3.9 Linux の FAQ

これらは、Linux ユーザーにとって初めての基本的な Linux コマンドとテクニックです。 より完全な情報は、Web 上または man ページを使用して見つけることができます。

3.9.1 自動ログイン

Ubuntu LiveCD を使用して LinuxCNC をインストールする場合、デフォルトでは、コンピューターの電源を入れるたびにログインする必要があります。 自動ログインを有効にするには、[システム]> [管理]> [ログインウィンドウ]に移動します。 新規インストールの場合、ログインウィンドウがポップアップするまでに 1〜3 秒かかる場合があります。 ログインウィンドウの設定ウィンドウにアクセスするには、インストールに使用したパスワードが必要です。 [セキュリティ]タブで、[自動ログインを有効にする]をオフにし、リストからユーザー名を選択します（それはあなたです）。

3.9.2 自動起動

コンピューターの電源を入れた後、LinuxCNC を構成で自動的に起動するには、[システム]> [設定]> [セッション]> [スタートアップアプリケーション]に移動し、[追加]をクリックします。 構成を参照し、.ini ファイルを選択します。 ファイルピッカーダイアログが閉じたら、.ini ファイルへのパスの前に linuxcnc とスペースを追加します。

例：

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

3.9.3 ターミナル

dmesg でカーネルメッセージバッファをチェックするなど、ターミナルから多くのことを行う必要があります。 Ubuntu と LinuxMint には、キーボードショートカット Ctrl + Alt + t があります。 最近のほとんどのファイルマネージャは、ターミナルを開くための正しいキーをサポートしています。ファイル名ではなく、空白の領域またはディレクトリを右クリックすることを確認してください。 ほとんどの OS には、通常はアクセサリのメニュー項目として端末があります。

3.9.4 マニュアルページ

マニュアルページ（マニュアルページの略）は、Unix または Linux のような Unix ライクなオペレーティングシステムで通常見られるソフトウェアドキュメンテーションの形式です。

マニュアルページを表示するには、ターミナルを開いて、ターミナルウィンドウの find コマンドに関する情報を確認します。次のように入力します。

```
man find
```

PageUp キーと PageDown キーを使用してマニュアルページを表示し、Q キーを使用して表示を終了します。

Note

ターミナルからマニュアルページを表示すると、期待されるマニュアルページが表示されない場合があります。 たとえば、man abs と入力すると、LinuxCNCabs ではなく Cabs が表示されます。 LinuxCNC のマニュアルページを HTML ドキュメントで表示することをお勧めします。

3.9.5 モジュールのリスト

トラブルシューティング時に、ロードされているモジュールのリストを取得する必要がある場合があります。 ターミナルウィンドウで次のように入力します。

```
lsmod
```

lsmod からの出力をターミナルウィンドウのテキストファイルに送信する場合は、次のように入力します。

```
lsmod > mymod.txt
```

ターミナルウィンドウを開いたときにディレクトリを変更しなかった場合、結果のテキストファイルはホームディレクトリに配置され、mymod.txt または任意の名前が付けられます。

3.9.6 ルートファイルの編集

ファイルブラウザを開いて、ファイルの所有者が root であることがわかったら、そのファイルを編集するために追加の手順を実行する必要があります。 一部のルートファイルを編集すると、悪い結果になる可能性がありま

す。 ルートファイルを編集するときは注意してください。 通常、ほとんどのルートファイルを開いて表示できますが、読み取り専用モードで開きます。

1.1.1.1 コマンドラインウェイ

ターミナルを開いて次のように入力します

```
sudo gedit
```

[ファイル]> [開く]> [編集]でファイルを開きます

3.9.6.1 GUI ウェイ

1. デスクトップを右クリックして、[ランチャーの作成]を選択します
2. sudoedit のように名前を入力します
3. コマンドとして gksudo "gnome-open%u"と入力し、ランチャーをデスクトップに保存します
4. ファイルをランチャーにドラッグして開き、編集します

3.9.6.2 ルートアクセス

Ubuntu では、ターミナルウィンドウに「sudo-i」と入力してからパスワードを入力することで、root になることができます。 自分が何をしているのかわからない場合は、ルートとして物事を実際に汚す可能性があるため、注意してください。

3.9.7 ターミナルコマンド

1.1.1.1 作業ディレクトリ

ターミナルウィンドウで現在の作業ディレクトリへのパスを見つけるには、次のように入力します。

```
pwd
```

3.9.7.1 ディレクトリの変更

ターミナルウィンドウで 1 つ上のレベルに移動するには、次のように入力します。

```
cd ..
```

ターミナルウィンドウで 2 レベル上に移動するには、次のように入力します。

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

ターミナルウィンドウの linuxcnc / configs サブディレクトリに移動するには、次のように入力します。

```
cd linuxcnc/configs
```

3.9.7.2 ディレクトリ内のファイルの一覧表示

ターミナルウィンドウですべてのファイルとサブディレクトリのリストを表示するには、次のように入力します。

```
dir
```

または

```
ls
```

3.9.7.3 ファイルの検索

find コマンドは、新しい Linux ユーザーにとって少し混乱する可能性があります。 基本的な構文は次のとおりです。

```
find starting-directory parameters actions
```

たとえば、linuxcnc ディレクトリ内のすべての.ini ファイルを検索するには、最初に pwd コマンドを使用してディレクトリを検索する必要があります。 新しいターミナルウィンドウを開き、次のように入力します。

```
pwd
```

また、pwd は次の結果を返す可能性があります。

```
/home/joe
```

この情報を使用して、次のようにコマンドをまとめます。

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

-name は探しているファイルの名前であり、-print は結果をターミナルウィンドウに出力するように指示します。 \ *。ini は、拡張子が.ini のすべてのファイルを返すように find に指示します。 シェルのメタ文字をエスケープするには、円記号が必要です。 検索の詳細については、検索のマニュアルページを参照してください。

3.9.7.4 テキストの検索

```
grep -irl 'text to search for' *
```

これにより、大文字と小文字を区別せずに、現在のディレクトリとその下のすべてのサブディレクトリで検索するテキストを含むすべてのファイルが検索されます。 -i は大文字と小文字を区別せず、-r は再帰的です（検索にすべてのサブディレクトリを含めます）。 -l オプションはファイル名のリストを返します。-l をオフのままにすると、「検索するテキスト」の各オカレンスが見つかったテキストも取得されます。 *は、すべてのファイルを検索するためのワイルドカードです。 詳細については、grep の man ページを参照してください。

3.9.7.5 診断メッセージ

診断メッセージを表示するには、コマンドウィンドウから「dmesg」を使用します。 診断メッセージをファイルに保存するには、次のようにリダイレクト演算子>を使用します。

```
dmesg > bootmsg.txt
```

このファイルの内容をコピーしてオンラインで貼り付け、問題の診断を支援しようとしている人々と共有することができます。 メッセージバッファをクリアするには、次のように入力します。

```
sudo dmesg -c
```

これは、LinuxCNC を起動する直前に実行すると役立つ場合があります。これにより、LinuxCNC の現在の起動に関連する情報のみが記録されます。

組み込みのパラレルポートアドレスを見つけるには、grep を使用して dmesg から情報をフィルタリングします。

起動後、ターミナルを開いて次のように入力します。

```
dmesg|grep parport
```

3.9.8 便利なアイテム

1.1.1.1 ターミナルランチャー

画面上部のパネルバーにターミナルランチャーを追加する場合は、通常、画面上部のパネルを右クリックして、[パネルに追加]を選択します。 [カスタムアプリケーションランチャー]を選択して[追加]を選択します。 名前を付けて、コマンドボックスに gnome-terminal を入力します。

3.9.9 ハードウェアの問題

1.1.1.1 ハードウェア情報

ターミナルウィンドウでマザーボードに接続されているハードウェアを確認するには、次のように入力します。

```
lspci -v
```

3.9.9.1 モニターの解像度

インストール中に、Ubuntu はモニター設定を検出しようとします。 これが失敗した場合、800x600 の最大解像度の汎用モニターが残ります。

これを修正するための手順はここにあります：

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

3.9.10 パス

相対パス相対パスは、ini ファイルを含むディレクトリであるスタートアップディレクトリに基づいています。相対パスを使用すると、構成の再配置が容易になりますが、Linux パス指定子を十分に理解している必要があります。

```
./f0 is the same as f0, e.g., a file named f0 in the startup directory
../f1 refers to a file f1 in the parent directory
../../f2 refers to a file f2 in the parent of the parent directory
../../../f3 etc.
```

3.10 旋盤ユーザー情報

3.10.1 旋盤モード

CNC マシンが旋盤の場合、LinuxCNC から最良の結果を得るには、ini ファイルにいくつかの特定の変更を加える必要があります。

AXIS ディスプレイを使用している場合は、Axis に旋盤ツールを正しく表示させます。 詳細については、「INI 構成」セクションを参照してください。

旋盤モード用に AXIS を設定します。

```
[DISPLAY]
# Tell the Axis Display our machine is a lathe.
LATHE = TRUE
```

Axis の旋盤モードはデフォルトの平面を G18（XZ）に設定しません。 次のように、各 gcode ファイルのプリアンブルでプログラムするか、（より適切に）ini ファイルに追加する必要があります。

```
[RS274NGC]
# g-code modal codes (modes) that the interpreter is initialized with on startup
RS274NGC_STARTUP_CODE = G18 G20 G90
```

Gmoccapy を使用している場合は、Gmoccapy 旋盤のセクションを参照してください。

3.10.2 旋盤工具テーブル

「ツールテーブル」は、各ツールに関する情報を含むテキストファイルです。 このファイルは構成と同じディレクトリにあり、デフォルトでは「tool.tbl」と呼ばれます。 ツールはツールチェンジャー内にあるか、手動で変更されている可能性があります。 ファイルはテキストエディタで編集するか、G10 L1、L10、L11 を使用して更新できます。 Axis ディスプレイには組み込みのツールテーブルエディタもあります。 ツールテーブルのエントリの最大数は 56 です。ツールとポケットの最大数は 99999 です。

LinuxCNC の以前のバージョンには、ミルと旋盤用に 2 つの異なるツールテーブル形式がありましたが、2.4.x リリース以降、すべてのマシンで 1 つのツールテーブル形式が使用されています。 マシンに関係のない、または使用する必要のないツールテーブルの部分は無視してください。 ツールテーブル形式の詳細については、ツールテーブルのセクションを参照してください。

3.10.3 Lathe Tool Orientation

次の図は、旋盤工具の方向と、各方向の中心線角度、および FRONTANGLE と BACKANGLE に関する情報を示しています。 FRONTANGLE と BACKANGLE は、Z +に平行な線から時計回りに始まります。

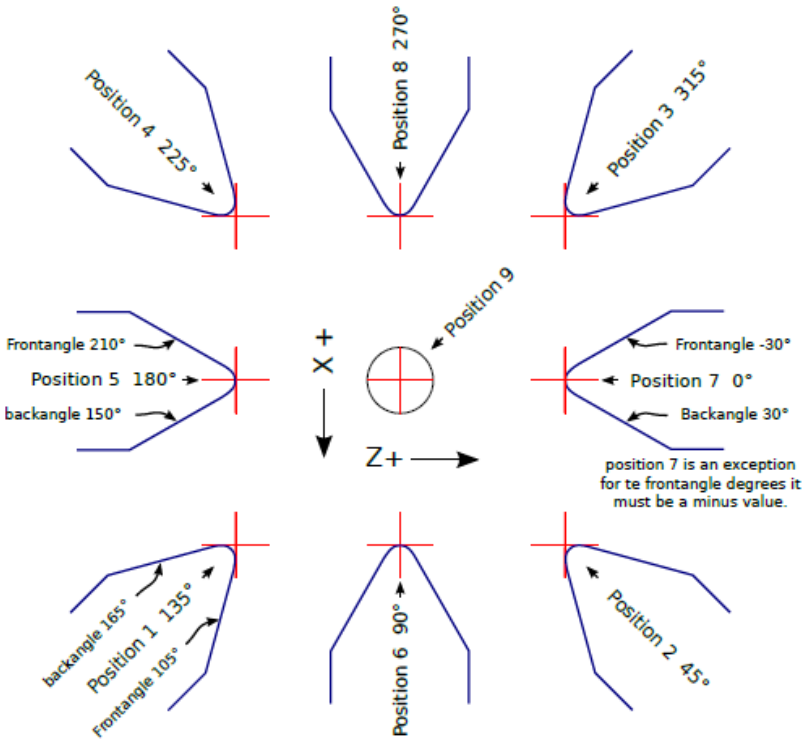
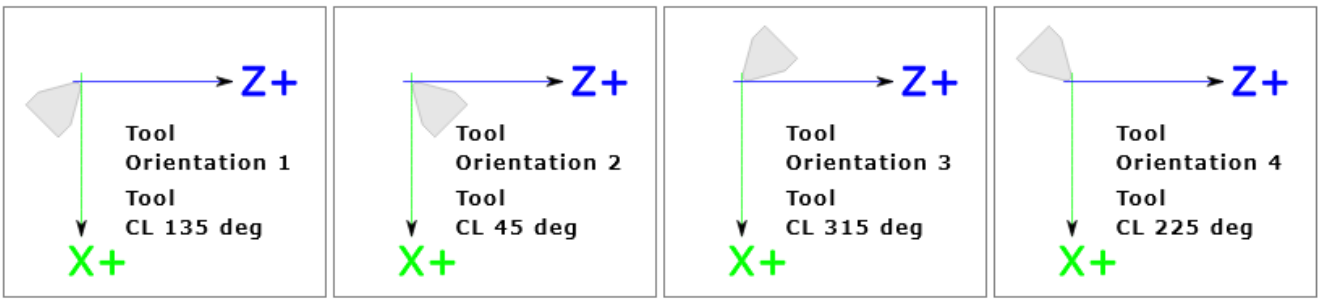


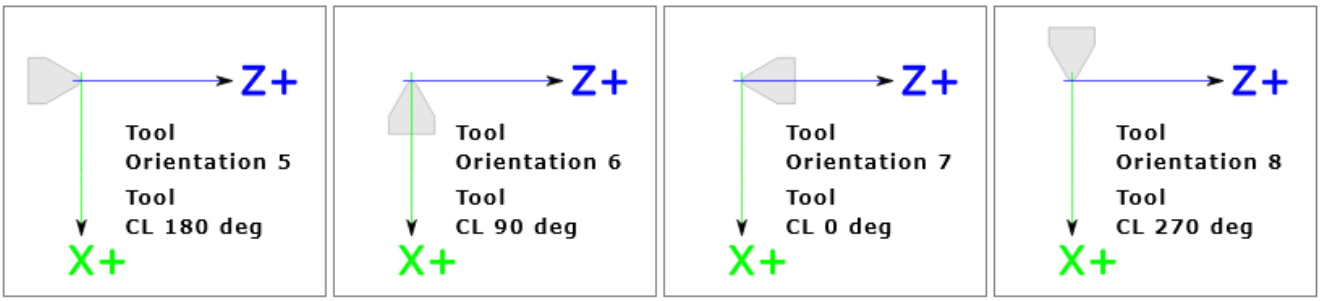
図 3-30

AXIS では、次の図は、ツールテーブルに入力されたツール位置がどのように見えるかを示しています。

ツール位置1、2、3、4



ツール位置5、6、7、8



3.10.4 ツールタッチオフ

AXIS で旋盤モードで実行している場合、タッチオフウィンドウを使用してツールテーブルで X と Z を設定できます。 ツールタレットを使用している場合は、通常、タレットのセットアップ時に[フィクスチャへのタッチオフ]が選択されています。 材料 Z をゼロに設定する場合、選択した材料にタッチオフします。 ツールに使用される G コードの詳細については、M6、Tn、および G43 を参照してください。 Axis のツールタッチオフオプションの詳細については、ツールタッチオフを参照してください。

1.1.1.1 X タッチオフ

各工具の X 軸オフセットは、通常、スピンドルの中心線からのオフセットです。

1 つの方法は、通常の旋削工具を使用して、ストックを既知の直径まで削ることです。 ツールタッチオフウィンドウを使用して、そのツールの測定された直径（または半径モードの場合は半径）を入力します。 次に、レイアウト液またはマーカーを使用してパーツをコーティングし、染料に触れるまで各ツールを持ち上げ、ツールのタッチオフを使用して使用するパーツの直径に X オフセットを設定します。 コントロールポイントが正しくなるように、コーナー象限のすべてのツールのノーズ半径がツールテーブルで適切に設定されていることを確認してください。 ツールタッチオフは自動的に G43 を追加するため、現在のツールが現在のオフセットになります。

一般的なセッションは次のとおりです。

1. ホームされていない場合は、各軸をホームします。
2. 現在の工具を TnM6 G43 で設定します。ここで、n は工具番号です。
3. 手動制御ウィンドウで X 軸を選択します
4. X を既知の位置に移動するか、テストカットを行って直径を測定します。
5. [タッチオフ]を選択し、[ツールテーブル]を選択して、位置または直径を入力します。
6. 同じ手順で Z 軸を修正します。

注：半径モードの場合は、直径ではなく半径を入力する必要があります。

3.10.4.1 Z タッチオフ

Z オフセットには 2 つの要素があるため、Z 軸オフセットは最初は少し混乱する可能性があります。 工具テーブルオフセットと機械座標オフセットがあります。 まず、ツールテーブルのオフセットを確認します。 1 つの方法は、旋盤の固定点を使用し、この点からすべての工具の Z オフセットを設定することです。 スピンドルノーズまたはチャック面を使用するものもあります。 これにより、すべてのツールをリセットしなくても、新しいツールに変更してその Z オフセットを設定することができます。

一般的なセッションは次のとおりです。

1. ホームされていない場合は、各軸をホームします。
2. 現在の座標系でオフセットが有効になっていないことを確認してください。
3. 現在の工具を TnM6 G43 で設定します。ここで、n は工具番号です。

4. 手動制御ウィンドウでZ軸を選択します。

5. ツールを操縦翼面に近づけます。 円柱を使用して、円柱がツールと操縦翼面の間をちょうど通過するまで、Zを操縦翼面から離します。

6. [タッチオフ]を選択し、[ツールテーブル]を選択して、位置を0.0に設定します。

7. 同じシリンダーを使用して、ツールごとに繰り返します。

これで、すべてのツールが標準位置から同じ距離だけオフセットされます。 ドリルビットのような工具を変更した場合は、上記を繰り返すと、Zオフセット用の残りの工具と同期します。 一部のツールでは、タッチオフポイントからコントロールポイントを決定するために少し暗号化が必要になる場合があります。 たとえば、0.125 "幅のパーティングツールがあり、左側をオフにタッチし、右側をZ0にしたい場合は、タッチオフウィンドウに0.125"と入力します。

3.10.4.2 Zマシンオフセット

すべての工具のZオフセットが工具テーブルに入力されると、任意の工具を使用して、機械座標系を使用して機械オフセットを設定できます。

一般的なセッションは次のとおりです。

1. ホームされていない場合は、各軸をホームします。

2. 現在の工具を「TnM6」で設定します。「n」は工具番号です。

3. G43を発行して、現在の工具オフセットが有効になるようにします。

4. 工具をワークピースに持ってきて、機械のZオフセットを設定します。

機械座標系オフセットを設定するときに現在のツールにG43を設定するのを忘れると、ツールがプログラムで使用されるときにツールオフセットが現在のオフセットに追加されるため、期待どおりの結果が得られません。

3.10.5 スピンドル同期モーション

スピンドル同期モーションには、1回転あたり1つのインデックスパルスでスピンドルに接続された直交エンコーダが必要です。 詳細については、モーションのマニュアルページとスピンドル制御の例を参照してください。

ねじ山G76ねじ山サイクルは、めねじとおねじの両方に使用されます。 詳細については、G76セクションを参照してください。

一定の表面速度CSSまたは一定の表面速度は、ツールXオフセットによって変更されたマシンXの原点を使用して、RPMでスピンドル速度を計算します。 CSSはツールオフセットの変更を追跡します。 Xマシンの原点は、参照ツール（オフセットがゼロのツール）が回転の中心にあるときでなければなりません。 詳細については、G96セクションを参照してください。

1 回転あたりの送り 1 回転あたりの送りは、1 回転あたり F 量だけ Z 軸を移動します。 これはスレッディング用ではありません。スレッディングには G76 を使用してください。 詳細については、G95 セクションを参照してください。

3.10.6 円弧

円弧の計算は、旋盤の半径と直径のモード、および機械の座標系の方向を考慮せずに、十分に困難な場合があります。 以下は、センターフォーマットアークに適用されます。 旋盤では、ユーザーインターフェース Axis で、旋盤モードの場合でもデフォルトは G17 であるため、プリアンブルに G18 を含める必要があります。 G18 XZ 平面の円弧は、I（X 軸）および K（Z 軸）オフセットを使用します。

1.1.1.1 円弧と旋盤の設計

典型的な旋盤は、オペレーターの左側にスピンドルがあり、スピンドルの中心線のオペレーター側に工具があります。 これは通常、架空の Y 軸（+）が床を指すように設定されます。

このタイプのセットアップでは、次のことが当てはまります。

- Z 軸（+）は、スピンドルから離れた右側を指します。
- X 軸（+）はオペレーターを指し、スピンドルのオペレーター側にある場合、X 値は正です。

裏側に工具を備えた旋盤の中には、想像上の Y 軸（+）が上を向いているものがあります。

G2 / G3 円弧の方向は、回転する軸に基づいています。 旋盤の場合、それは虚数の Y 軸です。 Y 軸（+）が床を指している場合、正しい方向に向かっているように見える円弧を探す必要があります。 したがって、上から見ると、G2 / G3 を逆にして、円弧が正しい方向に進んでいるように見えます。

3.10.6.1 半径と直径モード

半径モードで円弧を計算するときは、旋盤に適用される回転方向を覚えておくだけで済みます。

直径モードで円弧を計算する場合、G7 直径モードを使用している場合でも、X は直径であり、X オフセット（I）は半径です。

3.10.7 ツールパス

1.1.1.1 コントロールポイント

ツールのコントロールポイントは、プログラムされたパスに従います。 制御点は、X 軸と Z 軸に平行で、ツールの先端の直径に接する線の交点です。これは、そのツールの X 軸と Z 軸をタッチオフしたときに定義されます。 真っ直ぐな側面の部品を回転または向けるとき、切削経路と工具エッジは同じ経路をたどります。 半径と角度を回転させると、カッターコンプが有効になっていない限り、ツールチップのエッジはプログラムされたパスをたどりません。 次の図では、想定どおりにコントロールポイントがツールエッジに従わないことがわかります。

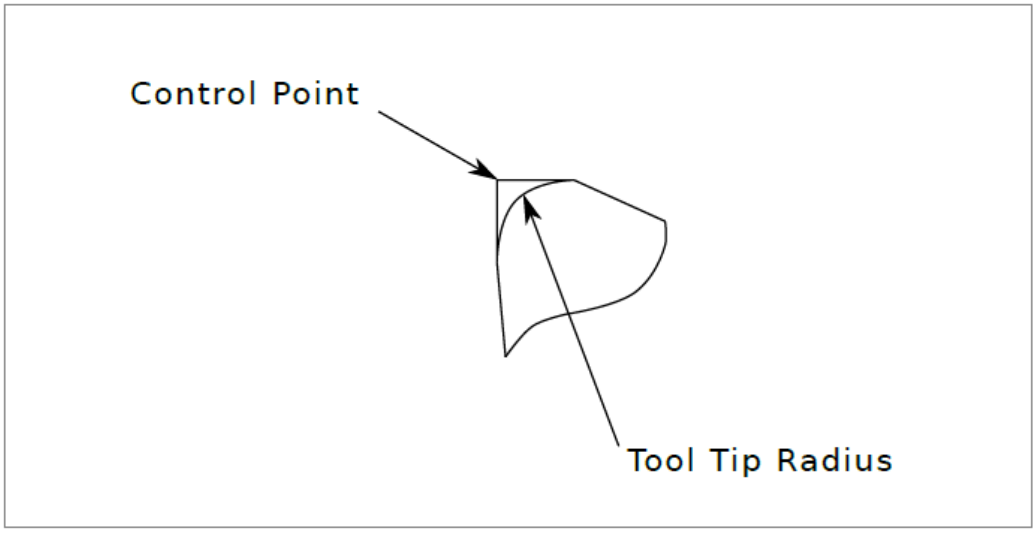


図 3-31

3.10.7.1 カッターコンプなしの切削角度

ここで、カッターコンプなしでランプをプログラムすると想像してください。 プログラムされたパスを次の図に示します。 図からわかるように、X または Z 方向にのみ移動している限り、プログラムされたパスと目的のカットパスは同じです。

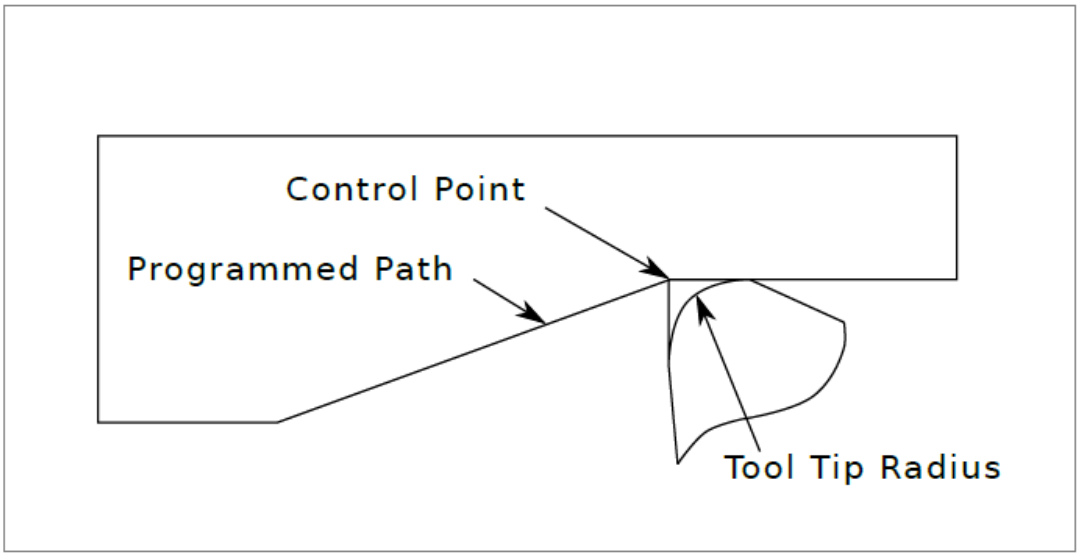


図 3-32

次の図に示すように、制御点がプログラムされたパスに沿って進むと、実際のカッターエッジはプログラムされたパスをたどりません。 これを解決するには、カッターコンプとチップ半径を補正するためにプログラムされたパスを調整する 2 つの方法があります。

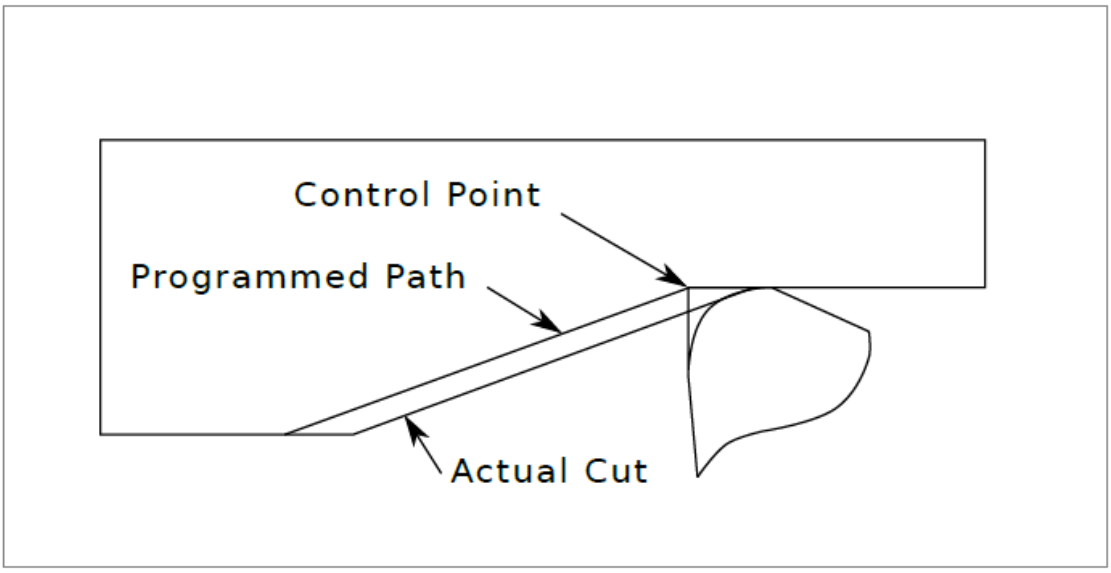


図 3-33

上記の例では、ランプのプログラムされたパスをツールチップの半径の左側に移動することにより、プログラムされたパスを調整して目的の実際のパスを与えるのは簡単な演習です。

3.10.7.2 半径を切る

この例では、カッターコンプなしのラジアスカット中に何が起こるかを調べます。 次の図では、ツールがパーツの外径を回転させているのがわかります。 ツールのコントロールポイントはプログラムされたパスをたどり、ツールはパーツの外径に接触しています。

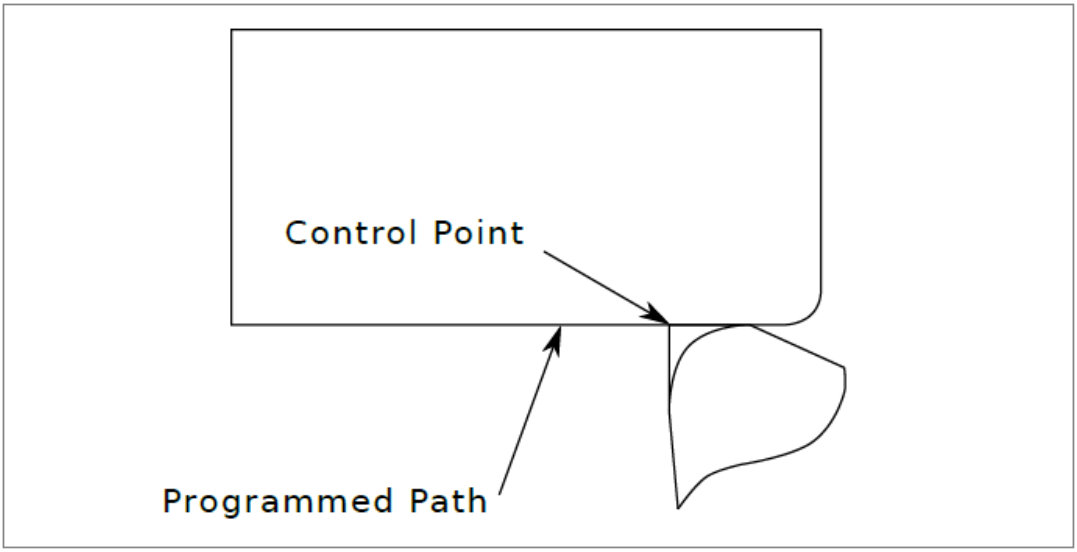


図 3-34

この次の図では、ツールがパーツの端に近づくと、コントロールポイントはまだパスをたどっていますが、ツールチップはパーツを離れて空気を切断していることがわかります。 また、半径がプログラムされていても、パーツは実際には正方形のコーナーになってしまうことがわかります。

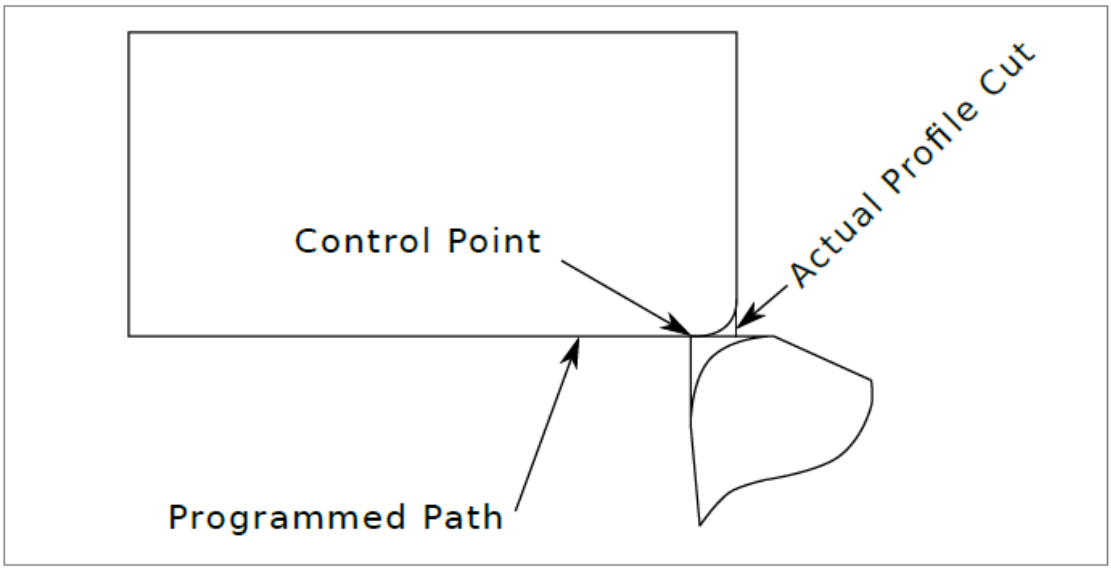


図 3-35

これで、コントロールポイントがプログラムされた半径に従うので、ツールチップがパーツを離れ、空気をカットしていることがわかります。

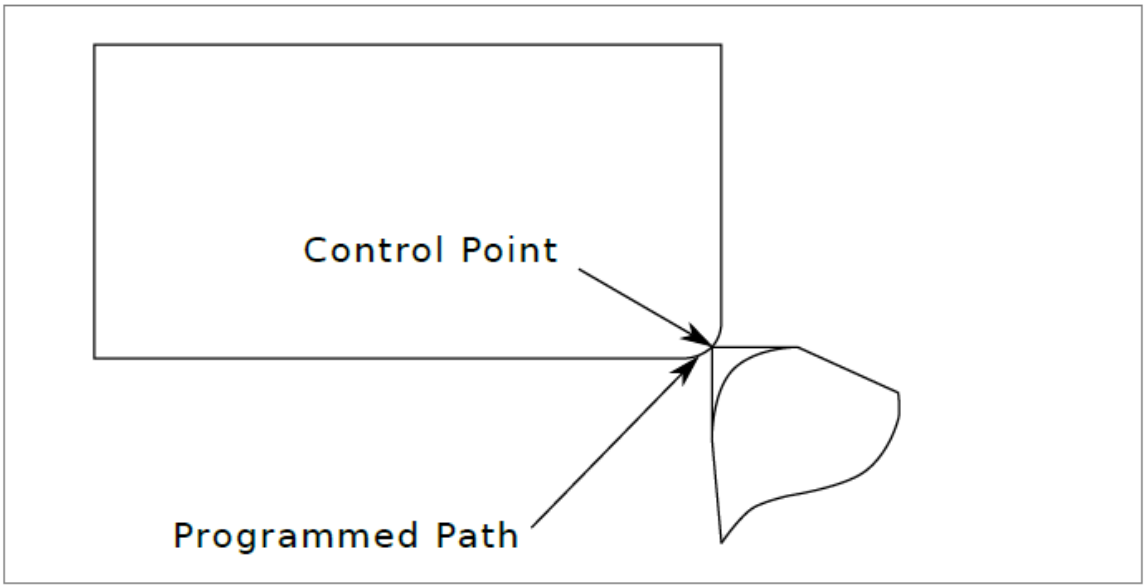


図 3-36

最終的な図では、ツールチップが面の切断を終了しますが、適切な半径ではなく正方形のコーナーを残していることがわかります。 パーツの中心で終了するようにカットをプログラムすると、ツールの半径から少量の材料が残ることにも注意してください。 パーツの中心へのフェースカットを終了するには、少なくともツールのノーズ半径の中心を通過するようにツールをプログラムする必要があります。

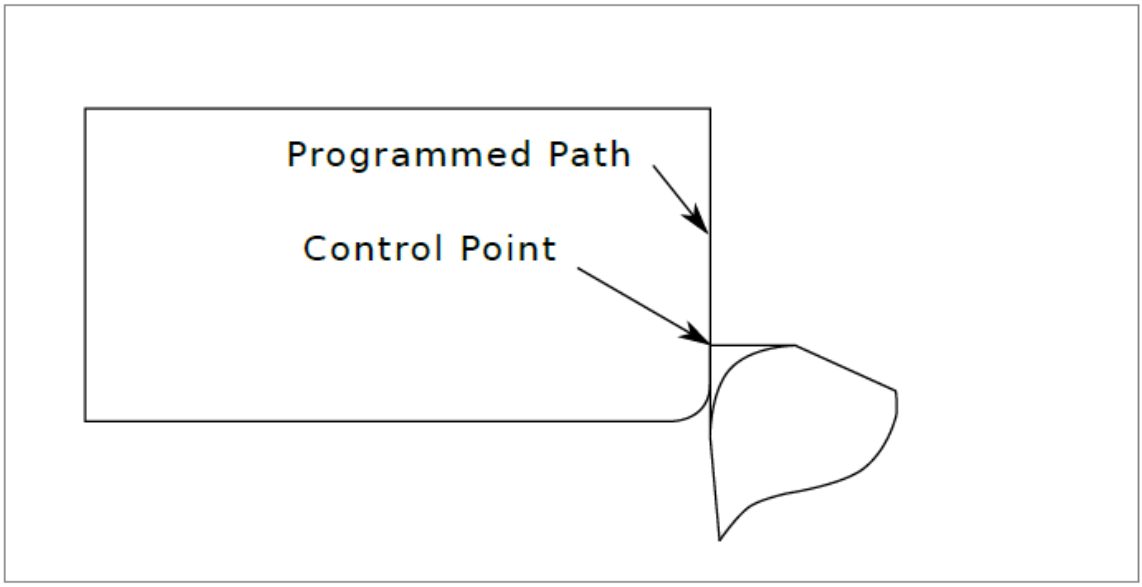


図 3-37

3.10.7.3 カッターコンプの使用

旋盤でカッターコンプを使用する場合、工具先端の半径を丸いカッターの半径と考えてください。 カッターコンプを使用する場合、パスは次の行に進まない丸い工具に十分な大きさである必要があります。 旋盤で直線をカットするときは、カッターコンプを使用したくない場合があります。 たとえば、タイトフィットのボーリングバーで穴を開けると、出口の移動を行うのに十分なスペースがない場合があります。 正しい結果を得るには、カッターコンプアークへのエントリの移動が重要です。

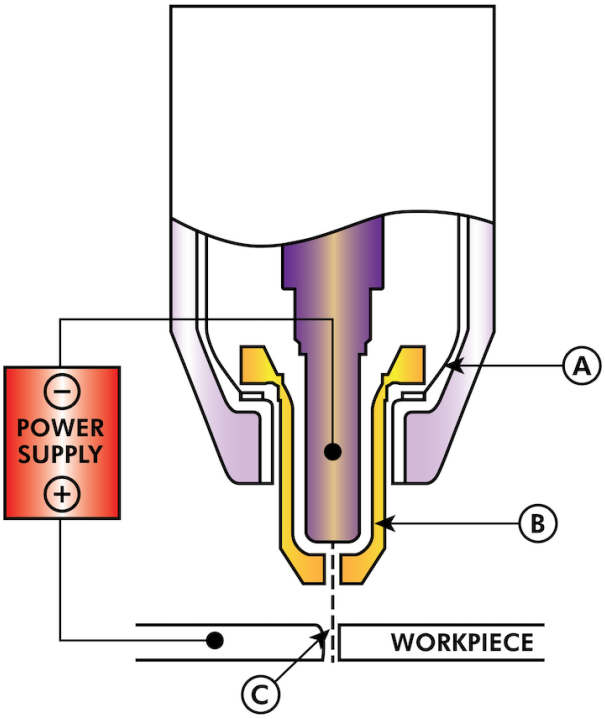
3.11 LinuxCNC ユーザー向けのプラズマ切断プライマー

3.11.1 プラズマとは？

プラズマは第 4 の物質の状態であり、非常に高温に加熱されてイオン化されて導電性になるイオン化ガスです。 プラズマアークの切断およびガウジングプロセスでは、このプラズマを使用して電気アークをワークピースに転送します。 切断または除去される金属は、アークの熱によって溶けてから吹き飛ばされます。 プラズマアーク切断の目的は材料の分離ですが、プラズマアークガウジングは、制御された深さと幅まで金属を除去するために使用されます。

プラズマトーチは、自動車のスパークプラグとデザインが似ています。 それらは、中央の絶縁体によって分離された負と正のセクションで構成されています。 トーチの内側では、パイロットアークは負に帯電した電極と正に帯電したチップの間のギャップから始まります。 パイロットアークがプラズマガスをイオン化すると、過熱されたガス柱がトーチ先端の小さなオリフィスを通って流れます。このオリフィスは、切断される金属に焦点を合わせています。

プラズマ切断トーチでは、冷たいガスがゾーン B に入り、そこで電極とトーチ先端の間のパイロットアークがガスを加熱してイオン化します。 次に、メインの切断アークがゾーン C のプラズマガスのカラムを介してワークピースに移動します。プラズマガスと電気アークを小さなオリフィスに通すことにより、トーチは小さな領域に高濃度の熱を供給します。 硬くて収縮したプラズマアークがゾーン C に示されています。図に示すように、プラズマ切断には直流（DC）ストレート極性が使用されます。 ゾーン A は、トーチを冷却する二次ガスを送ります。 このガスはまた、高速プラズマガスが溶融金属をカットから吹き飛ばすのを助け、スラグのない高速カットを可能にします。



TYPICAL TORCH HEAD DETAIL

3.11.2 アークの初期化

CNC 操作用に設計されたプラズマカッターのアーク初期化には、主に 2 つの方法があります。 一部のマシンでは他の方法が使用されますが（材料との物理的接触が必要なスクラッチスタートなど）、CNC アプリケーションには適していません。

1.1.1.1 高周波スタート

このスタートタイプは広く採用されており、最も長く使用されています。 古いテクノロジーですが、うまく機能し、すぐに起動します。 しかし、空気をイオン化するために生成される必要がある高周波高電圧電力のために、それはいくつかの欠点を持っています。 多くの場合、周囲の電子回路に干渉し、コンポーネントに損傷を与える可能性さえあります。 また、パイロットアークを作成するには特別な回路が必要です。 安価なモデルにはパイロットアークがなく、作業を開始するには消耗品に触れる必要があります。 HF 回路を使用すると、メンテナンスの問題が増える可能性があります。通常、調整可能なポイントがあり、時々清掃して再調整する必要があるためです。

3.11.2.1 ブローバックスタート

このスタートタイプは、カッターに供給される空気圧を使用して、トーチヘッド内の小さなピストンまたはカートリッジを押し戻し、消耗品の内面間に小さな火花を発生させ、空気をイオン化し、小さなプラズマ炎を生成します。 これはまた、金属と接触しているかどうかに関係なく、プラズマ炎を維持する「パイロットアーク」を作成します。 これは非常に優れたスタートタイプであり、現在いくつかのメーカーで使用されています。 利点は、必要な回路がいくらか少なく、信頼性が高く、電気ノイズの発生がはるかに少ないことです。

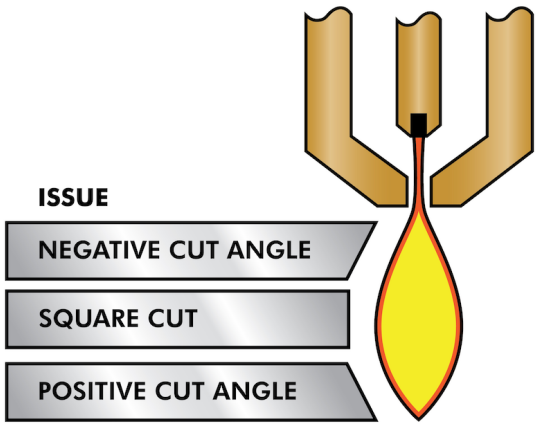
エントリーレベルのエアプラズマ CNC システムの場合、電子機器や標準 PC との電気的干渉を最小限に抑えるためにブローバックスタイルが非常に好まれますが、200 アンペア以上の大型マシンでは高周波始動が依然として最高です。 これらには産業レベルの PC と電子機器が必要であり、商業メーカーでさえ、設計で電気的ノイズを考慮に入れていないため、障害の問題がありました。

3.11.3 CNC プラズマ

CNC マシンでのプラズマ操作は、フライス盤や旋盤と比較して非常にユニークであり、少し孤立したプロセスです。 プラズマアークからの材料の不均一な加熱により、シートが曲がったり座屈したりします。 ほとんどの金属シートは、ミルから出たり、非常に均一または平坦な状態でプレスされたりすることはありません。 厚いシート（30mm 以上）は、50mm から 100mm まで平面から外れる可能性があります。 他のほとんどの CNCG コード操作は、既知の参照または既知のサイズと形状のストックから開始され、G コードは余分な部分を粗くし、最後に完成品をカットするように書き込まれます。 プラズマでは、シートの状態が不明なため、材料のこれらの変動に対応する G コードを生成できません。

プラズマアークは楕円形であり、面取りされたエッジを最小限に抑えるために切断高さを制御する必要があります。 トーチが高すぎたり低すぎたりすると、エッジが過度に面取りされる可能性があります。 トーチが表面に対して垂直に保持されることも重要です。

トーチから作業距離までの距離がエッジベベルに影響を与える可能性があります



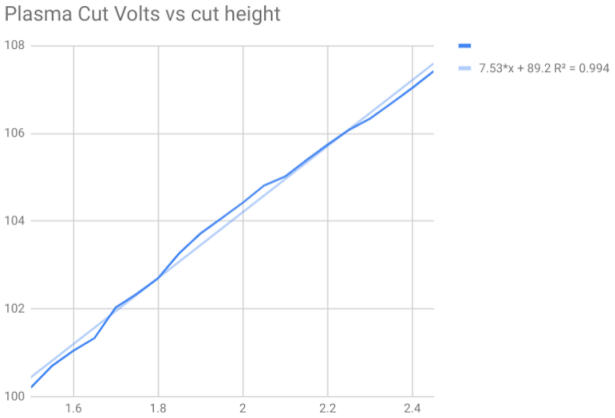
負の切断角度：トーチが低すぎる、トーチを作業距離まで増やします。

正の切断角度：トーチが高すぎます。トーチから作業距離までを短くしてください。

Note

許容範囲内である限り、切断角度のわずかな変動は正常である可能性があります。

このような敵対的で絶えず変化する環境で切断高さを正確に制御する能力は、非常に困難な課題です。 幸い、このグラフが示すように、トーチの高さ（アーク長）とアーク電圧の間には非常に直線的な関係があります。



このグラフは、さまざまなカット高さでの約 16,000 の読み取り値のサンプルから作成され、回帰分析は 99.4% の信頼度で 1mm あたり 7.53 ボルトを示しています。 この特定の例では、このサンプルは Linuxcnc によって制御されている Everlast50 アンペアマシンから取得されました。

トーチ電圧は、カット高さを調整するために使用する理想的なプロセス制御変数になります。 簡単にするために、電圧が 1mm あたり 10 ボルト変化すると仮定しましょう。 これは、0.1mm（0.04 インチ）あたり 1 ボルトに言い換えることができます。 主要なプラズママシンメーカー（Hypertherm、Thermal Dynamics、ESAB など）は、推奨される切断高さとこの高さでの推定アーク電圧、およびいくつかの追加データを指定する切断チャートを作成しています。 したがって、アーク電圧がメーカーの仕様より 1 ボルト高い場合、コントローラーはトーチを 0.1 mm（0.04 インチ）下げるだけで、目的の切断高さに戻ることができます。 トーチ高さ制御ユニット（THC）は、従来、このプロセスを管理するために使用されていました。

3.11.4 CNC 操作用のプラズママシンの選択

現在、市場には多数のプラズママシンがあり、そのすべてが CNC での使用に適しているわけではありません。 CNC プラズマ切断は複雑な操作であるため、インテグレーターは適切なプラズママシンを選択することをお勧めします。 これを怠ると、多くの人が必須機能と見なす機能の欠如を回避しようとして、何時間にもわたって無駄なトラブルシューティングが発生する可能性があります。

ルールが適用される理由を完全に理解していればルールは破られますが、新しいプラズマテーブルビルダーは次の機能を備えたマシンを選択する必要があると考えています。

- ブローバックスタートにより電気ノイズを最小限に抑え、建設を簡素化
- マシントーチが好まれますが、多くはハンドトーチを使用しています。
- オーミックセンシングを可能にする完全にシールドされたトーチチップ

予算がある場合は、ハイエンドのマシンが以下を提供します。

- メーカーが提供するカットチャートは、カットパラメータのキャリブレーションにかかる時間と材料の無駄を節約します。
- ArcOK のドライコンタクト
- ArcOn スイッチの端子
- 生のアーク電圧または分割アーク電圧出力
- Hypertherm プラズマカッターを使用していて、Linuxcnc コンソールから制御したい場合は、オプションで RS485 インターフェース。
- より高いデューティサイクル

最近では、これらの機能の一部を含む別のクラスのマシンが約 550 米ドルで利用できるようになりました。 一例として、Amazon で入手可能な Herocut55i がありますが、ユーザーからのフィードバックはまだありません。このマシンは、ブローバックトーチ、ArcOK 出力、トーチ開始接点、および生のアーク電圧を備えています。

3.11.5 トーチ高さ制御の種類

ほとんどの THC ユニットは外部デバイスであり、多くはかなり大雑把な「ビットバン」調整方法を備えています。これらは、LinuxCNC コントローラーに 2 つの信号を返します。1 つは、Z 軸が上に移動する必要がある場合にオンになり、もう 1 つは、Z 軸が下に移動する必要がある場合にオンになります。トーチが正しい高さにある場合、どちらの信号も真ではありません。人気のあるプロマ 150THC は、このタイプの THC の一例です。Linuxcnc THCUD コンポーネントは、このタイプの THC で動作するように設計されています。

Mesa THCAD 電圧から周波数へのインターフェースのリリースにより、LinuxCNC はエンコーダ入力を介して実際のトーチ電圧をデコードすることができました。これにより、LinuxCNC は Z 軸を制御し、外部ハードウェアを排除することができました。THCAD を利用した初期の実装では、「ビットバン」アプローチが複製されました。Linuxcnc THC コンポーネントは、このアプローチの例です。

Hypertherm の JimColt は、最高の THC コントローラーが CNC コントローラー自体に完全に統合されたと記録しています。もちろん、彼は Hypertherm、Esab、Thermal Dynamics、およびオーストラリアの Advanced Robotic Technology などの他の製品によって製造されたハイエンドシステムについて言及していましたが、オープンソースがこのアプローチを使用してハイエンドシステムに匹敵するシステムを製造できるとは夢にも思いませんでした。

Linuxcnc V2.8 に外部オフセットを含めることで、LinuxCNC のプラズマ制御をまったく新しいレベルに引き上げることができました。外部オフセットとは、モーションコントローラの外部の軸コマンド位置にオフセットを適用する機能を指します。これは、選択したプロセス制御方法に基づいてトーチの高さをリアルタイムで調整する方法として、プラズマ THC 制御に最適です。いくつかの実験的なビルドに続いて、PlasmaC 構成は LinuxCNC2.8 に組み込まれました。これは非常に野心的なプロジェクトであり、世界中の多くの人々が機能セットのテストと改善に携わってきました。PlasmaC は、その設計目標が、THCAD またはその他の電圧センサーを

介して LinuxCNC で電圧が利用可能になった場合に、単純なビットバンから高度なトーチ電圧制御までを含むすべての THC をサポートすることであったという点で独特です。さらに、PlasmaC は、追加の G コードサブルーチンを必要としないスタンドアロンシステムとして設計されており、ユーザーはシステムに保存され、ドロップダウンでアクセスできる独自のカットチャートを定義できます。

3.11.6 アーク OK 信号

CNC インターフェースを備えたプラズママシンには、有効なアークが確立され、これらの接点の両側が CNC インターフェースのピンに買い取られると閉じる、一連のドライ接点（リレーなど）が含まれています。プラズマテーブルビルダーは、これらのピンの一方をフィールド電源に接続し、もう一方を入力ピンに接続する必要があります。これにより、CNC コントローラーは、有効なアークが確立されたときと、アークが予期せず失われたときを知ることができます。入力が Mesa カードなどの高インピーダンス回路の場合、ここに潜在的なトラップがあります。ドライ接点が単純なリレーの場合、リレーを通過する電流が最小電流仕様よりも少ない可能性が高くなります。これらの条件下では、リレー接点は酸化物の蓄積に悩まされる可能性があり、時間の経過とともに断続的な接点動作が発生する可能性があります。これを防ぐために、コントローラの入力ピンにプルダウン抵抗を取り付ける必要があります。最小電流がリレーを通過し、回路内の電力を処理するのに十分なワット数になるように、この抵抗が選択されていることを確認するように注意する必要があります。最後に、抵抗器は、発生した熱が動作中に何も損傷しないように取り付ける必要があります。

ArcOK シグナルがある場合は、潜在的なビルドの問題を排除するために、合成されたシグナルに加えて使用することをお勧めします。 外部 THC または Plasmac のモード 0 から利用できる合成信号は、プラズマインバーターの ArcOK 回路を完全に置き換えることはできません。 合成された ArcOK 信号からプラズマインバーターとの構成ミスまたは非互換性が発生した場合、いくつかのビルドの問題が観察されています。 ただし、概して、正しく構成された合成 ArcOK 信号は問題ありません。

シンプルで効果的な arcOK 信号は、シンプルなリードリレーで実現できます。 プラズマ切断機の太いケーブル（材料クランプケーブルなど）の 1 つを 3 ターン巻き付けます。 保護のためにリレーを古いペンチューブに入れ、リレーの一方の側をフィールド電源に接続し、もう一方の端を ArcOK 入力ピンに接続します。

3.11.7 初期高さ検知

切削高さは非常に重要なシステムパラメータであり、材料表面は本質的に不均一であるため、Z 軸メカニズムには材料表面を検知する方法が必要です。 これを実現できる方法は 3 つあります。 モータートルクの増加を検出するための電流検出、「フロート」スイッチ、およびトーチシールドが材料に接触すると閉じる電気的または「オーミック」検出回路。 電流検出は DIY テーブルでは実行可能な手法ではありませんが、フロートスイッチとオーミック検出について以下で説明します。

1.1.1.1 フロートスイッチ

トーチは、トーチの先端が材料の表面に接触してスイッチまたはセンサーをトリガーすると上に移動できるスライドステージに取り付けられています。 多くの場合、これは G38 コマンドを使用した G コード制御の下で実現されます。 この場合、最初のプロービングの後、プローブ信号がより遅い速度で失われるまで、表面から離れてプロービングすることをお勧めします。 また、スイッチのヒステリシスが考慮されていることを確認してください。

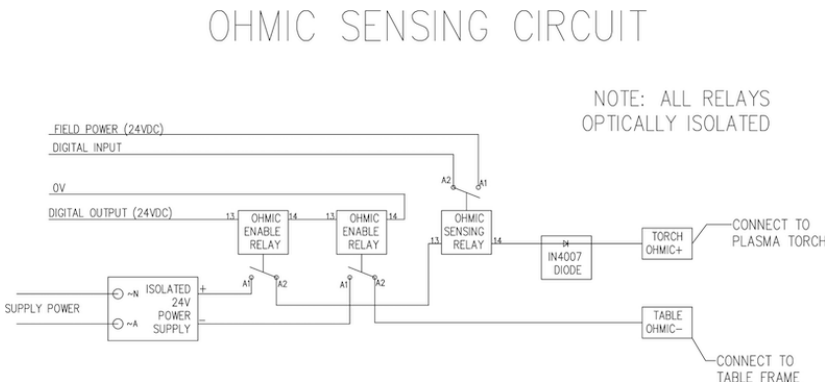
使用するプロービング方法に関係なく、クラッシュによるトーチへの損傷を避けるために、フォールバックまたはセカンダリ信号が発生するようにフロートスイッチを実装することを強くお勧めします。

3.11.7.1 オーミックセンシング

オーミックセンシングは、トーチとスイッチとして機能する材料との接触に依存して、CNC コントローラーによってセンシングされる電気信号をアクティブにします。 材料がきれいであれば、これは、材料表面のたわみを引き起こす可能性のあるフロートスイッチよりもはるかに正確に材料を検知する方法になります。 このオーミックセンシング回路は非常に過酷な環境で動作しているため、CNC 電子機器とオペレーターの両方の安全を確保するために多くのフェイルセーフを実装する必要があります。 プラズマ切断では、材料に取り付けられたアースクランプは正であり、トーチは負です。 次のことをお勧めします。

1. オーミックセンシングは、トーチが切断アークを伝達するトーチ先端から分離されたシールドを備えている場合にのみ実装されます。
2. オーミック回路は、完全に独立した絶縁電源を使用して、光絶縁リレーをアクティブにし、プロービング信号を CNC コントローラーに送信できるようにします。
3. 回路のプラス側はトーチにある必要があります
4. プロービングが行われるまで、回路の両側を光絶縁リレーで絶縁する必要があります
5. ブロッキングダイオードは、アーク電圧がオーム検知回路に入るのを防ぐために使用されます。

以下は、動作することが証明されており、LinuxcncPlasmaC 構成と互換性のある回路例です。



3.11.8 MESATHCAD-5 によるハイパーセンシング

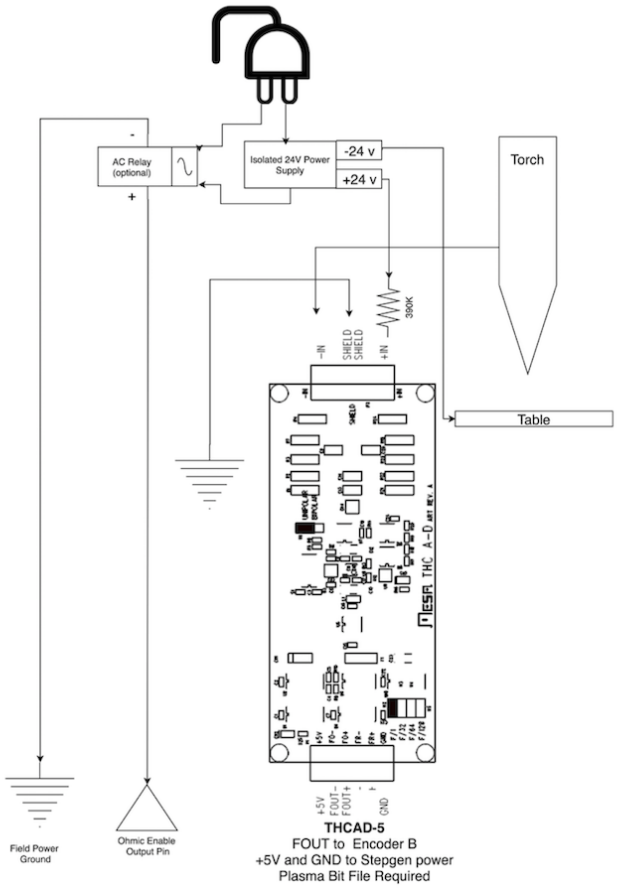
リレーとダイオードを排除するより洗練された材料検知方法は、別の THCAD-5 を使用して、絶縁された電源からの材料検知回路電圧を監視することです。 これが持つ利点は、THCAD が敵対的なプラズマ電気環境向けに設計されており、ロジック側を高電圧側から完全かつ安全に分離することです。

このメソッドを実装するには、2 番目のエンコーダ入力が必要です。

メサカードを使用している場合は、エンコーダ B ピンとエンコーダインデックスピンに 2 つの追加のエンコーダ A 入力を提供するために異なるファームウェアを使用できます。 このファームウェアは、製品ページの MesaWeb サイトから 7i76e および 7i96 ボード用にダウンロードできます。

THCAD は、接触圧力が増加するにつれて回路電圧が上昇するのを確認するのに十分な感度があります。 Linuxcnc に含まれている ohmic.comp コンポーネントは、検出電圧を監視し、電圧しきい値を設定できます。このしきい値を超えると、接触が行われ、出力が有効になります。 電圧を監視することにより、より低い「遮断回路」しきい値を設定して、強力なスイッチヒステリシスを組み込むことができます。 これにより、誤ったトリガーが最小限に抑えられます。 私たちのテストでは、この方法を使用したマテリアルセンシングは、より感度が高く、堅牢であるだけでなく、配線の実装も簡単であることがわかりました。 もう 1 つの利点は、物理 I / O ピンの代わりにソフトウェア出力を使用することです。これにより、ピンが解放され、他の目的に使用できるようになります。 この利点は、I / O ピンが制限されている Mesa7i96 を最大限に活用するのに役立ちます。

次の回路図は、ハイパーセンシング回路を実装する方法を示しています。



15W Mean WellHDR-15 ウルトラスリム DIN レール電源 24VDIN レールベースの絶縁型電源を使用しました。これは、端子に印加される可能性のあるアーク電圧に耐える二重絶縁絶縁クラス II デバイスです。

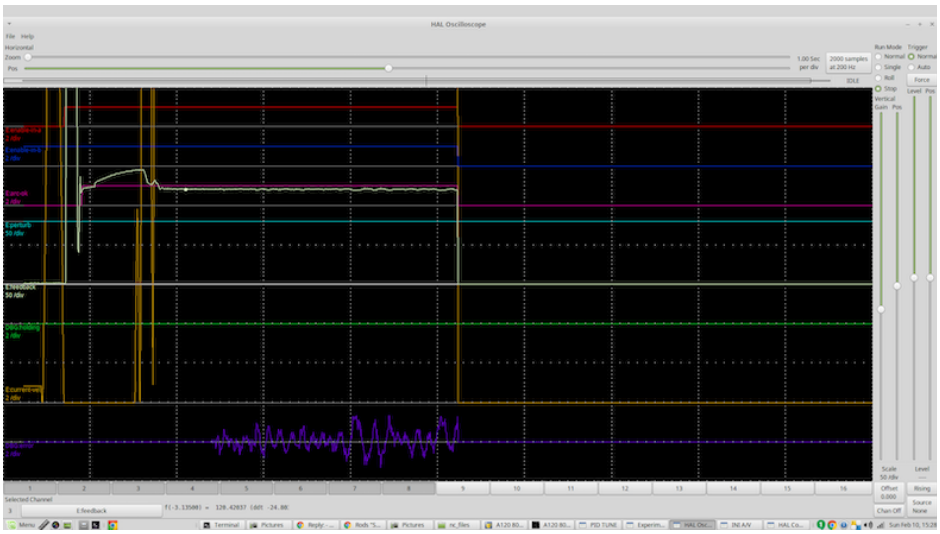
3.11.9 ハイパーセンシングの HAL コードの例

次の HAL コードを plasmac_connections.hal に貼り付けて、7i76e のエンコーダー 2 でオーミックセンシングを有効にすることができます。 正しいビットファイルをインストールし、THCAD を IDX +および IDX-に接続します。 THCAD-5 と一致するように、必ずキャリブレーション設定を変更してください。

```
# --- Load the Component ---
loadrt ohmic names=ohmicsense
addf ohmicsense servo-thread
# --- 7i76e ENCODER 2 SETUP FOR OHMIC SENSING---
setp hm2_7i76e.0.encoder.02.scale -1
setp hm2_7i76e.0.encoder.02.counter-mode 1
# --- Configure the component ---
setp ohmicsense.thcad-0-volt-freq 140200
setp ohmicsense.thcad-max-volt-freq 988300
setp ohmicsense.thcad-divide 32
setp ohmicsense.thcad-fullscale 5
setp ohmicsense.volt-divider 4.9
setp ohmicsense.ohmic-threshold 22.0
setp ohmicsense.ohmic-low 1.0
net ohmic-vel ohmicsense.velocity-in <= hm2_7i76e.0.encoder.02.velocity
# --- Replace Plasmac's Ohmic sensing signal ---
unlinkp debounce.0.2.in
net ohmic-true ohmicsense.ohmic-on => debounce.0.2.in
net plasmac:ohmic-enable => ohmicsense.is-probing
```

3.11.10 THC 遅延

アークが確立されると、アーク電圧は大幅にピークに達し、その後、切断高さで安定した電圧に落ち着きます。下の画像の緑色の線で示されているように。



トーチ電圧を自動サンプリングして THC 制御を開始する前に、プラズマコントローラーが「待機」することが重要です。 有効にするのが早すぎると、電圧が目的のカットボルトを超え、トーチが押し下げられて、知覚される高さ超過状態に対処しようとします。

私たちのテストでは、これは機械と材料によって 0.5 秒から 1.5 秒まで異なります。 したがって、有効な arcOK 信号を受信してから THC 制御を有効にするまでの 1.5 秒の遅延は、安全な初期設定です。 特定の材料についてこれを短縮したい場合、LinuxCNC の Halscope を使用すると、トーチ電圧をプロットし、使用される最短の安全遅延について情報に基づいた決定を行うことができます。

Note

この遅延の終了時にカット速度が目的のカット速度に近くない場合、コントローラーは THC を有効にする前にこれが達成されるまで待機する必要があります。

3.11.11 トーチ電圧サンプリング

多くの人（ライターを含む）は、メーカーのカットチャートに依存して希望のトーチ電圧を設定するのではなく、THC が有効になっているときに電圧をサンプリングし、それを設定値として使用することを好みます。

3.11.12 トーチブレイクアウェイ

材料または転倒した切断部品との衝突の場合にトーチが「脱落」または脱落することを可能にするメカニズムを提供することをお勧めします。 CNC コントローラーがこれが発生したかどうかを検出し、実行中のプログラムを一時停止できるように、センサーをインストールする必要があります。 通常、ブレークアウェイは、トーチを Z 軸ステージに固定するために磁石を使用して実装されます。

3.11.13 コーナーロック/ベロシティアンチダイブ

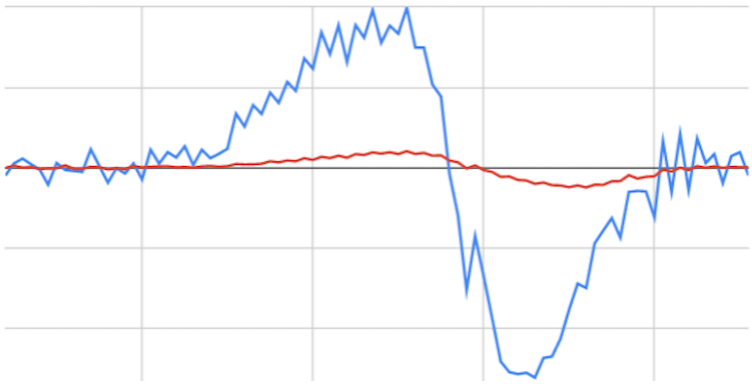
Linuxcnc 軌道プランナーは、速度と加速度のコマンドを物理法則に従うモーションに変換する役割を果たします。 たとえば、コーナーをネゴシエートするときにモーションが遅くなります。 これはフライス盤やルーターでは問題になりませんが、動きが遅くなるとアーク電圧が上昇するため、プラズマ切断に特に問題があります。 これにより、THC がトーチを押し下げます。 LinuxCNC モーションコントローラーに組み込まれた THC コントロールの大きな利点の 1 つは、何が起こっているかを常に認識していることです。 したがって、現在の速度（motion.current-velocity）を監視し、設定されたしきい値（たとえば、目的の送り速度より 10％低い）を下回った場合に THC 操作を保持することは簡単なことです。

3.11.14 ボイド/カーフクロッシング

プラズマトーチが切断中にボイドを通過すると、アーク電圧が急速に上昇し、THC は激しい下向きの動きで応答し、トーチを材料に押しつぶして損傷を与える可能性があります。 これは、検出と処理が難しい状況です。 ある程度、それは優れたネスティング技術によって軽減することができますが、それでもスラグが落ちたときに厚

い材料で発生する可能性があります。 これは、LinuxCNC オープンソース運動の中でまだ解決されていない問題の 1 つです。

推奨される手法の 1 つは、トーチボルトの経時変化率（dv / dt）を監視することです。これは、このパラメータが、材料の通常の反りによって発生するパラメータよりも、ボイドを通過するときに桁違いに大きいためです。 次のグラフは、ボイドを横切るときの dv / dt（青色）の低解像度プロットを示しています。 赤い曲線はトーチボルトの移動平均です。



したがって、移動平均を dv / dt と比較し、dv / dt が反りのために予想される通常の範囲を超えたら、THC 操作を停止することができるはずです。 LinuxCNC で機能するソリューションを考え出すには、この領域でさらに作業を行う必要があります。

3.11.15 穴と小さな形状の切断

穴や小さな形状をカットするときは、カットを遅くすることをお勧めします。

John Moore 氏は次のように述べています。「正確な小さな穴を開ける詳細が必要な場合は、Hypertherm の「TrueHole Technology」のセールスシートを調べて、PlasmaSpider も調べてください。ユーザー seanp は、単純な空気プラズマを使用した彼の仕事について広範囲に投稿しています。

直径 37mm から良好な穴を得る一般的に受け入れられている方法。 空気プラズマを使用して最小限のテーパーで材料の厚さまで下げると、次のようになります。

1. 消耗品には推奨切削電流を使用してください。
2. 消耗品には固定（THC なし）の推奨切断高さを使用してください。
3. 消耗品と材料の推奨送り速度の 60%から 70%にカットします。
4. 穴の中心またはその近くでリードを開始します。
5. で垂直リードを使用します。
6. 何が最適かによって、わずかなオーバーバーンまたは早期のトーチオフのいずれかで、リードアウトはありません。

この方法の切り口は通常のストレートカットよりも幅が広いため、正確な穴のサイズを取得するために実験する必要があります。」

この速度低下は、ポストプロセッサで直接送り速度を操作するか、入力として適応送りとアナログピンを使用することで実現できます。 これにより、M67 / M68 を使用して、カットする目的のフィードのパーセンテージを設定できます。

-送り速度を知る

前の議論から、プラズマコントローラがユーザーによって設定された供給速度を知る必要があることは明らかです。 これは、G コードがバッファリングおよび解析された後、フィードレートが LinuxCNC によって保存されないため、LinuxCNC で問題を引き起こします。 これを回避するには、次の 2 つの方法があります。

1. F コマンドを再マップし、M67 / M68 コマンドを介して G コードに設定されたコマンド送り速度を保存します
2. カットチャートをプラズマコントローラに保存し、現在の送り速度を G コードプログラムで照会できるようにします（PlasmaC と同様）。

プラズマ切断に役立つ実験的な Linuxcnc ブランチの 1 つは、状態タグブランチでした。 これにより、すべてのアクティブなモーションコマンドの現在のフィードと速度レートを含むモーションで使用できる「タグ」が追加されます。 マージされ、LinuxCNCv2.9 になります

3.11.16 プラズマコントローラの I / O ピン

プラズマ切断機には、いくつかの追加のピンが必要です。 LinuxCNC には、どのピンが何を実行するかについての厳格なルールはありません。 この説明では、プラズマインバーターに CNC インターフェースがあり、コントローラーカードにアクティブハイ入力が使用されていると仮定します（例：Mesa7i76e）。

プラズマテーブルは大型の機械である可能性があるため、ジョイントごとに個別の最大/最小リミットスイッチとホーミングスイッチを設置することをお勧めします。 例外は、Z 軸の下限である可能性があります。 ホーミングスイッチがトリガーされると、ジョイントはかなりゆっくりと減速し、最高の精度が得られます。 これは、テーブルサイズに見合ったホーミング速度を使用する場合、最初のトリガーポイントを 50〜100mm オーバーシュートできることを意味します。 共有ホーム/リミットスイッチを使用する場合は、最後の HOME_OFFSET でセンサーをトリガーポイントから移動する必要があります。そうしないと、マシンがホーミングから外れるときにリミットスイッチ障害がトリガーされます。 これは、共有のホーム/リミットスイッチで 50mm 以上の軸移動を失う可能性があることを意味します。 これは、ホームスイッチとリミットスイッチを別々に使用している場合は発生しません。

通常、次のピンが必要です（PlasmaC 構成には推奨される接続が適切でない場合があることに注意してください）。

1.1.1.1 アーク OK（入力）

- 有効なアークが確立されると、インバータはドライ接点を閉じます

- フィールド電源を 1 台のインバータ ArcOK 端子に接続します。
- 他のインバータ OK 端子を入力ピンに接続します。
- 通常、M66 で G コードから使用するために motion.digital- <nn>ピンの 1 つに接続されます

3.11.16.1 トーチオン（出力）

- リレーをトリガーして、インバーターのスイッチのトーチを閉じます
- インバータの端子のトーチをリレー出力端子に接続します
- コイルの片側を出力ピンに接続します
- コイルの反対側をフィールドパワーアースに接続します。
- 機械式リレーを使用する場合は、コイル端子間にフライバックダイオード（IN400x シリーズなど）を接続し、ダイオードのバンドを出力ピンに向けます。
- ソリッドステートリレーを使用する場合は、出力で極性を確認する必要がある場合があります
- 状況によっては、外部リレーの代わりに Mesa カードのオンボードスピンドルリレーを使用できます。
- 通常、spindle.0.on に接続されます

警告

このピンが false の間は、トーチを有効にしないことを強くお勧めします。有効にしないと、estop を押してもトーチが消えません。

3.11.16.2 フロートスイッチ（入力）

- 表面プロービングに使用されます。 トーチが材料に当たったときにトーチが上にスライドした場合にアクティブになるセンサーまたはスイッチ。
- 近接センサー出力を選択した入力ピンに接続します。 メカニカルスイッチを使用する場合。 スイッチの一方の側をフィールド電源に接続し、スイッチのもう一方の側を入力に接続します。
- 通常、motion.probe-input に接続されます

3.11.16.3 オーミックセンサーイネーブル（出力）

- オーミックセンシングの概略図を参照してください。
- 出力ピンを絶縁リレーの一方の側に接続し、もう一方の側をフィールド電源グランドに接続します。
- 非 PlasmaC 構成では、通常は motion.digital-out- <nn>によってトリガーされるため、M62 / M63 / M64 / M65 によって G コードで制御できます。

3.11.16.4 オーミックセンシング（入力）

- 前に示したオーム検知回路図に従うように注意してください。
- トーチシールドが材料に接触すると、絶縁された電源がリレーをトリガーします。

- フィールド電源を一方の出力端子に接続し、もう一方を入力に接続します。
- 光結合ソリッドステートリレーを使用する場合は、リレーの極性に注意してください。
- 通常、motion.probe-input に接続され、フロートスイッチを使用することもできます。

ご覧のとおり、プラズマテーブルはピンを集中的に使用し、通常の estops が追加される前にすでに約 15 の入力を消費しています。 他の人は他の見解を持っていますが、MPG、スケールと軸の選択スイッチ、および時間の経過とともに追加したいその他の機能を可能にするために、Mesa7i76e が安価な 7i96 よりも好まれるというのが筆者の意見です。 テーブルでサーボを使用している場合は、いくつかの選択肢があります。 他のサプライヤもありますが、Mesa エコシステムを中心にマシンを設計すると、THCAD ボードを使用してアーク電圧を読み取ることが簡単になります。

3.11.16.5 トーチブレイクアウェイ

- 前述のように、トーチがクラッシュして脱落した場合にトリガーされる分離センサーを取り付ける必要があります。
- 通常、これは halui.program-pause に接続されるため、障害を修正してプログラムを再開できます。

3.11.17 プラズマコントローラーの G コード

ほとんどのプラズマコントローラーは、G コードから設定を変更する方法を提供します。 Linuxcnc は、アナログコマンドの場合は M67 / M68、デジタル（オン/オフコマンド）の場合は M62-M65 を介してこれをサポートします。 これがどのように実装されるかは完全に任意です。 LinuxCNCPlasmaC 構成がこれをどのように行うかを見てみましょう。

1.1.1.1 PlasmaC で材料設定を選択し、その材料の送り速度を使用します。

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

Note

PlasmaC マテリアルテーブルに多数のエントリがあるユーザーは、Q1 パラメータを増やす必要がある場合があります（例：Q2）

3.11.17.1 THC 操作の有効化/無効化：

```
M62 P2 will disable THC (synchronised with motion)
M63 P2 will enable THC(synchronised with motion)
M64 P2 will disable THC (immediately)
M65 P2 will enable THC(immediately)
```

3.11.17.2 切削速度を下げる:(例：穴あけ用）

```
M67 E3 Q0 would set the velocity to 100% of requested~speed
M67 E3 Q40 would set the velocity to 40% of requested~speed
M67 E3 Q60 would set the velocity to 60% of requested~speed
M67 E3 Q100 would set the velocity to 100% of requested~speed
```

3.11.17.3 カッター補正：

```
G41.1 D#<_hal[plasmac_run.kerf-width-f]> ; for left of programmed path
G42.1 D#<_hal[plasmac_run.kerf-width-f]> for right of programmed path
G40 to turn compensation off
```

Note

インテグレーターは、上記のさまざまな LinuxcncG コードコマンドの Linuxcnc ドキュメントに精通している必要があります。

3.11.18 外部オフセットとプラズマ切断

外部オフセットは、バージョン 2.8 で Linuxcnc に導入されました。外部とは、軌道計画担当者が何も知らない G コードの外部にオフセットを適用できることを意味します。例を挙げて説明するのが最も簡単です。カム上のローブを加工するために数式によって外部オフセットが適用されている旋盤を想像してみてください。そのため、旋盤はカット直径を固定直径に設定して盲目的に回転し、外部オフセットがツールを出し入れして、適用された外部オフセットを介してカムローブを加工します。このカムを加工するように旋盤を構成するには、軸の速度と加速度の一部を外部オフセットに割り当てる必要があります。そうしないと、工具が移動できません。ここで、ini 変数 OFFSET_AV_RATIO が登場します。速度と加速度の 20％を Z 軸の外部オフセットに割り当てる必要があると判断したとします。これを 0.2 に設定します。この結果、旋盤の Z 軸の最大速度と加速度は 80％になります。

外部オフセットは、THC を介して Z 軸のトーチ高さを調整するための非常に強力な方法です。 しかし、プラズマはすべて高速で急速な加速であるため、これらのパラメータを制限することは意味がありません。 幸いなことに、プラズママシンでは、Z 軸は THC によって 100％制御されているか、そうでないかのどちらかです。 Linuxcnc の外部オフセットの開発中に、G コードと THC による Z 軸の動きは相互に排他的であることが認識されました。 これにより、外部オフセットをだまして、常に 100％の速度と加速度を与えることができます。 これを行うには、ini ファイルでマシンの Z 軸の速度と加速度の設定を 2 倍にし、OFFSET_AV_RATIO = 0.5 に設定します。 そうすれば、最大速度と加速度の 100％がプロービングと THC の両方で利用できるようになります。

例：5mm ボールねじへのダイレクトドライブを備えた NEMA23 モーターを備えたメートル法の機械では、60mm /秒の最大速度と 700mm /秒/秒の加速度がステップを失うことなく安全な値であると判断されました。 このマシンでは、ini ファイルの Z 軸を次のように設定します。

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

この軸に関連付けられたジョイントでは、速度変数と加速度変数が次のように設定されます。

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

外部オフセット（V 2.8 以降の場合）の詳細については、INI ファイルドキュメントの[AXIS_ <letter>]セクションおよび Linuxcnc ドキュメントの外部軸オフセットをお読みください。

3.11.19 MesaTHCAD でアーク電圧を読み取る

Mesa THCAD ボードは、プラズマ切断に関連する敵対的なノイズの多い電気環境向けに設計された、非常に手頃な価格の正確な電圧-周波数変換器です。 内部的には 0〜10 ボルトの範囲があります。 この範囲は、ドキュメントに記載されているように、いくつかの抵抗を追加することで簡単に拡張できます。 このボードには、0〜5 ボルトの範囲の新しい THCAD-5、0〜10 ボルトの範囲の THCAD-10、および 300 ボルトの拡張範囲用に事前に校正された THCAD-300 の 3 つのバージョンがあります。 各ボードは個別に校正され、0 ボルトおよびフルスケールでの周波数を示すステッカーがボードに貼られています。 LinuxCNC で使用する場合は、ボード上の適切なリンクで 1/32 除数を選択することをお勧めします。 この場合、指定された周波数も 32 で除算してください。これは 1 kHz サーボスレッドに適しています。また、THCAD が出力を平均化して平滑化するための時間を長くすることができます。

THCAD 出力をデコードする方法については多くの混乱があるため、次の架空のキャリブレーションデータを使用して Mesa7i76e と THCAD-10 について少し考えてみましょう。

- フルスケール 928,000Hz（1/32 29,000 Hz）
- 0 volt 121,600 Hz (1/32 3,800 Hz)

フルスケールは 10 ボルトであるため、1 ボルトあたりの周波数は次のようになります。

```
(29,000 - 3,800) / 10 = 2,520 Hz per volt
```

（29,000-3,800）/ 10 = 2,520 Hz /ボルト

```
(2520 * 5) + 3,800 = 16,400
```

これで、周波数を同等の電圧に変換する方法がかなり明確になるはずです。

```
Volts = (frequency - 3,800) / 2,520
```

1.1.1.1 THCAD 接続

高電圧側：

- 分割または生のアーク電圧を IN +および IN-に接続します
- 相互接続ケーブルシールドをシールド接続に接続します。
- もう一方のシールド端子をフレームアースに接続します。

7i76e に接続されていると仮定して、出力をスピンドルエンコーダ入力に接続します。

- THCAD + 5v から TB3 ピン 6（+5 VP）
- THCAD -5v から TB3 ピン 1（GND）
- THCAD FOUT +から TB3 ピン 7（ENC A +）
- THCAD FOUT-から TB3 ピン 8（ENC A-）

3.11.19.1 THCAD FOUT-から TB3 ピン 8（ENC A-）

ini ファイルに次の行があることを確認してください（Mesa 7i76e を想定）。

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

コントローラの電源を入れ、Halshow（Axis：Show Homing Configuration）を開き、ドリルダウンして hm2_7i76e.0.encoder.00.velocity ピンを見つけます。 0 ボルトが適用されると、0 ボルトの周波数（この例では 3,800）の周りをホバリングするはずです。 9 ボルトのバッテリーをつかみ、IN +と IN-に接続します。 THCAD-10 の場合、予想される速度を計算できるようになりました。 （架空の例では 26,480）。 このテストに合格すると、LinuxCNC プラズマコントローラーを構成する準備が整います。

3.11.19.2 使用するモデル THCAD

THCAD-5 は、オームセンシングに使用する場合に便利です。 THCAD-10 がより柔軟なデバイスであり、スケーリングを簡単に変更できることは間違いありません。 ただし、分圧器が組み込まれた安価なプラズマ切断機で問題となる可能性のある注意点が 1 つあります。 つまり、内部抵抗は THCAD によってそれ自体の外部抵抗の一部であると検出され、誤った結果を返す可能性があります。 たとえば、Everlast プラズマカッターの 16：1 仕切りは、24：1 として扱う必要があります（50：1 は 75：1 になります）。 これは、より評判の良いブランド（Thermal Dynamics、Hypertherm、ESAB など）では問題になりません。 したがって、予想よりも低い切断電圧が見られる場合は、THCAD を再構成して生のアーク電圧を読み取ることが望ましい場合があります。

プラズマアーク電圧は潜在的に致命的であることを思い出して、ここにいくつかの提案された基準があります。

パイロットアークスタート重大な EMI が発生する可能性は低いため、当社の建設ガイドラインに従えば、コントロールパネルに THCAD を安全に設置できるはずです。

- 分圧器がない場合は、プラズマ切断機内にスケーリング抵抗を取り付けてコントロールパネルに THCAD を取り付けるか、HF 始動機の提案に従ってください。

- 分圧器を使用している場合は、コントロールパネルに THCAD-10 を取り付けます。 120 アンペアのサーマルダイナミクスプラズマカッターを使用したこの構成では、問題はありませんでした。

HF スタート周波数信号は EMI ノイズの影響をはるかに受けにくいため、THCAD をインバーターに取り付けます。

- 分圧器がなく、プラズマ切断機の内部に余裕がある場合は、プラズマ切断機の内部に THCAD-300 を取り付けます。
- 分圧器がなく、プラズマ切断機の内部にスペースがない場合は、プラズマ切断機の外部の金属ケースに THCAD-10 を取り付け、内部の IN +と IN-のそれぞれにスケーリング抵抗の 50%を取り付けます。 プラズマカッターケースなので、ケースから致命的な電圧が出ることはありません。
- 分圧器をお持ちの場合は、プラズマ切断機の外側の金属ケースに THCAD-10 を取り付けてください

コネクタに表示される生のアーク電圧この場合、アークの開始方法に関係なく、致命的なショックを回避するために回路にすでに抵抗が含まれている可能性があるため、この抵抗（通常は 200k オーム）を考慮できるように THCAD-10 をお勧めします これらの抵抗器は THCAD-300 によって報告された電圧を歪めるため、スケーリング抵抗器を選択する場合。

3.11.20 ポストプロセッサとネスティング

プラズマは、次の点で他の CNC 操作と同じです。

1. CAD で設計されています（DXF または場合によっては SVG 形式で出力されます）。
2. CAM で処理され、マシンにロードされる最終的な G コードが生成されます
3. CNCG コードコマンドによる部品の切断。

Inkscape および G-Code ツールで良い結果を達成する人もいますが、SheetCam は非常に手頃な価格のソリューションであり、Linuxcnc で利用できるポストプロセッサが多数あります。 SheetCam は、プラズマ切断用に設計された多くの高度な機能を備えており、価格も手頃で、定期的なプラズマ切断を行う人にとっては簡単です。

3.11.21 ノイズの多い電気環境向けの設計

プラズマ切断は本質的に非常に敵対的で騒々しい電気環境です。 EMI の問題がある場合、問題は正しく機能しません。 より明白な例では、トーチを発射するとコンピューターが再起動しますが、他の奇妙な症状がいくつも発生する可能性があります。 それらはほとんどすべて、トーチが切断されているとき、つまり最初に発射されたときにのみ発生します。

したがって、システムビルダーは、コンポーネントを慎重に選択し、電磁干渉（EMI）の影響を回避するために、この過酷な環境に対処するためにゼロから設計する必要があります。 これを怠ると、無数の無駄なトラブルシューティングが発生する可能性があります。

Mesa 7i76e や安価な 7i96 などのイーサネットボードを選択すると、PC を電子機器やプラズママシンから離して配置できるようになります。 このハードウェアでは、ノイズ耐性がはるかに高い 24 ボルトのロジックシステムを使用することもできます。 コンポーネントは、メインアースに接続された金属製の筐体に取り付ける必要があります。 主電源接続に EMI フィルタを取り付けることを強くお勧めします。 最も簡単な方法は、PC や電化製品で一般的に使用されている EMI フィルター付き主電源 IEC コネクターを使用することです。これにより、余分な作業を行うことなくこれを実現できます。 主電源、高電圧モーターワイヤ、およびロジック信号が互いに可能な限り分離されるように、エンクロージャ内のコンポーネントのレイアウトを計画します。 交差する必要がある場合は、90 度に保ちます。

MesaElectronics の PeterWallace が提案します。 「分圧器を備えた CNC 互換のプラズマ源がある場合は、他のすべてのモーションハードウェアを使用して電子機器の筐体内に THCAD を取り付けます。 手動プラズマ源があり、生のプラズマ電圧を読み取っている場合は、THCAD をプラズマ源のできるだけ近くに取り付けます（プラズマ源のケースが収まる場合でも）。この場合、すべてのローサイドを確認してください。 THCAD 接続はプラズマ源から完全に分離されています。 THCAD にシールドボックスを使用する場合、シールドはプラズマ源のアースではなく、電子エンクロージャのアースに接続する必要があります。」

モーターケースとトーチから別のアース線を機械の中央のスター接地点に戻すことをお勧めします。 プラズマアースリードをこのポイントに接続し、オプションでアースロッドをマシンのできるだけ近くでアースに接続します（特に、HF スタートプラズママシンの場合）。

モーターへの外部配線は、回路を通過する電流を処理できるようにシールドし、適切なサイズにする必要があります。 シールドはモーター側で未接続のままにし、コントロールボックス側で接地する必要があります。 コントロールボックスへのコネクタに追加のピンを使用することを検討してください。これにより、アースをコントロールボックスまで延長し、ステッピング/サーボモーターコントローラー自体でシャーシにアースすることができます。

私たちは、オーム検知回路に誘導される電気ノイズに問題を抱えている商用システムビルダーを少なくとも 1 人知っています。 これはフェライトビーズを使用してケーブルを巻くことで軽減できますが、オーム検知信号が電子機器の筐体に入る場所に電力線フィルタを介してフィードを追加することもお勧めします。

予算内でプラズママシンを構築するマスターである TommyBerisha は、次のように述べています。 それらは非常に優れており、フィルタリングは優れており、完全に絶縁されており、電流が制限されています（これは何かがうまくいかない場合に非常に重要になります）。 負の出力端子に接続するので、切断する必要があります。接地接点のない電源ケーブルを使用するだけです）」。

3.11.22 ウォーターテーブル

トーチのカットレベルの下の最小水位は約 40mm である必要があります。スラットの下にスペースがあると、カット中に水が水平になって逃げることができます。カットされている金属プレートの上に少し水があると、それが取り除かれるので本当にいいです。 ほんの少しのほこりの中で、それを水没させるのが最善の方法ですが、トーチを腐食させるので、パートタイムで使用するシステムにはお勧めできません。 重曹を水に加えると、ス

ラットが水中にあるときに腐食することがなく、水蒸気の臭いも軽減されるため、テーブルを長年にわたって良好な状態に保つことができます。 一部の人々は、圧縮空気入口を備えた貯水池を使用して、必要に応じて貯水池から地下水面まで水を押し上げ、水位の変化を可能にします。

3.11.23 ダウンドラフトテーブル

多くの市販のテーブルはダウンドラフト設計を利用しているため、ファンを使用してスラットから空気を吸い込み、煙や火花を捕らえます。 多くの場合、テーブルはゾーン化されているため、トーチの下のセクションのみが出て行くベントに開かれ、多くの場合、エアラムとエアソレノイドを使用してシャッターが開きます。 モーションピンの 1 つと lincurve コンポーネントからの軸またはジョイントの位置を使用して、下降気流ゾーンを正しい出力ピンにマップする場合、これらのゾーンのトリガーは比較的簡単です。

3.11.24 速度と加速のための設計

プラズマ切断では、速度と加速が重要です。 加速が高ければ高いほど、コーナーをネゴシエートするときにマシンが減速する必要が少なくなります。 これは、ねじり剛性を犠牲にすることなく、ガントリーを可能な限り軽量にする必要があることを意味します。 100mm x 100mm x 2mm のアルミニウムボックスセクションは、80mm x 80mm の T スロット押し出しと同等のねじり剛性を備えていますが、62%軽量です。 では、T スロットの利便性は追加の構造を上回っていますか？

3.11.25 モーター回転あたりの移動距離

ステッピングモーターは共振に悩まされており、ダイレクトドライブピニオンはモーターが不利な条件下で動作していることを意味する可能性があります。 理想的には、プラズママシンの場合、モーター 1 回転あたり約 15～25mm の距離が理想的と見なされますが、1 回転あたり約 30mm でも許容されます。 Z 軸には 3：1 または 5：1 の減速ドライブを備えた 5mm ピッチのボールねじが理想的です。

3.11.26 PlasmaCLinuxCNC プラズマ構成

HAL コンポーネント（plasmac.hal）と Axis および Gmoccapy の完全な構成で構成される PlasmaC 構成は、2015 年頃からプラズマコントローラーの理解を深めてきた LinuxCNC オープンソース運動の多くからかなりの意見を受け取っています。 PlasmaC を現在の動作状態にするためのテストと開発作業。 回路設計から G コードの制御と構成まですべてが含まれています。 さらに、PlasmaC は Proma 150 などの外部 THC をサポートしますが、Mesa コントローラーと組み合わせると、インテグレーターが敵対的なプラズマ環境に対処するために構築された MesaTHCAD 電圧-周波数変換器を含めることができるため、実際に独自のものになります。

PlasmaC はスタンドアロンで設計されており、カッティングチャートを含める機能が含まれていますが、SheetCam などのポストプロセッサで使用される機能も含まれています。

PlasmaC システムは、Linuxcnc のバージョン 2.8 以降に含まれるようになりました。 このガイドの最初のバージョンが作成されて以来、現在はかなり成熟しており、大幅に強化されています。 Plasmac は、オープンソース価格で独自のハイエンドプラズマ制御システムのすべての機能を備えているため、今後何年にもわたって LinuxCNC のプラズマサポートを定義します。

3.11.27 HyperthermRS485 制御

一部の Hypertherm プラズマカッターには、コントローラー（Linuxcnc など）が amps.pressure とモードを設定できるようにする RS485 インターフェースがあります。 これを実現するために、多くの人が Python で記述されたユーザースペースコンポーネントを使用しています。 最近では、Plasmac はこのインターフェイスをネイティブにサポートするようになりました。 使用方法については、Plasmac のドキュメントを参照してください。

Hypertherm とユーザースペースコンポーネントで使用される遅いボーレートの組み合わせにより、マシンの状態を変更するのがかなり遅くなるため、通常、切断中にその場で設定を変更することはできません。

PC 側で使用する RS485 インターフェースを選択する際、ユーザーは USB から RS485 へのインターフェースは信頼できないと報告しています。 ハードウェアベースの RS232 インターフェース（PCI / PCIe やマザーボードポートなど）と適切な RS485 コンバーターを使用すると、信頼性の高い結果が得られます。 一部のユーザーは、Sunix P / N：SER5037A PCI RS2322 カード、汎用 XC4136 RS232 から RS485 へのコンバーター（USB ケーブルも含まれる場合があります）で成功したと報告しています。

3.11.28 プラズマ切断用ポストプロセッサ

CAM プログラム（コンピューター支援製造）は、CAD（コンピューター支援設計）と最終的な CNC（コンピューター数値制御）操作の間の架け橋です。 多くの場合、G コードの特定のマシンまたは方言に対して生成されるコードを定義するために、ユーザーが構成可能なポストプロセッサーが含まれています。

多くの Linuxcnc ユーザーは、Inkscape を使用して.SVG ベクターベースのファイルを G コードに変換することに完全に満足しています。 趣味や家庭でプラズマを使用している場合は、このオプションを検討してください。 ただし、ニーズがより複雑な場合は、おそらく最良で最も手頃な価格のソリューションは SheetCam です。 SheetCam は Windows と Linux の両方をサポートしており、PlasmaC 構成を含むポストプロセッサを利用できます。 SheetCam を使用すると、マテリアルのシート全体にパーツをネストでき、ニーズに合わせてツールセットとコードスニペットを構成できます。 SheetCam ポストプロセッサは、Lua プログラミング言語で記述されたテキストファイルであり、通常、正確な要件に合わせて簡単に変更できます。 詳細については、SheetCamWeb サイトとそのサポートフォーラムを参照してください。

別の人気のあるポストプロセッサーが人気のある Fusion360 パッケージに含まれていますが、含まれているポストプロセッサーにはいくつかのカスタマイズが必要です。

LinuxCNC は CNC アプリケーションであり、この入門的な議論以外の CAM 技術の議論は LinuxCNC の範囲外です。

4章 ユーザーインターフェース

4.1 AXIS GUI

3.11.29 前書き

AXIS は、ライブプレビューとバックプロットを備えた LINUXCNC のグラフィカルフロントエンドです。 PYTHON で書かれており、TK と OPENGL は、そのユーザーインターフェイスを表示します。

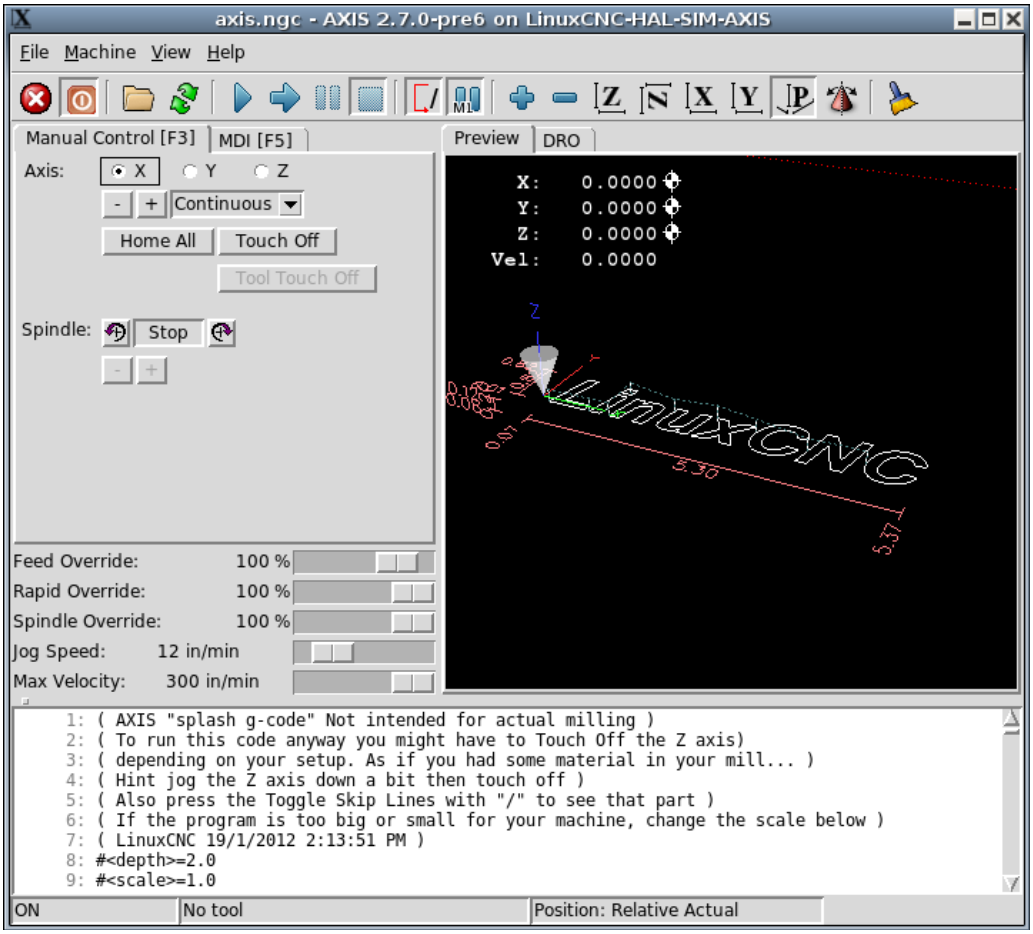


図 4-38 AXIAS ウインドウ

3.11.30 入門

構成が現在 AXIS を使用するように設定されていない場合は、.INI ファイルを編集して構成を変更できます。 [表示]セクション DISPLAY 行を変更して DISPLAY = AXIS を読み取ります。 サンプル構成 SIM / AXIS.INI は、フロントエンドとして AXIS を使用するようにすでに構成されています。

1.1.1.1 典型的なセッション

1. LINUXCNC を起動します。
2. E-STOP（F1）をリセットし、マシン電源（F2）をオンにします。
3. すべての軸を原点復帰します。
4. G コードファイルをロードします。
5. プレビュープロットを使用して、プログラムが正しいことを確認します。

6. 材料をロードします。
7. ジョギングし、必要に応じて TOUCH OFF ボタンを使用して、各軸に適切なオフセットを設定します。
8. プログラムを実行します。

Note

同じプログラムを再度実行するかどうかは、セットアップと要件によって異なります。 より多くのマテリアルをロードしてオフセットを設定する必要がある場合があります または、移動してオフセットを設定してから、プログラムを再実行してください。 マテリアルが固定されている場合は、プログラムを再度実行するだけでよい場合があります。 RUN コマンドの詳細については、マシンメニューを参照してください。

3.11.31 AXIS ディスプレイ

AXIS ウィンドウには、次の要素が含まれています。

- 次のいずれかを表示する表示領域：
- さまざまなアクションを実行できるメニューバーとツールバー
- 手動制御タブ-マシンを動かしたり、スピンドルをオンまたはオフにしたり、INI ファイルに含まれている場合はクーラントをオンまたはオフにしたりできます。
- MDI タブ-G コードプログラムを一度に 1 行ずつ手動で入力できます。 これは、どのモーダル G コードが有効であるかを示すアクティブ G コードも示します。
- フィードオーバーライド-プログラムされたモーションの速度をスケーリングできます。 デフォルトの最大値は 120％で、INI ファイルで別の値に設定できます。 詳細については、INI ファイルの表示セクションを参照してください。
- スピンドルオーバーライド-スピンドル速度を増減することができます。
- ジョグ速度-INI ファイルで設定された制限内でジョグ速度を設定できます。 詳細については、INI ファイルの表示セクションを参照してください。
- 最大速度-プログラムされたすべてのモーション（スピンドル同期モーションを除く）の最大速度を制限できます。
- ロードされた G コードを表示するテキスト表示領域。
- マシンの状態を示すステータスバー。 このスクリーンショットでは、マシンの電源がオンになっていて、ツールが挿入されておらず、表示される位置は相対（すべてのオフセットを表示）と実際（フィードバック位置を表示）です。

1.1.1.1 メニュー項目

.INI ファイルの構成方法によっては、一部のメニュー項目がグレー表示される場合があります。 構成の詳細については、INI の章を参照してください。

FILE MENU

- 開く・・・-標準のダイアログボックスを開いて、AXIS にロードする G コードファイルを開きます。 フィルタプログラムを使用するように LINUXCNC を設定している場合は、それを開くこともできます。 詳細については、INI 構成の FILTER セクションを参照してください。
- 最近のファイル・・・最近開いたファイルのリストを表示します。

- 編集・・・-INI ファイルにエディターが構成されている場合は、現在の G コードファイルを開いて編集します。 使用するエディターの指定の詳細については、DISPLAY セクションを参照してください。
- 再読込・・・現在の G コードファイルをリロードします。 編集した場合は、変更を有効にするためにリロードする必要があります。 ファイルを停止して最初からやり直したい場合は、ファイルをリロードしてください。 ツールバーのリロードはメニューと同じです。
- 名前をつけて保存・・・現在の G コードファイルを新しい名前で保存します。
- プロパティ・・・早送りと切削送りの合計。 アクセラレーション、ブレンディング、パスモードは考慮されていないため、報告される時間は実際の実行時間よりも短くなることはありません。
- 工具テーブルを編集・・・-エディタを定義している場合は編集と同じで、ツールテーブルを開いて編集できます。
- 工具テーブルの再読込・・・ツールテーブルを編集した後、リロードする必要があります。
- ラダーエディタ・・・クラシックラダーをロードしている場合は、ここから編集できます。 詳細については、CLASSICLADDER の章を参照してください。
- 終了・・・現在の LINUXCNC セッションを終了します。

MACHINE MENU

- 非常停止（トグル）F1・・・-非常停止の状態を変更します。
- 機械電源（トグル）F2・・・-緊急停止がオンになっていない場合は、機械電源の状態を変更します。
- プログラムの実行・・・現在ロードされているプログラムを最初から実行します。
- 選択した行から実行・・・最初から開始する行を選択します。 これにより、ツールが最初の行の前の予想される位置に移動し、その後、残りのコードが実行されるため、注意して使用してください。

Warning

G コードプログラムにサブルーチンが含まれている場合は、[選択した行から実行]を使用しないでください。

- ステップ・・・プログラムをシングルブロックで実行します。
- 一時停止・・・プログラムを一時停止します。
- 再開・・・一時停止から実行を再開します。
- 停止・・・実行中のプログラムを停止します。 停止後に実行を選択すると、プログラムは最初から開始されます。
- オプショナルストップ(M1)・・・M1 に到達し、これがチェックされている場合、プログラムの実行は M1 行で停止します。 [再開]を押して続行します。・「/」で始まる行をスキップする-行が/で始まり、これがチェックされている場合、その行はスキップされます。
- MDI 履歴のクリア・・・MDI 履歴ウィンドウをクリアします。
- MDI 履歴からコピー・・・MDI 履歴をクリップボードにコピーします
- MDI 履歴に貼り付け・・・クリップボードから MDI 履歴ウィンドウに貼り付けます
- キャリブレーション・・・キャリブレーションアシスタント（EMCCALIB.TCL）を開始します。 キャリブレーションは HAL ファイルを読み取り、[AXIS_L]、[JOINT_N]、[SPINDLE_S]、または[TUNE]セクションにある INI ファイルの変数を使用するすべての SETP に対して、編集およびテスト可能なエントリを作成します。
- HAL 構成の表示・・・HAL コンポーネント、ピン、パラメーター、信号、機能、およびスレッドを監視できる HAL 構成ウィンドウを開きます。

- HAL メーター・・・単一の HAL ピン、信号、またはパラメーターを監視できるウィンドウを開きます。
- HAL スコープ・・・HAL 値と時間の関係をプロットできる仮想オシロスコープを開きます。
- LINUXCNC ステータスの表示・・・LINUXCNC のステータスを示すウィンドウを開きます。
- デバッグレベルの設定・・・デバッグレベルを表示したり、一部を設定したりできるウィンドウを開きます。
- 原点復帰・・・1 つまたはすべての軸を原点復帰します。
- 原点復帰解消・・・1 つまたはすべての軸の原点復帰を解除します。
- ゼロ座標系・・・選択した座標系ですべてのオフセットをゼロに設定します。
- ワークピースへの工具タッチオフ・・・タッチオフを実行する場合、入力された値は、軸オフセット（G92）によって変更された、現在のワークピース（G5x）座標系を基準にしています。 タッチオフが完了すると、選択した軸の相対座標が入力された値になります。 G コードの章の G10L10 を参照してください。
- 工具のフィクスチャへのタッチオフ・・・タッチオフを実行する場合、入力された値は 9 番目（G59.3）の座標系を基準にしており、軸オフセット（G92）は無視されます。 これは、マシンの固定位置に工具タッチオフフィクスチャがあり、相対座標で長さがゼロの工具の先端がフィクスチャの原点になるように 9 番目（G59.3）の座標系が設定されている場合に便利です。 は 0 です。G コードの章の G10L11 を参照してください。

VIEW MENU

- 上面図・・・上面図（または Z ビュー）は、Z 軸に沿って正から負に見た G コードを表示します。 このビューは、X と Y を見るのに最適です。
- 回転した上面図・・・回転した上面図（または回転した Z ビュー）には、Z 軸に沿って正から負に見た G コードも表示されます。 ただし、ディスプレイに合わせて X 軸と Y 軸を 90 度回転させて表示すると便利な場合があります。 このビューは、X と Y を見るのにも最適です。
- 側面図・・・側面図（または X ビュー）は、X 軸に沿って正から負に見た G コードを表示します。
- 正面図・・・正面図（または Y ビュー）は、Y 軸に沿って負から正に見た G コードを表示します。 このビューは、X と Z を見るのに最適です。
- パースペクティブビュー・・・パースペクティブビュー（または P ビュー）は、調整可能な視点からパーツを見る G コードを表示します。デフォルトでは、X +、Y-、Z +です。 位置は、マウスとドラッグ/回転セレクターを使用して調整できます。 このビューは妥協したビューであり、2 次元ディスプレイに 3 つ（9 つまで！）の軸を表示しようとするとうまくいきますが、見づらい機能があり、視点を変更する必要があります。 。 このビューは、3 つ（から 9 つ）の軸すべてを一度に表示する場合に最適です。

Point of View

AXIS ディスプレイピックメニュービューは、上面図、正面図、および側面図を指します。 これらの用語は、CNC マシンの Z 軸が垂直で、正の Z が上にある場合に正しいものです。 これは、おそらく最も人気のあるアプリケーションである垂直ミルに当てはまり、ほとんどすべての EDM マシン、さらには部品が工具の下で回転する垂直タレット旋盤にも当てはまります。

ただし、トップ、フロント、サイドという用語は、Z 軸が水平である標準旋盤、Z 軸が水平である水平ミル、または逆垂直タレットなど、他の CNC マシンでは混乱する可能性があります。 部品が工具の上で回転し、Z 軸の正の方向が下になっている旋盤！

正の Z 軸は（ほぼ）常にパーツから離れていることを覚えておいてください。 したがって、マシンの設計に精通し、必要に応じてディスプレイを解釈してください。

- ディスプレイインチ・・・AXIS ディスプレイのスケーリングをインチに設定します。
- DISPLAYMM・・・AXIS ディスプレイのスケーリングをミリメートルに設定します。
- プログラムの表示・・・必要に応じて、ロードされた G コードプログラムのプレビュー表示を完全に無効にすることができます。
- プログラムラピッズの表示・・・ロードされた G コードプログラムのプレビュー表示では、常に送り速度の動き（G1、G2、G3）が白で表示されます。 ただし、必要に応じて、シアンの高速移動（G0）の表示を無効にすることができます。
- アルファブレンドプログラム・・・このオプションを使用すると、複雑なプログラムのプレビューが見やすくなりますが、プレビューの表示が遅くなる可能性があります。
- ライブプロットの表示・・・必要に応じて、工具の移動に伴う送り速度パス（G1、G2、G3）の強調表示を無効にすることができます。
- ツールの表示・・・必要に応じて、ツールコーン/シリンダーの表示を無効にすることができます。
- 範囲の表示・・・必要に応じて、ロードされた G コードプログラムの範囲（各軸方向の最大移動量）の表示を無効にすることができます。
- オフセットの表示・・・選択したフィクスチャオフセット（G54-G59.3）の原点位置は、赤、青、緑の 3 本の直交線のセットとして表示できます。 このオフセット原点（またはフィクスチャゼロ）の表示は、必要に応じて無効にすることができます。
- マシン移動限界の表示・・・INI ファイルで設定されている、各軸のマシンの最大移動制限は、赤い破線で描かれた長方形のボックスとして表示されます。 これは、新しい G コードプログラムをロードするとき、または G コードプログラムをマシンの移動制限内に収めるために必要なフィクスチャオフセットの量を確認するときに役立ちます。 必要がなければ、遮断することができます。
- 速度の表示・・・速度の表示は、マシンが設計速度にどれだけ近づいているかを確認するのに役立つ場合があります。 必要に応じて無効にすることができます。
- 移動距離を表示・・・移動距離は、不明な G コードプログラムを初めて実行するときに知っておくと非常に便利な項目です。 ラピッドオーバーライドおよび送り速度オーバーライド制御と組み合わせることで、不要な工具および機械の損傷を回避できます。 G コードプログラムがデバッグされ、スムーズに実行されたら、必要に応じて「移動距離」表示を無効にすることができます。
- ライブプロットの消去・・・ツールが軸ディスプレイ内を移動すると、G コードパスが強調表示されます。 プログラムを繰り返すため、または関心のある領域をよりよく見るために、以前に強調表示されたパスをクリアすることができます。

- コマンド位置を表示・・・これは、LinuxCNC が移動しようとする位置です。 モーションが停止すると、これは LinuxCNC が保持しようとする位置です。
- 実際の位置を表示・・・実際の位置は、システムのエンコーダーから読み取られた、またはステップジェネレーターによってシミュレートされた測定位置です。 これは、PID 調整、物理的制約、位置量子化などの多くの理由により、コマンド位置とわずかに異なる場合があります。
- 機械座標を表示-これは、ホーミングによって確立された、オフセットされていない座標での位置です。
- 相対位置を表示-これは、G5x、G92、および G43 オフセットによって変更されたマシン位置です。

HELP MENU

- Axis について・・・私たちは皆、これが何であるかを知っています。
- クイックリファレンス・・・キーボードショートカットキーを表示します。

3.11.31.1 ツールバーボタン

Axis ディスプレイの左から右に、ツールバーボタン（[括弧内に]表示されているキーボードショートカット）は次のとおりです。


緊急停止（トグル）[F1]（E-Stop とも呼ばれます）


機械主電源（トグル）


G コードファイルを開く[O]


現在のファイルを再読み込み[Ctrl+R]


現在のファイルの実行 [R]


次の行の実行


実行の一時停止[P]実行の再開[S]


プログラム実行の停止[ESC]


オプショナルブロックスキップ（トグル）[Alt-M- /]

オプショナルストップ（トグル）[Alt-M-1]


ズームイン


ズームアウト


トップビュー


トップビュー（回転）


サイドビュー


フロントビュー


パースペクティブビュー


ドラッグモードと回転モードを切り替えます[D]


ライブバックプロットをクリアする[CTRL-K]

3.11.31.2 グラフィック表示領域

座標表示プログラム表示の左上隅には、各軸の座標位置表示があります。 軸がホームになっている場合は、番号の右側に原点記号が表示されます。

軸がリミットスイッチの 1 つにある場合、座標位置番号の右側にリミット記号が表示されます。

座標位置番号を正しく解釈するには、ステータスバーの POSITION：インジケーターを参照してください。 位置が MACHINEACTUAL の場合、表示される数値は機械座標系にあります。 相対実績の場合、表示される数値はオフセット座標系にあります。 表示される座標が相対的でオフセットが設定されている場合、表示にはシアンの機械原点マーカーが含まれます。

位置が COMMANDED の場合、G コードコマンドで指定された正確な座標が表示されます。 それが実際の場合、それはマシンが実際に移動した位置です。 これらの値は、次のエラー、不感帯、エンコーダの分解能、またはステップサイズが原因で、コマンドされた位置とは異なる場合があります。 たとえば、ミルで X 0.0033 への移動をコマンドしたが、ステッピングモーターの 1 ステップまたは 1 エンコーダカウントが 0.00125 の場合、コマンド位置は 0.0033 になる可能性がありますが、実際の位置は 0.0025（2 ステップ）または 0.00375 になります。 （3 ステップ）。

プレビュープロットファイルがロードされると、そのプレビューが表示領域に表示されます。 高速移動（G0 コマンドによって生成される移動など）は、シアンの線で表示されます。 送り速度での移動（G1 コマンドによって生成される移動など）は、白い実線で表示されます。 ドウェル（G4 コマンドによって生成されたものなど）は、小さなピンクの X マークとして表示されます。

フィード移動の前の G0（高速）移動は、プレビュープロットに表示されません。 T <N>（工具交換）後の早送りは、最初の送り移動が終わるまでプレビューに表示されません。 これらの機能のいずれかをオフにするには、G0 が移動する前に移動せずに G1 をプログラムします。

プログラムの範囲各軸のプログラムの範囲が表示されます。 最後に、最小および最大の座標値が示されます。 真ん中には、座標の違いが表示されています。

一部の座標が.INI ファイルのソフト制限を超えると、関連する寸法が別の色で表示され、ボックスで囲まれます。 下の図では、座標値を囲むボックスで示されているように、X 軸で最大ソフト制限を超えています。 プログラムの最小 X 移動量は-1.95、最大 X 移動量は 1.88、プログラムには 3.83 インチの X 移動量が必要です。 この場合、プログラムを移動してマシンの移動範囲内に収めるには、左にジョギングして、もう一度[X]をタッチします。

ツールコーンツールがロードされていない場合、ツールの先端の位置はツールコーンによって示されます。 ツールコーンは、ツールの形状、長さ、または半径に関するガイダンスを提供しません。

ツールがロードされると（たとえば、MDI コマンド T1 M6 を使用して）、円錐は円柱に変わり、ツールテーブルファイルで指定されたツールの直径が表示されます。

ジョギングの場合は黄色、高速移動の場合はかすかな緑、送り速度での直線移動の場合は赤、送り速度での円形移動の場合はマゼンタ。

グリッド軸は、直交ビューの場合、オプションでグリッドを表示できます。 [表示]の下の[グリッド]メニューを使用して、グリッドを有効または無効にします。 有効にすると、グリッドは上面図と回転した上面図に表示されます。 座標系が回転していない場合、グリッドは正面図と側面図にも表示されます。 グリッドメニューのプリセットは、INIFILE アイテム[DISPLAY] GRIDS によって制御されます。 指定しない場合、デフォルトは 10MM 20MM 50MM 100MM 1IN 2IN 5IN10IN です。

非常に小さいグリッドを指定すると、パフォーマンスが低下する可能性があります。

相互作用プレビュープロットの一部を左クリックすると、グラフィック表示とテキスト表示の両方で線が強調表示されます。 空の領域を左クリックすると、強調表示が削除されます。

マウスの左ボタンを押したままドラッグすると、プレビュープロットがシフト（パン）されます。

SHIFT キーを押しながらマウスの左ボタンを押したままドラッグするか、マウスホイールを押したままドラッグすると、プレビュープロットが回転します。 線が強調表示されている場合、回転の中心は線の中心です。 それ以外の場合、回転の中心はプログラム全体の中心になります。

マウスホイールを回転させるか、マウスの右ボタンを押したままドラッグするか、コントロールを押しながらマウスの左ボタンを押してドラッグすると、プレビュープロットがズームインまたはズームアウトされます。

プリセットビューアイコンの１つをクリックするか、V キーを押すと、複数のプリセットビューを選択できます。

3.11.31.3 テキスト表示領域

プログラムの行を左クリックすると、その行がグラフィック表示とテキスト表示の両方で強調表示されます。

プログラムの実行中は、現在実行中の行が赤で強調表示されます。 ユーザーが行を選択していない場合、テキストディスプレイは自動的にスクロールして、現在の行を表示します。

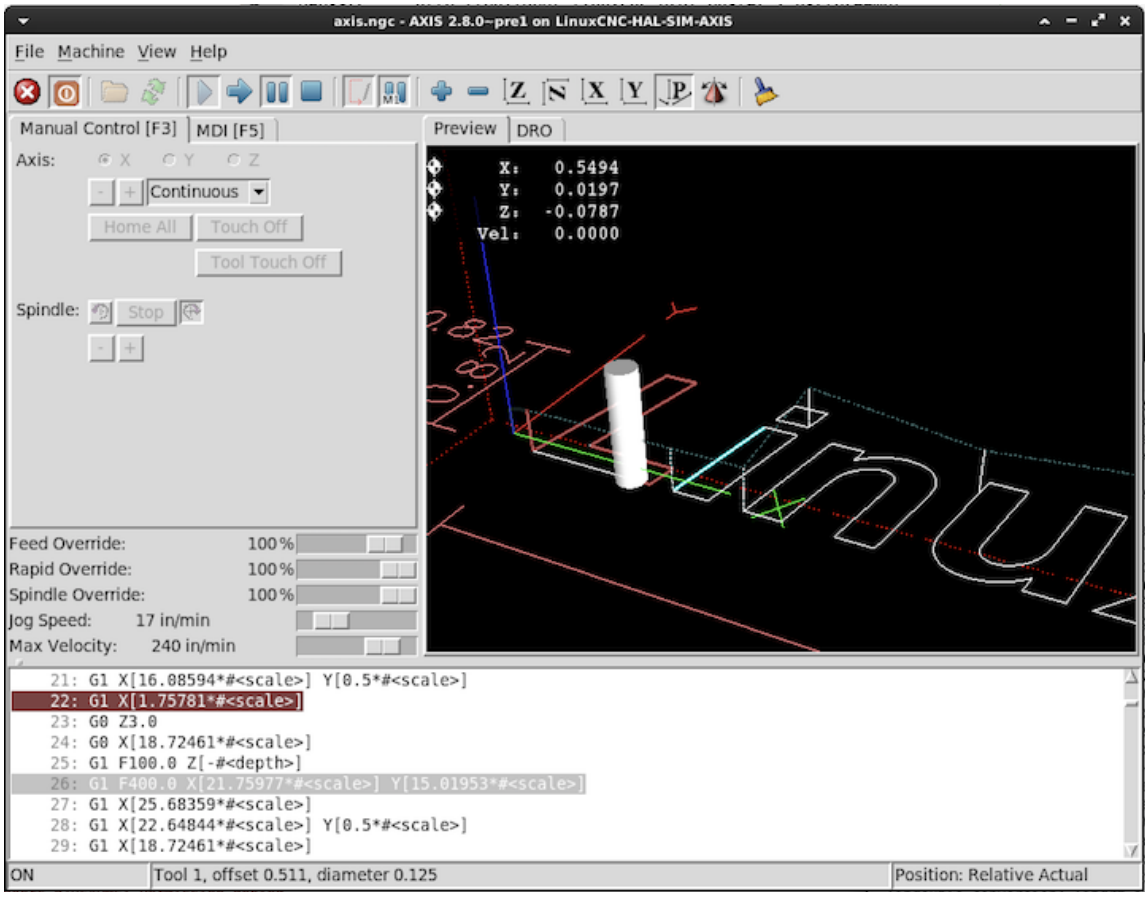


図 4-39

3.11.31.4 手動操作

マシンの電源がオンになっているがプログラムを実行していないときは、[手動制御]タブの項目を使用して、マシンを移動したり、スピンドルとクーラントを制御したりできます。

マシンの電源が入っていない場合、またはプログラムが実行されている場合、手動制御は使用できません。

以下に説明する項目の多くは、すべてのマシンで役立つわけではありません。 AXIS が特定のピンが HAL に接続されていないことを検出すると、[手動制御]タブの対応する項目が削除されます。 たとえば、HAL ピン SPLAND.0.BRAKE が接続されていない場合、ブレーキボタンは画面に表示されません。

環境変数 AXIS_NO_AUTOCONFIGURE が設定されている場合、この動作は無効になり、すべての項目が表示されます。

AXIS グループ AXIS を使用すると、マシンを手動で移動できます。 このアクションはジョギングとして知られています。 まず、移動する軸をクリックして選択します。 次に、目的の移動方向に応じて、[+]または[-]ボタンをクリックして押したままにします。 最初の 4 つの軸は、矢印キー（X および Y）、PAGEUP および PAGEDOWN キー（Z）、および[および]キー（A）によっても移動できます。

[連続]が選択されている場合、ボタンまたはキーが押されている限り、モーションは続行されます。 別の値を選択した場合、ボタンがクリックされるか、キーが押されるたびに、マシンは表示された距離を正確に移動します。 デフォルトでは、使用可能な値は 0.1000、0.0100、0.0010、0.0001 です。

増分の設定の詳細については、「表示」セクションを参照してください。

HOMING（IDENTITY KINEMATICS）INIFILE 設定[KINS] JOINTS は、システムのジョイントの総数を定義します。 ジョイントは、ホームスイッチまたは即時ホーミング用に構成できます。 ジョイントは、ジョイントのホーミンググループの順序を整理するホームシーケンスを指定できます。

すべてのジョイントがホーミング用に構成されており、有効なホームシーケンスがある場合、ホーミングボタンに[すべてホーム]と表示されます。 [すべてホーム]ボタン（または CTRL-HOME キー）を押すと、定義されたホームシーケンスを使用してすべてのジョイントのホーミングが開始されます。 HOME キーを押すと、原点復帰シーケンスが定義されていない場合でも、現在選択されている軸に対応するジョイントが原点復帰します。

すべての軸に有効なホームシーケンスがあるとは限らない場合、原点復帰ボタンに[原点軸]が表示され、現在選択されている軸のジョイントのみが原点復帰されます。 各軸を個別に選択して配置する必要があります。

ドロップダウンメニューの[マシン/ホーミング]は、ホーム軸の代替方法を提供します。 ドロップダウンメニューの[マシン/ホーミング解除]は、軸のホームを解除する手段を提供します。

詳細については、ホーミング構成の章を参照してください。

ホーミング（非アイデンティティキネマティクス）操作はアイデンティティキネマティクスの操作と似ていますが、ホーミングの前に、選択ラジオボタンでジョイントを番号で選択します。 すべてのジョイントがホーミング用に構成されており、有効なホームシーケンスがある場合、ホーミングボタンには[すべてホーム]と表示されます。 それ以外の場合、ホーミングボタンにはホームジョイントが表示されます。

詳細については、ホーミング構成の章を参照してください。

タッチオフタッチオフまたは END キーを押すと、現在の軸の G5X オフセットが変更され、現在の軸の値が指定された値になります。 式は、変数が参照されない場合を除いて、RS274NGC プログラムのルールを使用して入力できます。 結果の値は数値として表示されます。

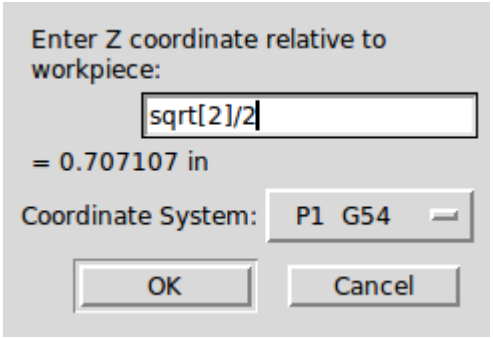


図 4-40

ツールタッチオフツールタッチオフボタンを押すと、現在ロードされているツールのツール長とオフセットが変更され、現在のツールチップ位置が入力された座標と一致するようになります。

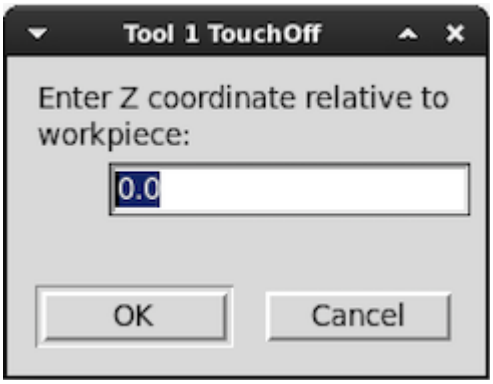


図 4-41

マシンメニューのワークピースへのツールタッチオフおよびフィクスチャオプションへのツールタッチオフも参照してください。

オーバーライド制限オーバーライド制限を押すと、マシンは一時的に物理的なリミットスイッチからジョギングすることができます。 このチェックボックスは、リミットスイッチが作動した場合にのみ使用できます。 オーバーライドは、1 回のジョグ後にリセットされます。 軸が別々の正と負のリミットスイッチで構成されている場合、LinuxCNC は正しい方向でのみジョグを許可します。 オーバーライド制限は、ソフト制限を超えるジョギングを許可しません。 軸のソフト制限を無効にする唯一の方法は、軸のホームを解除することです。

スピンドルグループ最初の行のボタンは、スピンドルが回転する方向を選択します：反時計回り、停止、時計回り。 反時計回りは、ピンスピンドル.0.REVERSE が HAL ファイルにある場合にのみ表示されます（ネットトリック軸スピンドル.0.REVERSE の場合もあります）。 次の行のボタンは、回転速度を増減します。 3 行目のチェックボックスを使用すると、スピンドルブレーキをオンまたはリリースできます。 マシンの構成によっては、このグループのすべてのアイテムが表示されるとは限りません。 主軸スタートボタンを押すと、S 速度が 1 に設定されます。

クーラントグループ 2 つのボタンを使用すると、ミストクーラントとフラッドクーラントのオンとオフを切り替えることができます。 マシンの構成によっては、このグループのすべてのアイテムが表示されるとは限りません。

3.11.31.5 MDI

MDI を使用すると、G コードコマンドを手動で入力できます。 マシンの電源がオンになっていない場合、またはプログラムが実行されている場合、MDI コントロールは使用できません。

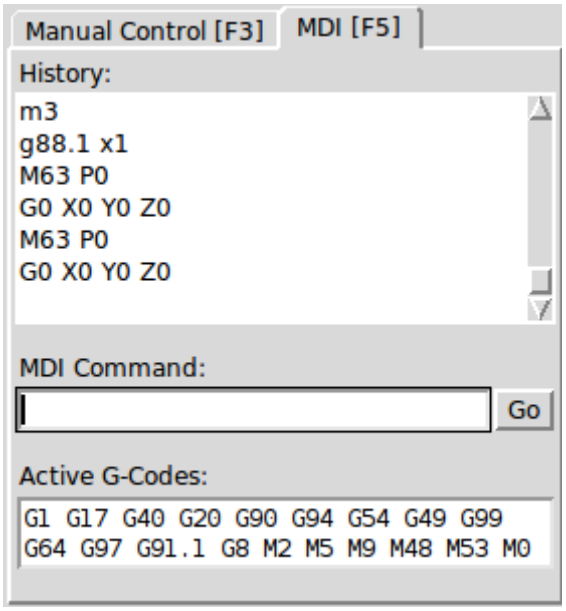


図 4-42

履歴-これは、このセッションの前半で入力された MDI コマンドを示しています。

MDI コマンド-これにより、実行する G コードコマンドを入力できます。 ENTER キーを押すか、[実行] をクリックして、コマンドを実行します。

アクティブな G コード-これは、インタープリターでアクティブなモーダルコードを示します。 たとえば、G54 は、入力されたすべての座標に G54 オフセットが適用されることを示します。 自動の場合、アクティブ G コードは、インタプリタが先読みした後のコードを表します。

3.11.31.6 フィードオーバーライド

このスライダーを動かすことにより、プログラムされた送り速度を変更することができます。 たとえば、プログラムが F60 を要求し、スライダーが 120%に設定されている場合、結果の送り速度は 72 になります。

3.11.31.7 スピンドル速度オーバーライド

このスライダーを動かすことにより、プログラムされたスピンドル速度を変更することができます。 たとえば、プログラムが S8000 を要求し、スライダーが 80%に設定されている場合、結果のスピンドル速度は 6400 になります。この項目は、HAL ピン SPLAND.0.SPEED-OUT が接続されている場合にのみ表示されます。

3.11.31.8 ジョグ送り速度

このスライダーを動かすことで、ジョグの速度を変えることができます。 たとえば、スライダーが 1 インチ/分に設定されている場合、.01 インチのジョグは約 0.6 秒、つまり 1/100 分で完了します。 左側（遅いジョグ）の近くでは値の間隔が狭く、右側（速いジョグ）の近くでは値の間隔がはるかに離れているため、最も重要なときに細かく制御して幅広いジョグ速度を実現できます。

回転軸のある機械では、2 番目のジョグ速度スライダーが表示されます。 このスライダーは、回転軸（A、B、C）のジョグ速度を設定します。

3.11.31.9 最大速度

このスライダーを動かすことで、最大速度を設定できます。 これにより、スピンドル同期移動を除くすべてのプログラムされた移動の最大速度が制限されます。

3.11.32 キーボード操作

AXIS のほとんどすべてのアクションは、キーボードを使用して実行できます。 キーボードショートカットの完全なリストは、AXIS クイックリファレンスにあります。これは、[ヘルプ]> [クイックリファレンス]を選択して表示できます。 MDI モードでは、ショートカットの多くは使用できません。

フィードオーバーライドキー手動モードの場合、フィードオーバーライドキーの動作は異なります。キー ’12345678 は、プログラムされている場合、軸を選択します。 3 軸の場合、 ’は軸 0 を選択し、1 は軸 1 を選択し、2 は軸 2 を選択します。残りの数字キーは引き続きフィードオーバーライドを設定します。 プログラム ’1234567890 を実行すると、フィードオーバーライドが 0%～100%に設定されます。

最も頻繁に使用されるキーボードショートカットを次の表に示します。

Keystroke	Action Taken	Mode
F1		
F2		
`, 1 .. 9, 0		
X, `		
Y, 1		
Z, 2		
A, 3		
I		
C		
Control-Home		
End		

Left, Right

Up, Down

Pg Up, Pg Dn

[,]

O

Control-R

R

P

S

ESC

Control-K

V

Shift-Left,Right

Shift-Up,Down

Shift-PgUp, PgDn

@

#

3.11.33 LINUXCNC ステータス（LINUXCNCTOP）

AXIS には、LINUXCNC の状態の詳細の一部を表示する LINUXCNCTOP と呼ばれるプログラムが含まれています。 このプログラムは、[マシン]> [LINUXCNC ステータスの表示]を呼び出すことで実行できます。

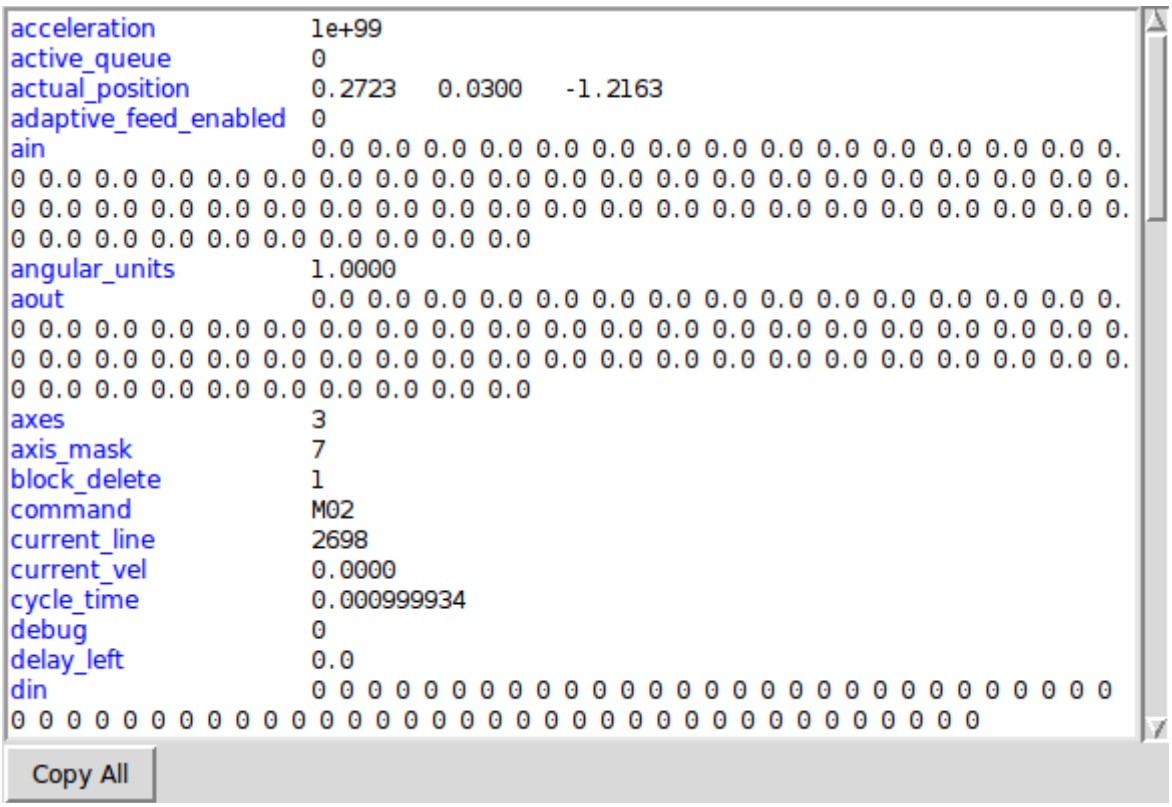


図 4-43

各アイテムの名前は左の列に表示されます。 現在の値が右の列に表示されます。 値が最近変更された場合は、赤い背景に表示されます。

3.11.34 MDI インターフェース

AXIS には、実行中の LinuxCNC セッションへの MDI コマンドのテキストモード入力を可能にする MDI と呼ばれるプログラムが含まれています。 ターミナルを開いて入力すると、このプログラムを実行できます

Mdi

実行されると、プロンプト MDI>が表示されます。 空白行を入力すると、機械の現在位置が表示されます。 コマンドを入力すると、LinuxCNC に送信されて実行されます。 これは MDI のサンプルセッションです。

$ mdi

MDI>

(0.0, 0.0, 0.0, 0.0, 0.0, 0.0)

MDI> G1 F5 X1

MDI>

(0.5928500000000374, 0.0, 0.0, 0.0, 0.0, 0.0)

MDI>

(1.0000000000000639, 0.0, 0.0, 0.0, 0.0, 0.0)

3.11.35 AXIS-REMOTE

AXIS には、実行中の AXIS に特定のコマンドを送信できる AXIS-REMOTE というプログラムが含まれています。 使用可能なコマンドは、AXIS-REMOTE --HELP を実行することで表示され、AXIS が実行されているかどうかの確認（--PING）、名前によるファイルのロード、現在ロードされているファイルの再ロード（--RELOAD）、および AXIS の終了（- 終了する）。

3.11.36 手動工具交換

LINUXCNC には、HAL_MANUALTOOLCHANGE と呼ばれるユーザースペース HAL コンポーネントが含まれています。これは、M6 コマンドが発行されたときに予期されるツールを示すウィンドウプロンプトを表示します。 OK ボタンを押すと、プログラムの実行が続行されます。

HAL_MANUALTOOLCHANGE コンポーネントには、物理ボタンに接続してツールの変更を完了し、ウィンドウプロンプト（HAL_MANUALTOOLCHANGE.CHANGE_BUTTON）を削除できるボタンの HAL ピンが含まれています。

HAL 構成ファイル LIB / HALLIB / AXIS_MANUALTOOLCHANGE.HAL は、このコンポーネントを使用するために必要な HAL コマンドを示しています。 HAL_MANUALTOOLCHANGE は、AXIS が GUI として使用されていない場合でも使用できます。 このコンポーネントは、プリセット可能なツールがあり、ツールテーブルを使用する場合に最も役立ちます。

Note

重要な注意：T <N>が発行された後、M6 の後の次のフィード移動まで、RAPIDS はプレビューに表示されません。 これは、ほとんどのユーザーにとって非常に混乱する可能性があります。 現在の工具交換プログラムでこの機能をオフにするには、T <N>の後に移動せずに G1 を実行します。

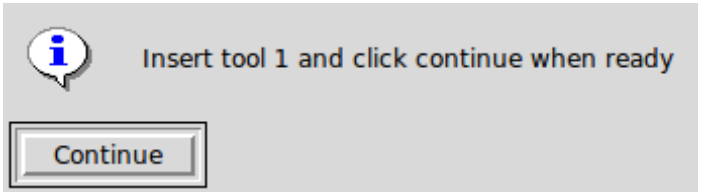


図 4-44

3.11.37 Python モジュール

AXIS には、他の人に役立つ可能性のあるいくつかの PYTHON モジュールが含まれています。 これらのモジュールの詳細については、PYDOC <モジュール名>を使用するか、ソースコードをお読みください。 これらのモジュールには次のものが含まれます。

EMC は、LINUXCNC コマンド、ステータス、およびエラーチャネルへのアクセスを提供します

GCODE は RS274NGC インタープリターへのアクセスを提供します

RS274 は、RS274NGC ファイルを操作するための追加ツールを提供します

HAL を使用すると、PYTHON で記述されたユーザースペース HAL コンポーネントを作成できます。

_TOGL は、TKINTER アプリケーションで使用できる OPENGL ウィジェットを提供します

MINIGL は、AXIS が使用する OPENGL のサブセットへのアクセスを提供します

これらのモジュールを独自のスクリプトで使用するには、モジュールが存在するディレクトリが PYTHON のモジュールパス上にあることを確認する必要があります。 インストールされているバージョンの LINUXCNC を実行すると、これは自動的に発生します。 インプレースで実行する場合、これは SCRIPTS / RIP-ENVIRONMENT を使用して実行できます。

3.11.38 旋盤モードで AXIS を使う

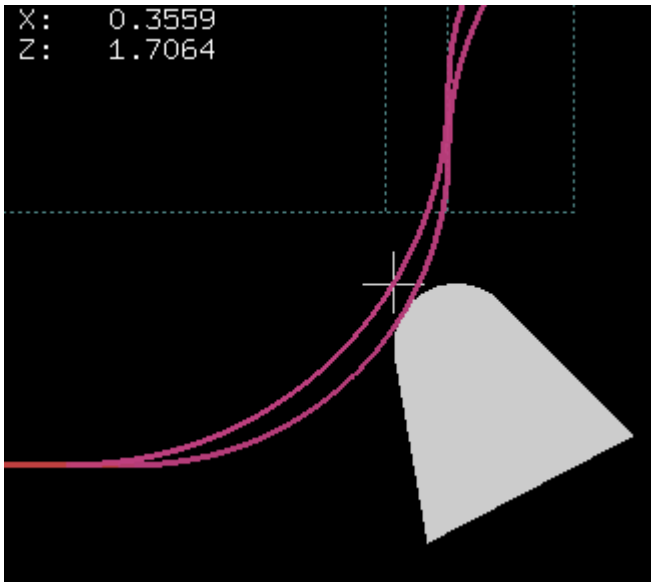


図 4-45

3.11.39 フォームカッティングモードで AXIS を使う

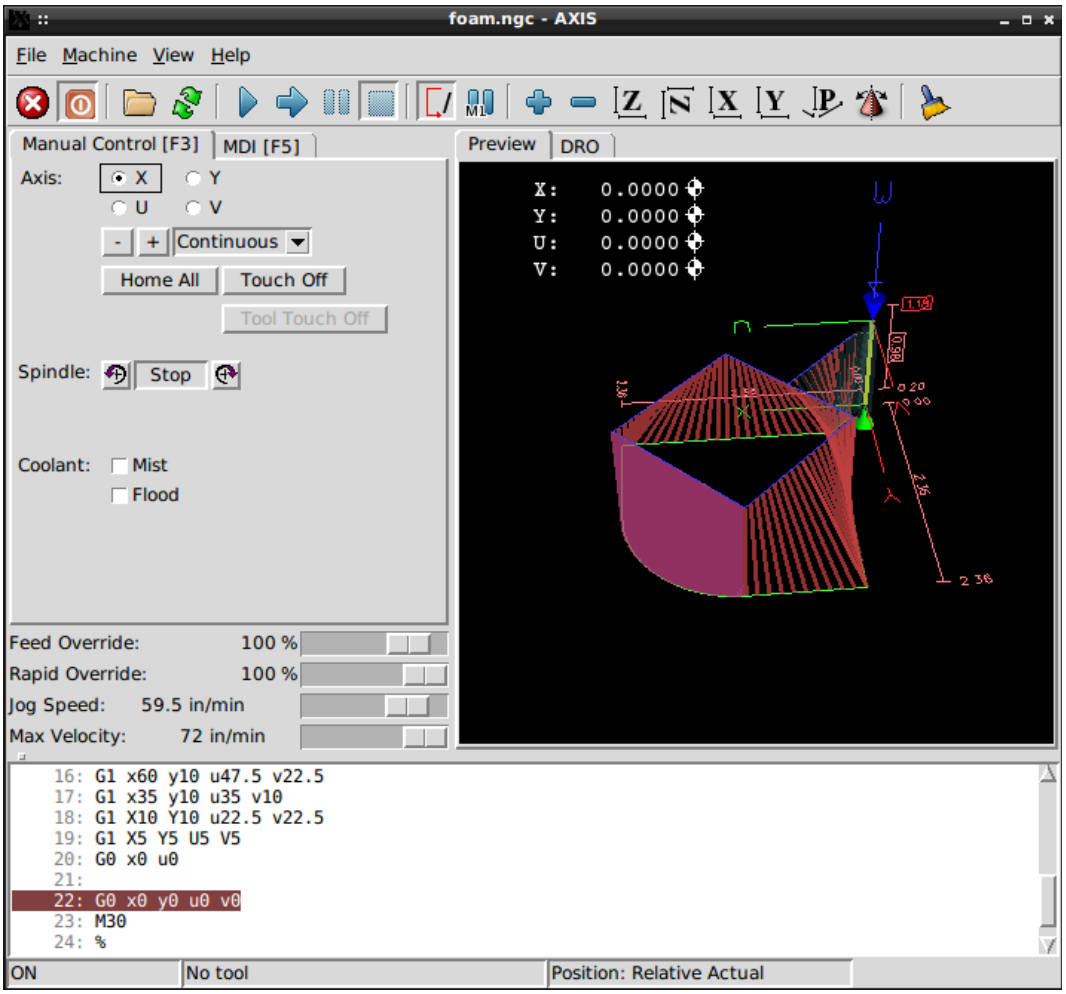


図 4-46

3.11.40 高度な構成

AXIS が起動すると、GUI の HAL ピンが作成され、INI ファイルの[HAL] POSTGUI_HALFILE で指定された HAL ファイルが実行されます。 使用できるポスト GUI ファイルは 1 つだけです。 GUIHAL ピンに接続するすべての HAL コマンドを GUI 後の HAL ファイルに配置します。

1.1.1.1 プログラムフィルター

AXIS には、ロードされたファイルをフィルタープログラムを介して送信する機能があります。 このフィルターは、ファイルが M2 で終わることを確認するような単純なものから、画像から G コードを生成するような複雑なものまで、任意のタスクを実行できます。

INI ファイルの[FILTER]セクションは、フィルターの動作を制御します。 まず、ファイルの種類ごとに、PROGRAM_EXTENSION 行を記述します。 次に、ファイルの種類ごとに実行するプログラムを指定します。 このプログラムには、最初の引数として入力ファイルの名前が指定されており、RS274NGC コードを標準出力に書き込む必要があります。 この出力は、テキスト領域に表示され、表示領域でプレビューされ、実行時に LINUXCNC によって実行されるものです。 次の行は、LINUXCNC に含まれているイメージから GCODE へのコンバーターのサポートを追加します。

[FILTER]

PROGRAM_EXTENSION = .png,.gif Greyscale Depth Image

png = image-to-gcode

gif = image-to-gcode

インタプリタを指定することもできます。

PROGRAM_EXTENSION = .py Python Script

py = python

このようにして、任意の PYTHON スクリプトを開くことができ、その出力は G コードとして扱われます。 そのようなサンプルスクリプトの 1 つは、NC_FILES /HOLECIRCLE.PY で入手できます。 このスクリプトは、円の円周に沿って一連の穴を開けるための G コードを作成します。

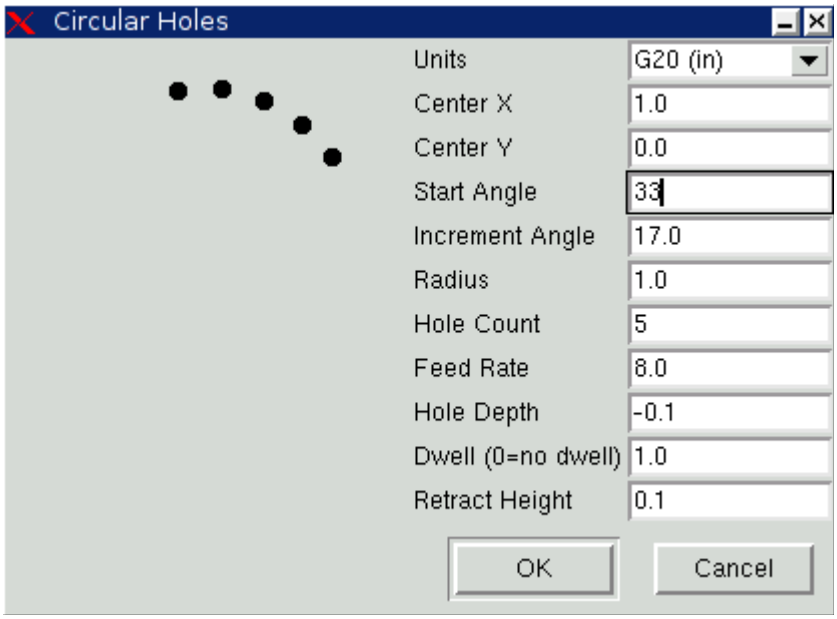


図 4-47

環境変数 AXIS_PROGRESS_BAR が設定されている場合、FILTER_PROGRESS =%の形式で STDERR に書き込まれる行

FILTER_PROGRESS =%D

AXIS プログレスバーを指定されたパーセンテージに設定します。 この機能は、長時間実行されるすべてのフィルターで使用する必要があります。

3.11.40.1 The X Resource Database

AXIS ユーザーインターフェイスのほとんどの要素の色は、X リソースデータベースを介してカスタマイズできます。 サンプルファイル AXIS_LIGHT_BACKGROUND は、バックプロットウィンドウの色を白い背景スキームの暗い線に変更し、表示領域の構成可能な項目の参照としても機能します。 サンプル

ファイル AXIS_BIG_DRO は、位置の読み取り値をより大きく変更します サイズフォント。 これらのファイルを使用するには：

```
xrdb -merge /usr/share/doc/emc2/axis_light_background

xrdb -merge /usr/share/doc/emc2/axis_big_dro
```

TK アプリケーションで構成できるその他の項目については、TK のマニュアルページを参照してください。

最新のデスクトップ環境では、AXIS に悪影響を与えるいくつかの設定が X リソースデータベースに自動的に行われるため、デフォルトではこれらの設定は無視されます。 X リソースデータベースアイテムが AXIS のデフォルトを上書きするようにするには、X リソースに次の行を含めます。

```
*Axis*optionLevel: widgetDefault
```

これにより、組み込みオプションがオプションレベル WIDGETDEFAULT で作成されるため、X リソース（レベル USERDEFAULT）がそれらをオーバーライドできます。

3.11.40.2 ~/.axisrc

存在する場合、～/ .AXISRC の内容は、AXISGUI が表示される直前に PYTHON ソースコードとして実行されます。 ～/ .AXISRC に記述される内容の詳細は、開発サイクル中に変更される可能性があります。

以下は、QUIT のキーボードショートカットとして CONTROL-Q を追加します。

```
root_window.bind("<Control-q>", "destroy .")

help2.append(("Control-Q", "Quit"))
```

以下は、「本当に終了しますか」ダイアログを停止します。

```
root_window.tk.call("wm","protocol",".","WM_DELETE_WINDOW","destroy .")
```

3.11.40.3 USER_COMMAND_FILE

構成固有の PYTHON ファイルは、INI ファイル設定[DISLAY] USER_COMMAND_FILE = FILENAME.PY で指定できます。 ～/ .AXISRC ファイルと同様に、このファイルは AXISGUI が表示される直前に供給されます。 このファイルは、ユーザーのホームディレクトリではなく、INI ファイル構成に固有のものです。 このファイルを指定すると、既存の～/ .AXISRC ファイルは無視されます。

3.11.40.4 USER_LIVE_UPDATE（）

軸の GUI には、LIVEPLOTTER クラスの UPDATE（）関数の終了時に実行される USER_LIVE_UPDATE（）という名前の NO-OP（プレースホルダー）関数が含まれています。 この関数は、～/ .AXISRCPYTHON スクリプトまたは[DISPLAY] USER_COMMAND_FILE PYTHON スクリプト内に実装

して、カスタムの定期的なアクションを実行できます。 この機能で実行できることの詳細は、軸の GUI の実装に依存し、開発サイクル中に変更される可能性があります。

3.11.40.5 USER_HAL_PINS（）

軸の GUI には、USER_HAL_PINS（）という名前の NO-OP（プレースホルダー）関数が含まれています。

.AXISRC ファイルが呼び出された直後、および GLADEVCP パネル/埋め込みタブが初期化される直前に実行されます。

この関数は、〜/ .AXISRCPYTHON スクリプトまたは[DISPLAY] USER_COMMAND_FILE PYTHON スクリプト内に実装して、AXISUI を使用するカスタム HAL ピンを作成できます。 プレフィックス。

COMP を HAL コンポーネントインスタンス参照として使用します。

HAL COMP.READY（）は、この関数が戻った直後に呼び出されます。

3.11.40.6 外部エディター

メニューオプション[ファイル]> [編集]。。。 およびファイル>ツールテーブルの編集。。。 INI セクション[DISPLAY]でエディターを定義した後に使用可能になります。 有用な値には、EDITOR = GEDIT および EDITOR = GNOME-TERMINAL -EVIM が含まれます。 詳細については、「INI 構成」の章の「表示」セクションを参照してください。

3.11.40.7 仮想コントロールパネル

AXIS は、右側のペインにカスタムの仮想コントロールパネルを表示できます。 ボタン、インジケーター、データ表示などをプログラムできます。 詳細については、PYVCP の章および GLADEVCP の章を参照してください。

3.11.40.8 プレビューコントロール

特別なコメントを G コードファイルに挿入して、AXIS のプレビューの動作を制御できます。 プレビューの描画を制限したい場合は、これらの特別なコメントを使用してください。（AXIS、HIDE）と（AXIS、SHOW）の間は、プレビュー中に描画されません。（AXIS、HIDE）と（AXIS、SHOW）は、（AXIS、HIDE）を最初にしてペアで使用する必要があります。（AXIS、STOP）の後は、プレビュー中に描画されません。

これらのコメントは、プレビュー表示を整理するのに役立ちます（たとえば、より大きな G コードファイルをデバッグしているときに、すでに正常に機能している特定の部分のプレビューを無効にすることができます）。

- （AXIS、HIDE）プレビューを停止します（最初にする必要があります）
- （AXIS、SHOW）プレビューを再開します（非表示に従う必要があります）
- （AXIS、STOP）ここからファイルの終わりまでプレビューを停止します。

- （AXIS、NOTIFY、THE_TEXT）THE_TEXT を情報表示として表示しますこの表示は、（DEBUG、MESSAGE）コメントが表示されていない場合の AXIS プレビューで役立ちます。

3.11.40.9 AXISUI ピン

AXIS と物理的なジョグホイールとの相互作用を改善するために、GUI で現在選択されている軸は、AXISUI.JOG.X のような名前のピンでも報告されます。 これらのピンの 1 つは一度に TRUE になり、残りは FALSE になります。 これらは、モーションのジョグ対応ピンを制御するためのものです。

AXISUI ピン AXIS には、[手動制御]タブで選択されているジョグラジオボタンを示す HAL ピンがあります。

Type	Dir	Name
bit	OUT	axisui.jog.x
bit	OUT	axisui.jog.y
bit	OUT	axisui.jog.z
bit	OUT	axisui.jog.a
bit	OUT	axisui.jog.b
bit	OUT	axisui.jog.c
bit	OUT	axisui.jog.u
bit	OUT	axisui.jog.v
bit	OUT	axisui.jog.w

軸には、[手動]タブで選択されたジョグ増分を示す HAL ピンがあります。

Type	Dir	Name
Float	OUT	axisui.jog.increment

AXIS には、エラーや情報のポップアップ通知をクリアするための HAL 入力ピンがあります。

Type	Dir	Name
Bit	IN	axisui.notifications-clear
Bit	IN	axisui.notifications-clear-error

Bit	IN	axisui.notifications-clear-info

AXIS には、一時停止/再開機能を無効/有効にするハーフ入力ピンがあります。

Type	Dir	Name
Bit	IN	axisui.resume-inhibit

3.11.41 AXIS カスタマイズのヒント

AXIS はかなり大きく、侵入が難しいコードベースです。これは、コードを安定させておくのに役立ちますが、カスタマイズが難しくなります。 ここでは、画面の動作やビジュアルを変更するためのコードスニペットを示します。 AXIS の内部コードは時々変更される可能性があることに注意してください。

これらのスニペットが引き続き機能することは保証されていません。調整が必要な場合があります。

1.1.1.1 更新機能

AXIS には USER_LIVE_UPDATE という名前の関数があり、AXIS がそれ自体を更新するたびに呼び出されます。 これを使用して、独自の機能を更新できます。

```
# continuous update function
def user_live_update():
print 'i am printed every update...'
```

3.11.41.1 閉じるダイアログを無効にする

```
# disable the do you want to close dialog
root_window.tk.call("wm","protocol",".","WM_DELETE_WINDOW","destroy .")
```

3.11.41.2 テキストフォントを変更する

```
# change the font
font = 'sans 11'
fname,fsize = font.split()
root_window.tk.call('font','configure','TkDefaultFont','-family',fname,'-size',fsize)

# redo the text in tabs so they resize for the new default font

root_window.tk.call('.pane.top.tabs','itemconfigure','manual','-text',' Manual - F3 ')
root_window.tk.call('.pane.top.tabs','itemconfigure','mdi','-text',' MDI - F5 ')
root_window.tk.call('.pane.top.right','itemconfigure','preview','-text',' Preview ')
root_window.tk.call('.pane.top.right','itemconfigure','numbers','-text',' DRO ')
```

```
# gcode font is independent

root_window.tk.call('.pane.bottom.t.text','configure','-foreground','blue')
#root_window.tk.call('.pane.bottom.t.text','configure','-foreground','blue','-font',font)
#root_window.tk.call('.pane.bottom.t.text','configure','-foreground','blue','-font',font,'-  -
height','12')
```

3.11.41.3 キーボードショートカットを使用して高速レートを変更する

```
# use control + ‘ or 1-0 as keyboard shortcuts for rapidrate and keep ‘ or 1-0 for feedrate
# also adds text to quick reference in help

help1.insert(10,("Control+ ‘,1..9,0", _("Set Rapid Override from 0% to 100%")),)

root_window.bind('<Control-Key-quoteleft>',lambda event: set_rapidrate(0))
root_window.bind('<Control-Key-1>',lambda event: set_rapidrate(10))
root_window.bind('<Control-Key-2>',lambda event: set_rapidrate(20))
root_window.bind('<Control-Key-3>',lambda event: set_rapidrate(30))
root_window.bind('<Control-Key-4>',lambda event: set_rapidrate(40))
root_window.bind('<Control-Key-5>',lambda event: set_rapidrate(50))
root_window.bind('<Control-Key-6>',lambda event: set_rapidrate(60))
root_window.bind('<Control-Key-7>',lambda event: set_rapidrate(70))
root_window.bind('<Control-Key-8>',lambda event: set_rapidrate(80))
root_window.bind('<Control-Key-9>',lambda event: set_rapidrate(90))
root_window.bind('<Control-Key-0>',lambda event: set_rapidrate(100))
root_window.bind('<Key-quoteleft>',lambda event: set_feedrate(0))
root_window.bind('<Key-1>',lambda event: set_feedrate(10))
root_window.bind('<Key-2>',lambda event: set_feedrate(20))
root_window.bind('<Key-3>',lambda event: set_feedrate(30))
root_window.bind('<Key-4>',lambda event: set_feedrate(40))
root_window.bind('<Key-5>',lambda event: set_feedrate(50))
root_window.bind('<Key-6>',lambda event: set_feedrate(60))
root_window.bind('<Key-7>',lambda event: set_feedrate(70)
root_window.bind('<Key-8>',lambda event: set_feedrate(80))
root_window.bind('<Key-9>',lambda event: set_feedrate(90))
root_window.bind('<Key-0>',lambda event: set_feedrate(100))
```

3.11.41.4 INI ファイルを読む

```
# read an ini file item
```

```
machine = inifile.find('EMC','MACHINE')
print 'machine name =',machine
```

3.11.41.5 LINUXCNC ステータスを読む

```
# linuxcnc status can be read from s.

print s.actual_position
print s.paused
```

3.11.41.6 現在のビューを変更する

```
# set the view of the preview
# valid views are view_x view_y view_y2 view_z view_z2 view_p

commands.set_view_z()
```

3.11.41.7 新しい AXISUIHAL ピンの作成

```
def user_hal_pins():
comp.newpin('my-new-in-pin', hal.HAL_BIT, hal.HAL_IN)
comp.ready()
```

3.11.41.8 新しい HAL コンポーネントとピンの作成

```
# create a component

mycomp = hal.component('my_component')
mycomp.newpin('idle-led',hal.HAL_BIT,hal.HAL_IN)
mycomp.newpin('pause-led',hal.HAL_BIT,hal.HAL_IN)
mycomp.ready()

# connect pins

hal.new_sig('idle-led',hal.HAL_BIT)
hal.connect('halui.program.is-idle','idle-led')
hal.connect('my_component.idle-led','idle-led')

# set a pin

hal.set_p('my_component.pause-led','1')
```

```
# get a pin 2,8+ branch

value = hal.get_value('halui.program.is-idle')
print 'value is a',type(value),'value of',value
```

3.11.41.9 HAL ピン付きのスイッチタブ

```
# hal pins from a GladeVCP panel will not be ready when user_live_update is run
# to read them you need to put them in a try/except block

# the following example assumes 5 HAL buttons in a GladeVCP panel used to switch
# the tabs in the Axis screen.
# button names are 'manual-tab', 'mdi-tab', 'preview-tab', 'dro-tab', 'user0-tab'
# the user_0 tab if it exists would be the first GladeVCP embedded tab

# for linuxCNC 2.8+ branch

def user_live_update():
try:
if hal.get_value('gladevcp.manual-tab'):
root_window.tk.call('.pane.top.tabs','raise','manual')
elif hal.get_value('gladevcp.mdi-tab'):
root_window.tk.call('.pane.top.tabs','raise','mdi')
elif hal.get_value('gladevcp.preview-tab'):
root_window.tk.call('.pane.top.right','raise','preview')
elif hal.get_value('gladevcp.numbers-tab'):
root_window.tk.call('.pane.top.right','raise','numbers')
elif hal.get_value('gladevcp.user0-tab'):
root_window.tk.call('.pane.top.right','raise','user_0')
except:
pass
```

3.11.41.10 GOTO ホームボタンを追加する

```
def goto_home(axis):
if s.interp_state == linuxcnc.INTERP_IDLE:
home = inifile.find('JOINT_' + str(inifile.find('TRAJ', 'COORDINATES').upper().  -
index(axis)), 'HOME')
mode = s.task_mode
if s.task_mode != linuxcnc.MODE_MDI:
```

```
c.mode(linuxcnc.MODE_MDI)
c.mdi('G53 G0 ' + axis + home)

# make a button to home y axis
root_window.tk.call('button','.pane.top.tabs.fmanual.homey','-text','Home Y','-
command','  -
goto_home Y','-height','2')

# place the button
root_window.tk.call('grid','.pane.top.tabs.fmanual.homey','-column','1','-row','7','-  -
columnspan','2','-padx','4','-sticky','w')
# any function called from tcl needs to be added to TclCommands
TclCommands.goto_home = goto_home
commands = TclCommands(root_window)
```

3.11.41.11 手動フレームにボタンを追加

```
# make a new button and put it in the manual frame

root_window.tk.call('button','.pane.top.tabs.fmanual.mybutton','-text','My Button','-  -
command','mybutton_clicked','-height','2')
root_window.tk.call('grid','.pane.top.tabs.fmanual.mybutton','-column','1','-row','6','-  -
columnspan','2','-padx','4','-sticky','w')

# the above send the "mybutton_clicked" command when clicked
# other options are to bind a press or release (or both) commands to the button
# these can be in addition to or instead of the clicked command
# if instead of then delete '-command','mybutton_clicked', from the first line

# Button-1 = left mouse button, 2 = right or 3 = middle

root_window.tk.call('bind','.pane.top.tabs.fmanual.mybutton','<Button-
1>','mybutton_pressed  -
')
root_window.tk.call('bind','.pane.top.tabs.fmanual.mybutton','<ButtonRelease-1>','  -
mybutton_released')

# functions called from the buttons

def mybutton_clicked():
print 'mybutton was clicked'
```

```
def mybutton_pressed():
print 'mybutton was pressed'
def mybutton_released():
print 'mybutton was released'

# any function called from tcl needs to be added to TclCommands

TclCommands.mybutton_clicked = mybutton_clicked
TclCommands.mybutton_pressed = mybutton_pressed
TclCommands.mybutton_released = mybutton_released
commands = TclCommands(root_window)
```

3.11.41.12 内部変数の読み取り

```
print vars.machine.get()
print vars.emcini.get()
```

```
jog_speed = DoubleVar
kinematics_type = IntVar
linuxcnctop_command = StringVar
machine = StringVar
max_aspeed = DoubleVar
max_maxvel = DoubleVar
max_queued_mdi_commands = IntVar
max_speed = DoubleVar
maxvel_speed = DoubleVar
mdi_command = StringVar
metric = IntVar
mist = BooleanVar
motion_mode             = IntVar
on_any_limit            = BooleanVar
optional_stop           = BooleanVar
override_limits     = BooleanVar
program_alpha       = IntVar
queued_mdi_commands = IntVar
rapidrate           = IntVar
rotate_mode             = BooleanVar
running_line            = IntVar
show_distance_to_go     = IntVar
show_extents            = IntVar
show_live_plot      = IntVar
show_machine_limits     = IntVar
```

```
show_machine_speed      = IntVar
show_program        = IntVar
show_pyvcppanel     = IntVar
show_rapids             = IntVar
show_tool           = IntVar
show_offsets            = IntVar
spindledir          = IntVar
spindlerate         = IntVar
task_mode           = IntVar
task_paused             = IntVar
task_state          = IntVar
taskfile            = StringVar
teleop_mode             = IntVar
tool                = StringVar
touch_off_system    = StringVar
trajcoordinates     = StringVar
tto_g11             = BooleanVar
view_type           = IntVar
```

3.11.41.13 ウィジェットを非表示

```
# hide a widget
# use 'grid' or 'pack' depending on how it was originally placed
root_window.tk.call('grid','forget','.pane.top.tabs.fmanual.jogf.zerohome.tooltouch')
```

3.11.41.14 ラベルを変更する

```
# change label of a widget
root_window.tk.call('setup_widget_accel','.pane.top.tabs.fmanual.mist','Downdraft')

# make sure it appears (only needed in this case if the mist button was hidden)
root_window.tk.call('grid','.pane.top.tabs.fmanual.mist','-column','1','-row','5','-  -
columnspan','2','-padx','4','-sticky','w')
```

3.11.41.15 既存のコマンドをリダイレクトする

```
# hijack an existing command
# originally the mist button calls the mist function

root_window.tk.call('.pane.top.tabs.fmanual.mist','configure','-
command','hijacked_command'  -
```

```
)

# The new function

def hijacked_command():
print 'hijacked mist command'

# add the function to TclCommands

TclCommands.hijacked_command = hijacked_command
commands = TclCommands(root_window)
```

3.11.41.16 DRO の色を変更する

```
# change dro screen

root_window.tk.call('.pane.top.right.fnumbers.text','configure','-foreground','green','-
background','black')
```

3.11.41.17 ツールバーのボタンを変更する

```
# change the toolbar buttons

buW = '3'
buH = '2'
boW = '3'

root_window.tk.call('.toolbar.machine_estop','configure','-image','','-text','ESTOP','-
width',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.machine_power','configure','-image','','-text','POWER','-
width',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.file_open','configure','-image','','-text','OPEN','-width',
buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.reload','configure','-image','','-text','RELOAD','-width',buW
,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_run','configure','-image','','-text','RUN','-width',
buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_step','configure','-image','','-text','STEP','-width'
,buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_pause','configure','-image','','-text','PAUSE','-
width',buW,'-height',buH,'-borderwidth',boW)
```

4.2 GMOCCAPY

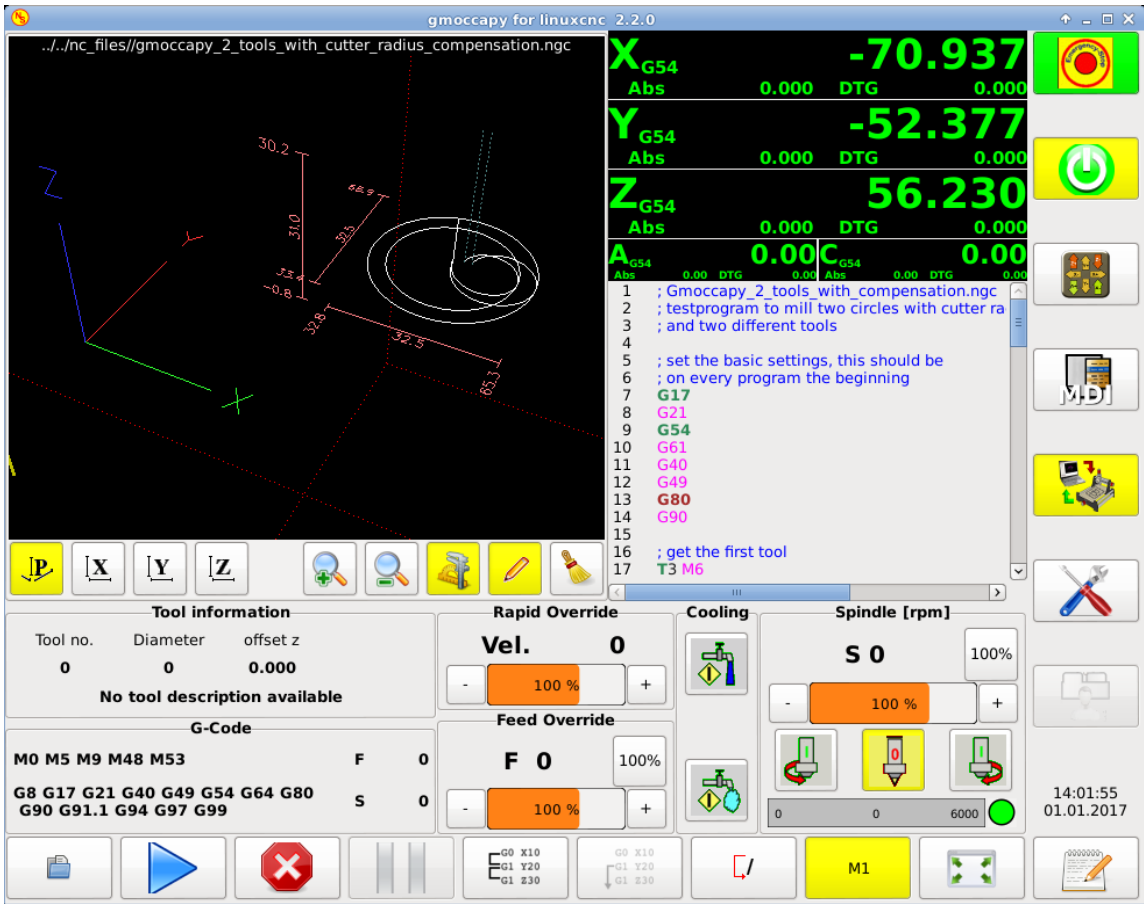
3.11.42 序章

GMOCCAPY は LinuxCNC 用の GUI であり、タッチスクリーンで使用するように設計されていますが、最も一般的なニーズに対応する HAL ピンを提供するため、マウスまたはハードウェアボタンと MPG ホイールを備えた通常の画面でも使用できます。 詳細については、以下をご覧ください。

gmoccapy は埋め込みタブとサイドパネルをサポートしているため、最大 9 軸を表示し、通常旋盤とバックツール旋盤の旋盤モードをサポートし、ほぼすべてのニーズに適合させることができます。 その良い例として、gmoccapy_plasma を参照してください。* gmoccapy 3 は、最大 9 つの軸と 9 つのジョイントをサポートします。 gmoccapy 3 は、LinuxCNC でのジョイント/軸の変更をサポートするようにコードが変更されているため、2.7 または 2.6 ブランチでは機能しません。

統合された仮想キーボード（オンボードまたはマッチボックスキーボード）をサポートしているため、ハードウェアキーボードやマウスは必要ありませんが、そのハードウェアで使用することもできます。 Gmoccapy は、ファイルを編集せずに GUI のほとんどの設定を構成するための個別の設定ページを提供します。

対応するファイルが linuxcnc.po ファイルから分離されているため、GMOCCAPY は非常に簡単にローカライズできます。したがって、不要なものを翻訳する必要はありません。 ファイルは/ src / po / gmoccapy に配置されます。 gmoccapy.pot ファイルを fr.po のようなものにコピーし、そのファイルを gtranslator または poedit で翻訳するだけです。 再構築後、好みの言語で GUI を取得しました。 翻訳を公開してください。公式パッケージに含めて、他のユーザーに公開することができます。 現時点では、英語、ドイツ語、スペイン語、ポーランド語、セルビア語、ハンガリー語で利用できます。 nieson@web.de で、もっと多くの言語を紹介するのを手伝ってください。 サポートが必要な場合は、遠慮なくお問い合わせください。



3.11.43 要件

Gmoccapy 3 は、Debian Jessie、Debian Stretch、および LinuxCNC マスターと 2.8 リリースの MINT18 でテストされています。 LinuxCNC のジョイント/軸の変更を完全にサポートしているため、スカラ、ロボット、または軸よりもジョイントが多いその他の構成の GUI として適しています。 したがって、ガントリー構成もサポートします。 他のバージョンを使用している場合は、LinuxCNC フォーラム、ドイツの CNC Ecke フォーラム、または LinuxCNC ユーザーのメーリングリストで問題や解決策についてお知らせください。

gmoccapy の最小画面解像度は、サイドパネルなしで使用した場合、979 x 750 ピクセルであるため、すべての標準画面に収まるはずです。 1024x748 の最小解像度の画面を使用することをお勧めします

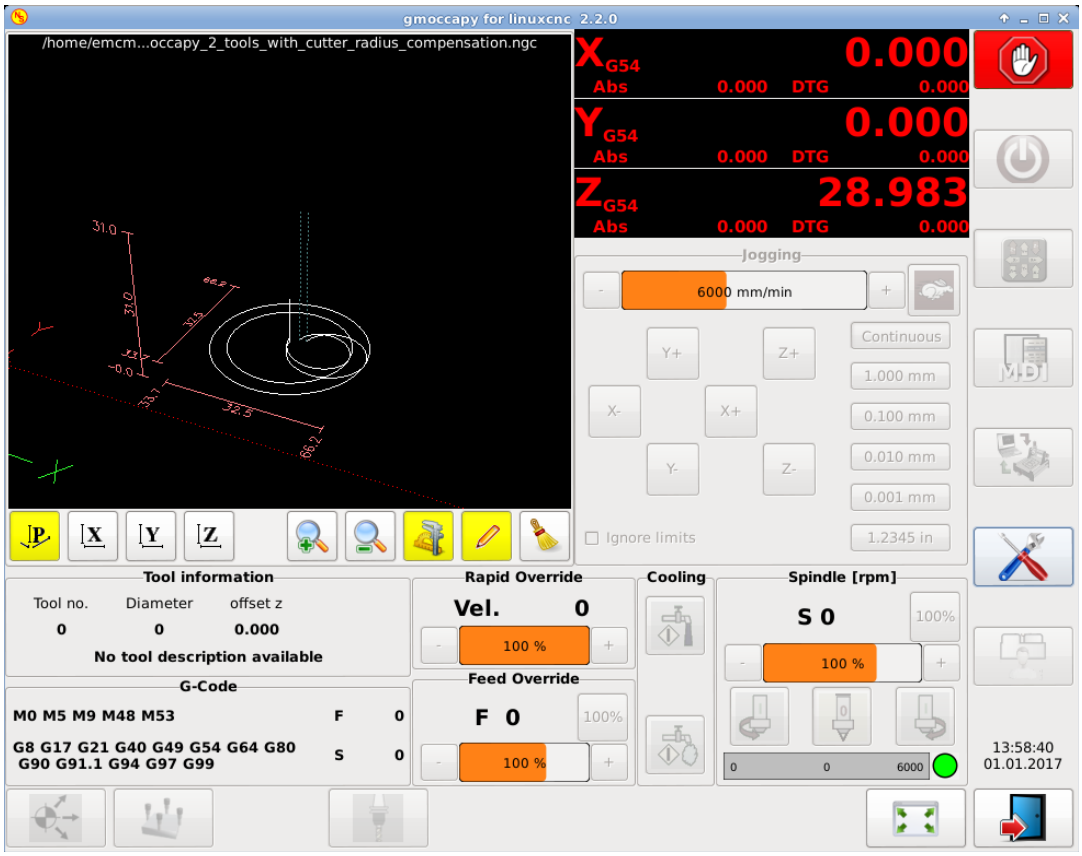
3.11.44 gmoccapy を取得する方法

LinuxCNC 2.8 以降、gmoccapy3 が標準インストールに含まれています。 したがって、PC を制御しているときに gmoccapy を取得する最も簡単な方法は、ISO を取得し、CD / DVD / USB-Stick からインストールすることです。

通常の deb パッケージでアップデートを受け取ります。

Gmoccapy 3 は、実際の 2.8 およびマスターリリースにのみ含まれています。

次のような画面が表示されます。構成によってデザインが異なる場合があります。



3.11.45 基本構成

gmoccapy を実行するためだけに構成することは実際にはそれほど多くありませんが、GUI のすべての機能を使用する場合は、注意が必要な点がいくつかあります。

基本を示すために、多くのシミュレーション構成（INI ファイル）が含まれています。

* gmoccapy.ini
* gmoccapy_4_axis.ini
* lathe_configs/gmoccapy_lathe.ini
* lathe_configs/gmoccapy_lathe_imperial.ini
* gmoccapy_left_panel.ini
* gmoccapy_right_panel.ini
* gmoccapy_messages.ini
* gmoccapy_pendant.ini
* gmoccapy_sim_hardware_button.ini
* gmoccapy_tool_sensor.ini
* gmoccapy_with_user_tabs.ini
* gmoccapy_XYZAB.ini
* gmoccapy_XYZAC.ini
* gmoccapy_XYZCW.ini
* gmoccapy-JA/Gantry/gantry_mm.ini
* gmoccapy-JA/scara/scara.ini

```
* gmoccapy-JA/table-rotary-tilting/xyzac-trt.ini
* and a lot more . . .
```

名前は、さまざまな INI ファイルの主な目的を説明する必要があります。

マシンの既存の構成を使用する場合は、このドキュメントに従って INI を編集するだけです。

重要

MACROS を使用する場合は、以下に説明するように、マクロまたはサブルーチンフォルダーへのパスを設定することを忘れないでください。

それでは、INI ファイルと、マシンで gmoccapy を使用するために含める必要のあるものを詳しく見てみましょう。

1.1.1.1 ディスプレイセクション

```
[DISPLAY]
DISPLAY = gmoccapy
PREFERENCE_FILE_PATH = gmoccapy_preferences
MAX_FEED_OVERRIDE = 1.5
MAX_SPINDLE_OVERRIDE = 1.2
MIN_SPINDLE_OVERRIDE = 0.5
LATHE = 1
BACK_TOOL_LATHE = 1
PROGRAM_PREFIX = ../../nc_files/
```

最も重要な部分は、LinuxCNC に gmoccapy を使用するように指示し、[DISPLAY]セクションを編集することです。

[DISPLAY]

DISPLAY = gmoccapy

PREFERENCE_FILE_PATH = gmoccapy_preferences

gmoccapy 3 は、次のコマンドラインオプションをサポートしています。

- -usermode：設定すると、セットアップボタンが無効になるため、通常の機械オペレーターは機械の設定を編集できません*
- -logo <ロゴファイルへのパス>：指定すると、ロゴは手動モードでジョグボタンタブを非表示にします。これは、ジョギングと増分選択用のハードウェアボタンを備えたマシンでのみ役立ちます。

行 PREFERENCE_FILE_PATH は、使用する設定ファイルの場所と名前を示します。 ほとんどの場合、この行は必要ありません。テーマ、DRO ユニット、色、キーボード設定など、GUI の設定を保存するために gmoccapy によって使用されます。詳細については、設定ページを参照してください。

---

note

パスまたはファイルが指定されていない場合、gmoccapy はデフォルトの<your_machinename> .pref として使用します。INI ファイルにマシン名が指定されていない場合、gmoccapy.pref を使用します。ファイルは config ディレクトリに保存されるため、設定は行われません。 複数の構成を使用する場合は、混合してください。 複数のマシンで 1 つのファイルのみを使用する場合は、INI に PREFERENCE_FILE_PATH を含める必要があります。

```
MAX_FEED_OVERRIDE = 1.5
```

最大フィードオーバーライドを設定します。与えられた例では、フィードを 150％オーバーライドできます。

Note

値が指定されていない場合は、1.0 に設定されます

```
MAX_SPINDLE_OVERRIDE = 1.2
MIN_SPINDLE_OVERRIDE = 0.5
```

50％から 120％の制限内でスピンドルオーバーライドを変更できます。

Note

値が指定されていない場合、MAX は 1.0 に設定され、MIN は 0.1 に設定されます。

```
LATHE = 1
BACK_TOOL_LATHE = 1
```

最初の行は、旋盤を制御するための画面レイアウトを設定します。

2 行目はオプションで、バックツール旋盤に必要な方法で X 軸を切り替えます。 また、キーボードショートカットは異なる方法で反応します。 gmoccapy を使用すると、追加の軸を使用して旋盤を構成できるため、旋盤に XZCW 構成を使用することもできます。

Tip

旋盤固有のセクションも参照してください

- PROGRAM_PREFIX = ../../nc_files/

linuxcnc / gmoccapy に ngc ファイルを探す場所を指示するエントリです。

Note

指定されていない場合、Gmoccapy は次の順序で ngc ファイルを検索します：linuxcnc / nc_files、次にユーザーのホームディレクトリ。

タブとサイドパネルの構成

axis、touchy、gscreen でできるように、埋め込みプログラムを gmoccapy に追加できます。

DISPLAY セクションの INI ファイルに数行を含めると、すべてが gmoccapy によって自動的に実行されます。

空き地パネルを使用したことがない場合は、優れたドキュメントを読むことをお勧めします。 グレイド VCP

例

```
EMBED_TAB_NAME = DRO
EMBED_TAB_LOCATION = ntb_user_tabs
EMBED_TAB_COMMAND = gladevcp -x {XID} dro.glade
EMBED_TAB_NAME = Second user tab
EMBED_TAB_LOCATION = ntb_preview
EMBED_TAB_COMMAND = gladevcp -x {XID} vcp_box.glade
```

注意しなければならないのは、すべてのタブまたはサイドパネルに上記の 3 行を含めることだけです。

- EMBED_TAB_NAME =タブまたはサイドパネルの名前を表します。使用する名前はユーザー次第ですが、存在する必要があります。
- EMBED_TAB_LOCATION =プログラムが GUI に配置される場所です。

有効な値は次のとおりです。

- ntb_user_tabs（メインタブとして、画面全体をカバー）」
- ntb_preview（プレビュー側のタブとして）」
- box_left（左側、画面の完全な高さ）
- box_right（右側、通常の画面とボタンリストの間）
- box_coolant_and_spindle（クーラントとスピンドルフレームを非表示にし、ここにグレイドファイルを導入します）
- box_cooling（冷却フレームを非表示にし、空き地ファイルを導入します）
- box_spindle（スピンドルフレームを非表示にし、グレイドファイルを導入します）
- box_vel_info（ベロシティフレームを非表示にし、グレイドファイルを導入します）
- box_custom_1（vel_frame の左側にある空き地ファイルを紹介します）
- box_custom_2（cooling_frame の左側にある空き地ファイルを導入します）
- box_custom_3（spland_frame の左側にある空き地ファイルを紹介します）
- box_custom_4（spland_frame の右側に空き地ファイルを導入します）

違いを確認するには、含まれているさまざまな INI ファイルを参照してください

- EMBED_TAB_COMMAND =実行するコマンド、つまり

```
gladevcp -x {XID} dro.glade
```

上記の場所に dro.glade というカスタムグレイドファイルが含まれています。ファイルは、マシンの config フォルダーに配置する必要があります。

```
gladevcp h_buttonlist.glade
```

h_buttonlist.glade という新しいユーザーウィンドウを開くだけです。違いに注意してください。これはスタンドアロンであり、gmoccapy ウィンドウから独立して移動できます。

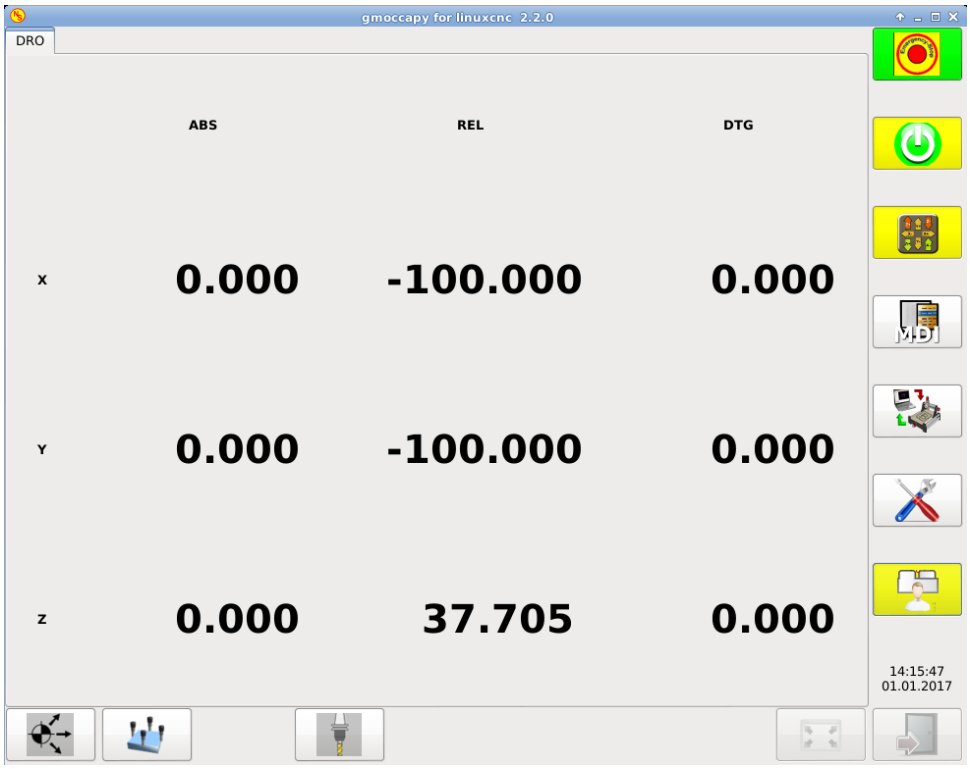
```
gladevcp -c gladevcp -u hitcounter.py -H manual-example.hal manual-example.ui
```

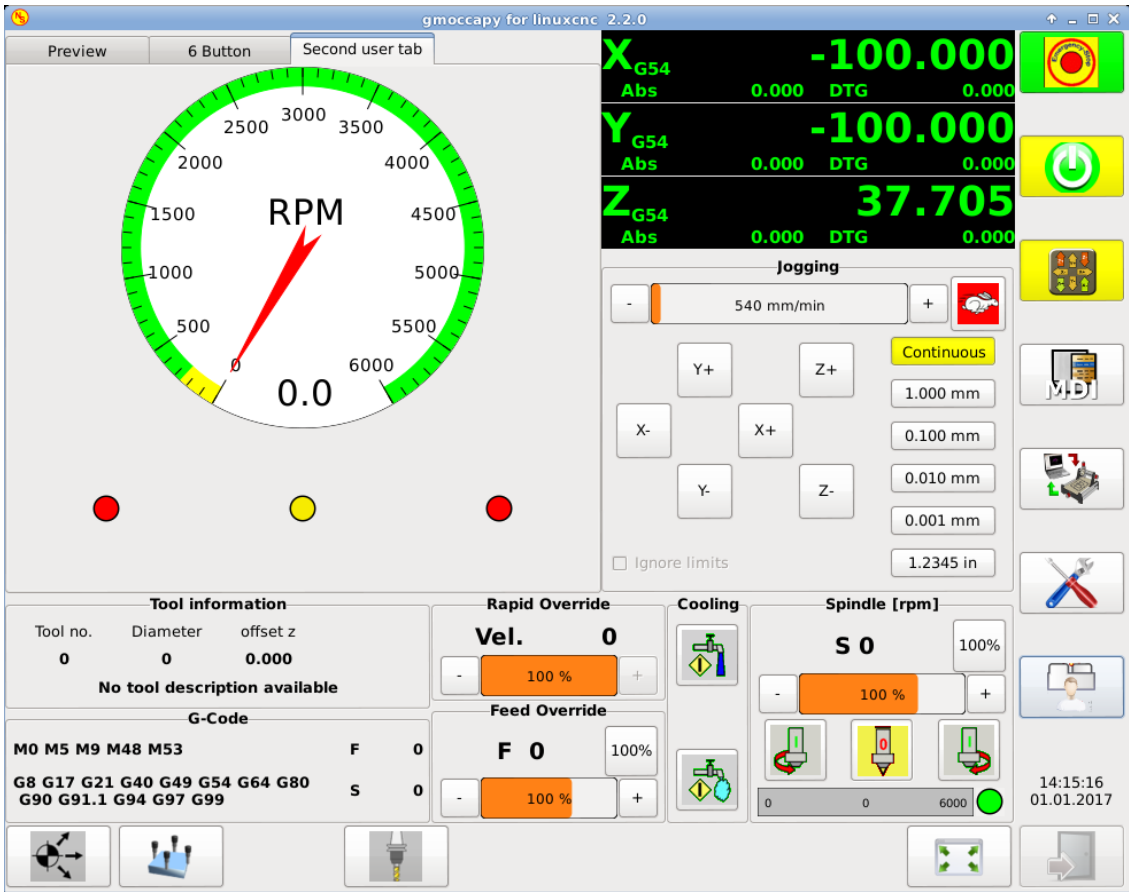
パネル manual-example.ui を追加し、カスタム python ハンドラー hitcounter.py を含め、manual-example.hal に従ってパネルを実現した後にすべての接続を確立します。

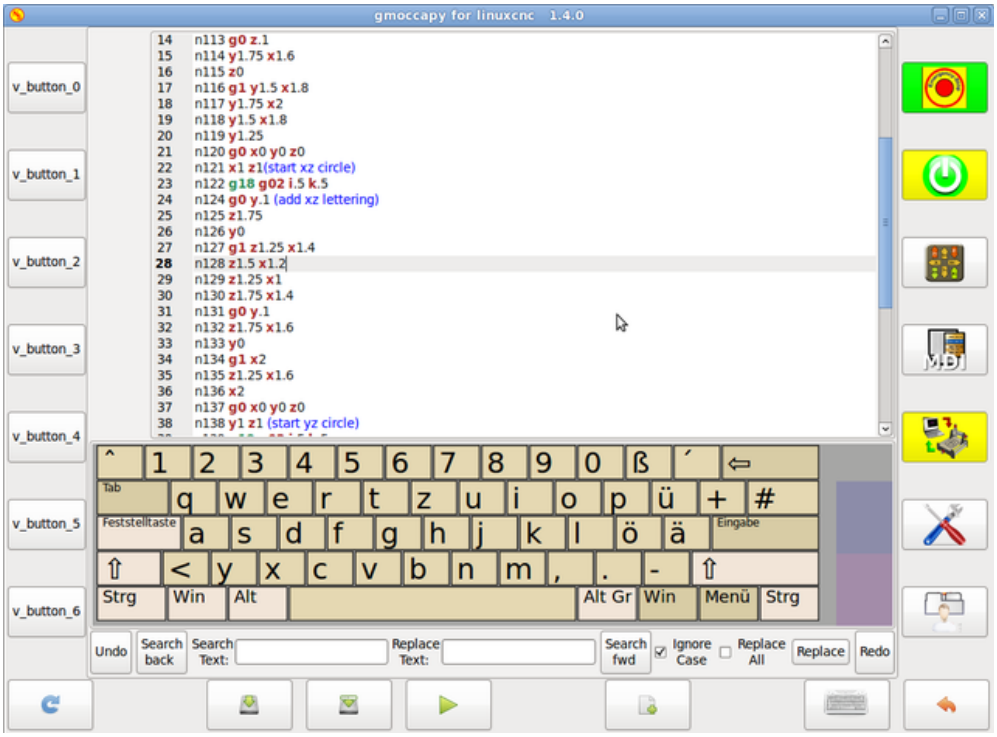
Note

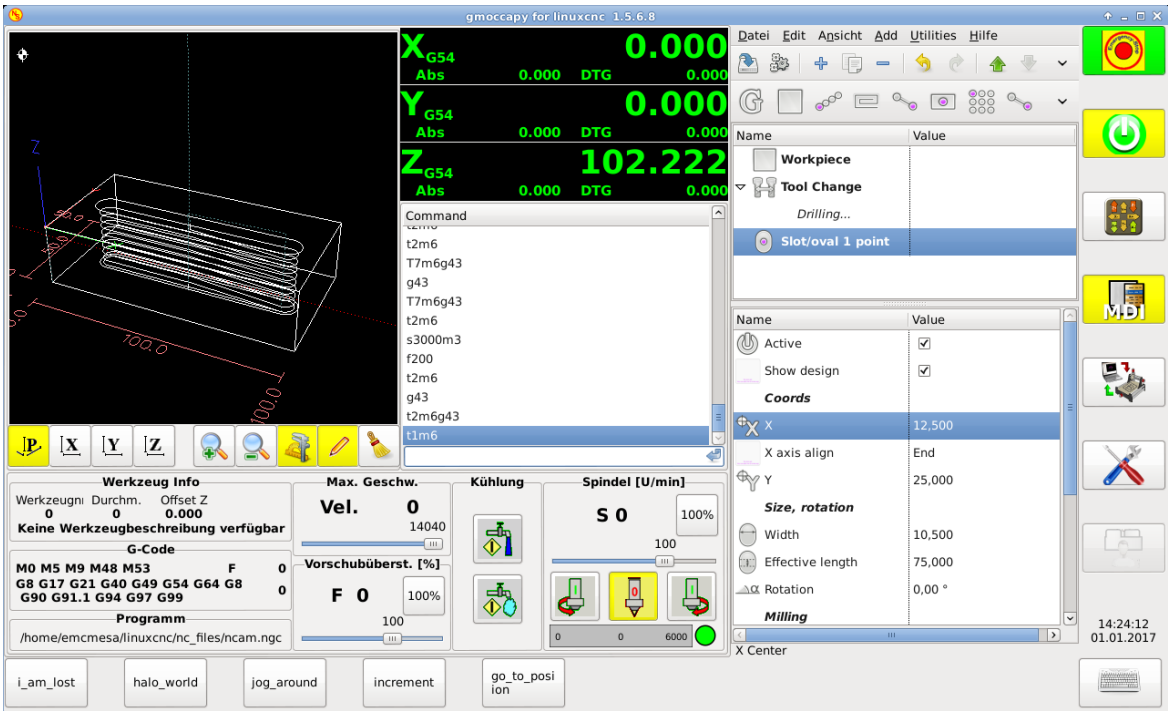
カスタムグレイドパネルに hal 接続を行う場合は、EMBEDDED_TAB_COMMAND 行で指定された hal ファイルで行う必要があります。そうしないと、hal ピンが存在しないというエラーが発生する可能性があります。これは、hal をロードする競合状態が原因です。 ファイル。 gmoccapy hal ピンへの接続は、GUI を実現する前にこのピンが存在しないため、INI ファイルで指定された postguihal ファイルで行う必要があります。

ここではいくつかの例を示します。




gmoccapy for linuxcnc 2.2.0
Preview
6 Button
Second user tab
RPM
0.0
X G54 -100.000
Abs 0.000 DTG 0.000
Y G54 -100.000
Abs 0.000 DTG 0.000
Z G54 37.705
Abs 0.000 DTG 0.000
Jogging
540 mm/min
Continuous
1.000 mm
0.100 mm
0.010 mm
0.001 mm
1.2345 in
Ignore limits
Tool information
Tool no. Diameter offset z
0 0 0.000
No tool description available
G-Code
M0 M5 M9 M48 M53 F 0
G8 G17 G21 G40 G49 G54 G64 G80 G90 G91.1 G94 G97 G99 S 0
Rapid Override
Vel. 0
100 %
Feed Override
F 0
100%
100 %
Cooling
Spindle [rpm]
S 0
100%
100 %
14:15:16
01.01.2017


gmoccapy for linuxcnc 1.4.0
v_button_0
v_button_1
v_button_2
v_button_3
v_button_4
v_button_5
v_button_6
Undo
Search back
Search Text:
Replace Text:
Search fwd
Ignore Case
Replace All
Replace
Redo



ユーザー作成メッセージの構成 Gmoccapy には、hal 駆動のユーザーメッセージを作成する機能があります。 それらを使用するには、INI ファイルの[DISPLAY]セクションにいくつかの行を導入する必要があります。

メッセージがパンゴマークアップ言語をサポートする 3 つのユーザーポップアップメッセージダイアログを設定する方法は次のとおりです。 マークアップ言語の詳細については、PangoMarkup をご覧ください。

```
MESSAGE_TEXT = The text to be displayed, may be pango markup formatted
MESSAGE_TYPE = "status" , "okdialog" , "yesnodialog"
MESSAGE_PINNAME = is the name of the hal pin group to be created
```

- ステータス：gmoccapy のメッセージングシステムを使用して、メッセージをポップアップウィンドウとして表示します
- okdialog：メッセージダイアログにフォーカスを保持し、「待機中」の Hal_PinOUT をアクティブにします。 メッセージを閉じると、待機中のピンがリセットされます
- yesnodialog：メッセージダイアログにフォーカスを保持し、「待機中」の Hal_Pin ビット OUT をアクティブにし、「応答」の Hal_Pin ビット出力へのアクセスも提供します。ユーザーが[OK]をクリックすると、このピンは 1 を保持します。 メッセージを閉じると待機中のピンがリセットされます応答 HalPin は、ダイアログが再度呼び出されるまで 1 のままです。

Example

```
MESSAGE_TEXT = This is a <span background="#ff0000" foreground="#ffffff">
info-message</span> test
```

```
MESSAGE_TYPE = status
MESSAGE_PINNAME = statustest
MESSAGE_TEXT = This is a yes no dialog test
MESSAGE_TYPE = yesnodialog
MESSAGE_PINNAME = yesnodialog
MESSAGE_TEXT = Text can be <small>small</small>, <big>big</big>, <b>bold</b <i>italic</
i>,  -
and even be <span color="red">colored</span>.
MESSAGE_TYPE = okdialog
MESSAGE_PINNAME = okdialog
```

これらの特定の halpin 規則は、ユーザーメッセージの halpin セクションにあります。

3.11.45.1 RS274NGC セクション

```
[RS274NGC]
SUBROUTINE_PATH = macros
```

マクロやその他のサブルーチンを検索するためのパスを設定します。 複数のサブルーチンパスを使用する場合は、「：」で区切ってください。

3.11.45.2 マクロセクション

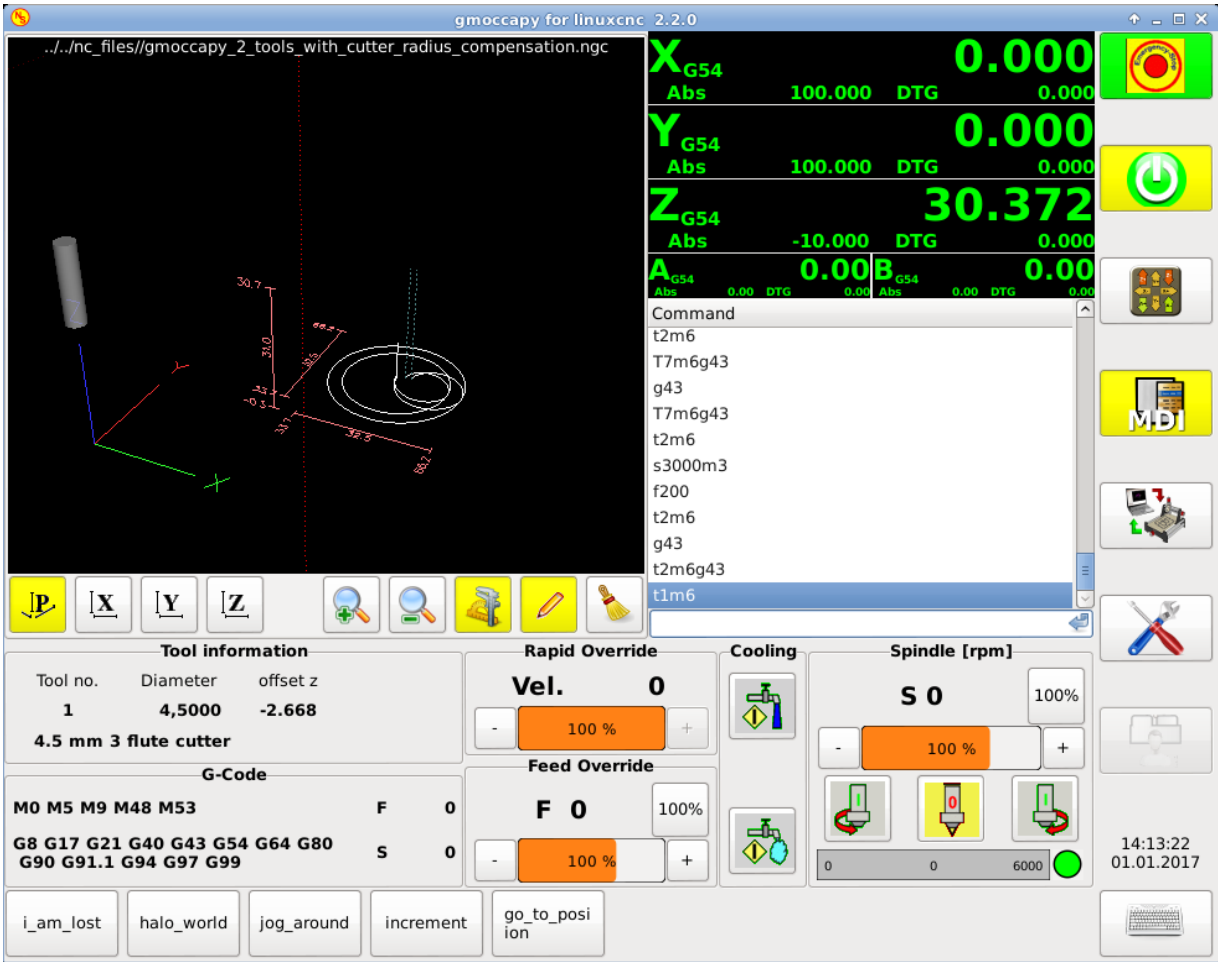
touchy の方法と同様に、gmoccapy にマクロを追加できます。 マクロは ngc ファイルに他なりません。 ボタンを 1 つ押すだけで、MDI モードで完全な CNC プログラムを実行できます。 これを行うには、次のようなセクションを追加する必要があります。

```
[MACROS]
MACRO = i_am_lost
MACRO = hello_world
MACRO = jog_around
MACRO = increment xinc yinc
MACRO = go_to_position X-pos Y-pos Z-pos
```

これにより、MDI ボタンリストに 5 つのマクロが追加されます。

Note

最大 16 個のマクロが GUI に表示されるため、スペース上の理由から、矢印をクリックしてページを切り替え、非表示のマクロボタンを表示する必要がある場合があります。 INI ファイルにさらに多くを配置することはエラーではありません。 マクロボタンは、INI エントリの順序で表示されます。



ファイルの名前は、MACRO 行で指定された名前と完全に同じである必要があります。 したがって、マクロ i_am_lost はファイル i_am_lost.ngc を呼び出します。

マクロ NGC ファイルはいくつかのルールに従う必要があります。

- ファイルの名前は、マクロ行に記載されている名前と完全に同じである必要がありますが、拡張子は ngc（大文字と小文字が区別されます）のみです。
- ファイルには、次のようなサブルーチンが含まれている必要があります。O<i_am_lost> sub、sub の名前は、マクロの名前と正確に一致する必要があります（大文字と小文字が区別されます）。
- ファイルは、endsub O <i_am_lost> endsub とそれに続く M2 コマンドで終了する必要があります
- ファイルは、RS274NGC セクションの INI ファイルで指定されたフォルダーに配置する必要があります（RS274NGC を参照）。

sub と endsub の間のコードは、対応するマクロボタンを押すことで実行されます。

Note

サンプルマクロは、gmoccapysim フォルダーにある macros フォルダーにあります。 複数のサブルーチンパスを指定した場合、それらは指定されたパスの順序で検索されます。 最初に見つかったファイルが使用されます。

Gmoccapy は、次のようなパラメーターを要求するマクロも受け入れます。

```
go_to_position X-pos Y-pos Z-pos
```

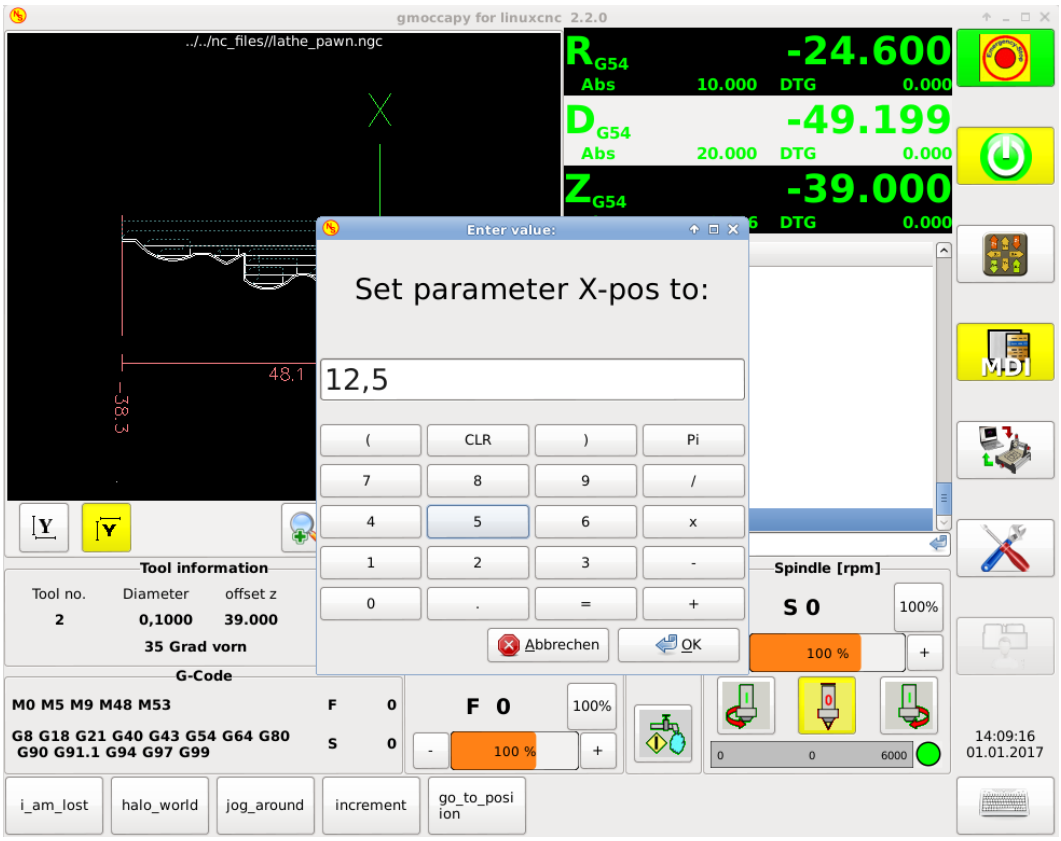
パラメータはスペースで区切る必要があります。 これにより、次の内容のファイル go_to_position.ngc が呼び出されます。

```
; Test file go to position
; will jog the machine to a given position
O<go_to_position> sub
G17
G21
G54
G61
G40
G49
G80
G90
;#1 = <X-Pos>
;#2 = <Y-Pos>
;#3 = <Z-Pos>
(DBG, Will now move machine to X = #1 , Y = #2 , Z = #3)
G0 X #1 Y #2 Z #3
O<go_to_position> endsub
M2
```

マクロの実行ボタンを押すと、X-pos Y-pos Z-pos の値を入力するように求められ、すべての値が指定されている場合にのみマクロが実行されます。

Note

移動せずにマクロを使用する場合は、既知の問題の注記も参照してください。



3.11.45.3 TRAJ セクション

```
DEFAULT_LINEAR_VELOCITY = 85.0
MAX_VELOCITY = 230.000
```

機械の最大速度とデフォルトのジョグ速度を設定します。

Note

DEFAULT_LINEAR_VELOCITY が指定されていない場合、MAX_VELOCITY の半分が使用されます。 その値も指定されていない場合、デフォルトで 180 になります。MAX_VELOCITY が指定されていない場合、デフォルトで 600 になります。

3.11.46 HAL ピン

gmoccapy は、ハードウェアデバイスに反応できるように、いくつかの hal ピンをエクスポートします。 目標は、ツールショップで完全にほとんどマウスやキーボードなしで操作できる GUI を取得することです。

Note

GUI を完全にロードする前に gmoccapy ピンを使用できないため、postgui.hal ファイル内の gmoccapy ピンへのすべての接続を行う必要があります。 複数の postguihal ファイルを呼び出すことができるため、構成のデバッグが容易になります。 postgui_call_list.hal ファイルを使用するだけです。 パネルは GUI の後

にロードされるため、ユーザーパネルへの接続は別の hal ファイルで行う必要があります。 詳細については、タブとサイドパネルを参照してください。

1.1.1.1 右と下のボタンリスト

画面には 2 つのメインボタンリストがあり、1 つは右側に、もう 1 つは下部にあります。 右利きのボタンは操作中に変更されませんが、下のボタンリストは頻繁に変更されます。 ボタンは、「0」で始まり、上から下、左から右に数えられます。

Note

gmoccapy2 のピン名が変更され、順序が改善されました。

- gmoccapy.v-button.button-0
- gmoccapy.v-button.button-1
- gmoccapy.v-button.button-2
- gmoccapy.v-button.button-3
- gmoccapy.v-button.button-4
- gmoccapy.v-button.button-5
- gmoccapy.v-button.button-6

下部（水平）ボタンは次のとおりです。

- gmoccapy.h-button.button-0
- gmoccapy.h-button.button-1
- gmoccapy.h-button.button-2
- gmoccapy.h-button.button-3
- gmoccapy.h-button.button-4
- gmoccapy.h-button.button-5
- gmoccapy.h-button.button-6
- gmoccapy.h-button.button-7
- gmoccapy.h-button.button-8
- gmoccapy.h-button.button-9

下のリストのボタンはモードやその他の影響に応じて変化するため、ハードウェアボタンはさまざまな機能をアクティブにします。これは完全に gmoccapy によって行われるため、機能を半分に切り替える必要はありません。 3 軸 XYZ マシンの場合、hal ピンは次のように反応します。

手動モードの場合：

- gmoccapy.h-button.button-0 == open homing button
- gmoccapy.h-button.button-1 == open touch off stuff
- gmoccapy.h-button.button-2 ==
- gmoccapy.h-button.button-3 == open tool dialogs
- gmoccapy.h-button.button-4 ==
- gmoccapy.h-button.button-5 ==
- gmoccapy.h-button.button-6 ==
- gmoccapy.h-button.button-7 ==
- gmoccapy.h-button.button-8 == full-size preview
- gmoccapy.h-button.button-9 == exit if machine is off, otherwise no reaction

mdi モードの場合：

- gmoccapy.h-button.button-0 == macro_0 or nothing
- gmoccapy.h-button.button-1 == macro_1 or nothing
- gmoccapy.h-button.button-2 == macro_2 or nothing
- gmoccapy.h-button.button-3 == macro_3 or nothing
- gmoccapy.h-button.button-4 == macro_4 or nothing
- gmoccapy.h-button.button-5 == macro_5 or nothing
- gmoccapy.h-button.button-6 == macro_6 or nothing
- gmoccapy.h-button.button-7 == macro_7 or nothing
- gmoccapy.h-button.button-8 == macro_8 or switch page to additional macros
- gmoccapy.h-button.button-9 == open keyboard or abort if macro is running

自動モードで

- gmoccapy.h-button.button-0 == open file
- gmoccapy.h-button.button-1 == reload program
- gmoccapy.h-button.button-2 == run
- gmoccapy.h-button.button-3 == stop
- gmoccapy.h-button.button-4 == pause
- gmoccapy.h-button.button-5 == step by step

- gmoccapy.h-button.button-6 == run from line if enabled in settings, otherwise Nothing
- gmoccapy.h-button.button-7 == optional blocks
- gmoccapy.h-button.button-8 == full-size preview
- gmoccapy.h-button.button-9 == edit code

設定モードの場合：

- gmoccapy.h-button.button-0 == delete MDI history
- gmoccapy.h-button.button-1 ==
- gmoccapy.h-button.button-2 ==
- gmoccapy.h-button.button-3 ==
- gmoccapy.h-button.button-4 == open classic ladder
- gmoccapy.h-button.button-5 == open hal scope
- gmoccapy.h-button.button-6 == open hal status
- gmoccapy.h-button.button-7 == open hal meter
- gmoccapy.h-button.button-8 == open hal calibration
- gmoccapy.h-button.button-9 == open hal show

ホーミングモードの場合：

- gmoccapy.h-button.button-0 ==
- gmoccapy.h-button.button-1 == home all
- gmoccapy.h-button.button-2 ==
- gmoccapy.h-button.button-3 == home x
- gmoccapy.h-button.button-4 == home y
- gmoccapy.h-button.button-5 == home z
- gmoccapy.h-button.button-6 ==
- gmoccapy.h-button.button-7 ==
- gmoccapy.h-button.button-8 == unhome all
- gmoccapy.h-button.button-9 == back

in touch off mode:

- gmoccapy.h-button.button-0 == edit offsets
- gmoccapy.h-button.button-1 == touch X

- gmoccapy.h-button.button-2 == touch Y
- gmoccapy.h-button.button-3 == touch Z
- gmoccapy.h-button.button-4 ==
- gmoccapy.h-button.button-5 ==
- gmoccapy.h-button.button-6 == zero G92
- gmoccapy.h-button.button-7 ==
- gmoccapy.h-button.button-8 == set selected
- gmoccapy.h-button.button-9 == back

ツールモードの場合：

- gmoccapy.h-button.button-0 == delete tool(s)
- gmoccapy.h-button.button-1 == new tool
- gmoccapy.h-button.button-2 == reload tool table
- gmoccapy.h-button.button-3 == apply changes
- gmoccapy.h-button.button-4 == change tool by number T? M6
- gmoccapy.h-button.button-5 == set tool by number without change M61 Q?
- gmoccapy.h-button.button-6 == change tool to the selected one
- gmoccapy.h-button.button-7 ==
- gmoccapy.h-button.button-8 == touch of tool in Z
- gmoccapy.h-button.button-9 == back

編集モードの場合：

- gmoccapy.h-button.button-0 ==
- gmoccapy.h-button.button-1 == reload file
- gmoccapy.h-button.button-2 == save
- gmoccapy.h-button.button-3 == save as
- gmoccapy.h-button.button-3 == save as
- gmoccapy.h-button.button-5 ==
- gmoccapy.h-button.button-6 == new file
- gmoccapy.h-button.button-7 ==
- gmoccapy.h-button.button-8 == show keyboard

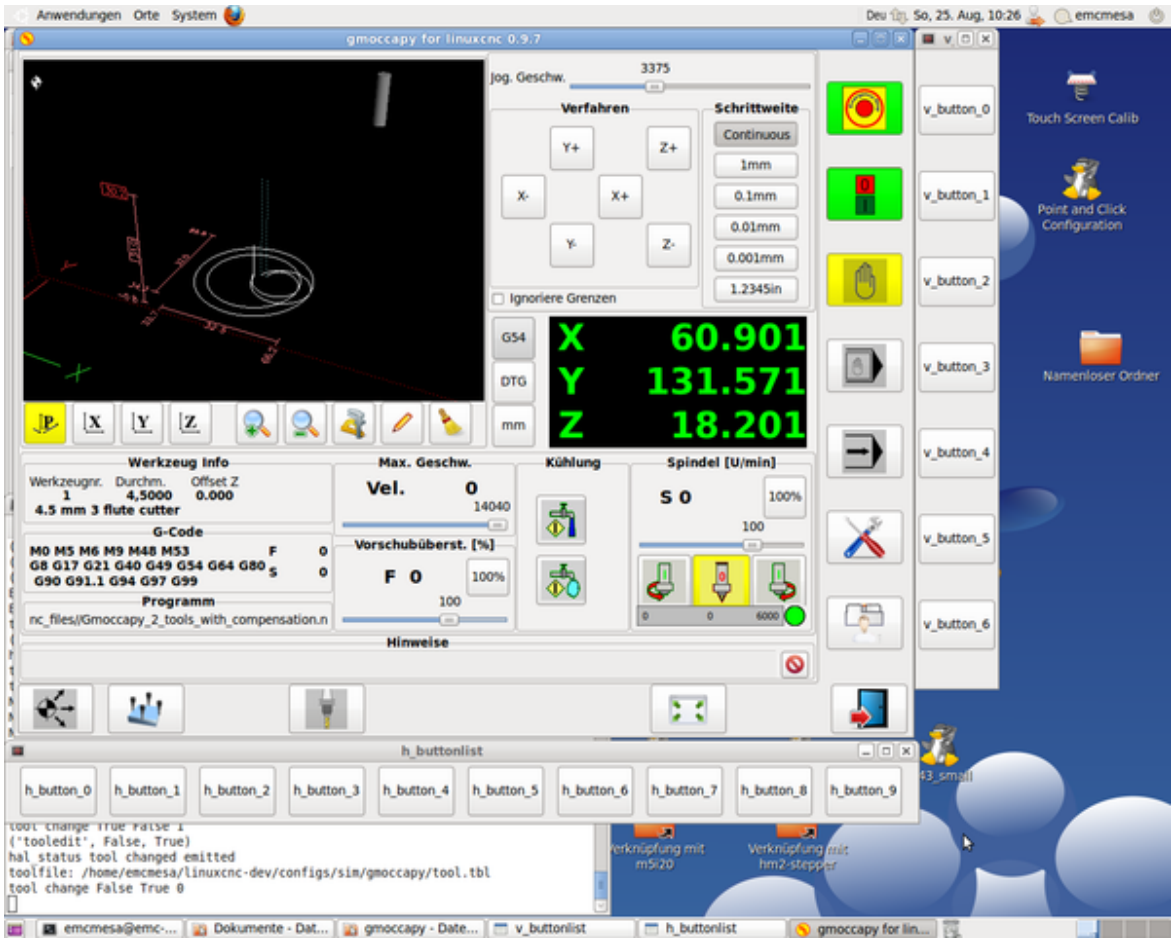
- gmoccapy.h-button.button-8 == show keyboard

ファイル選択モードの場合：

- gmoccapy.h-button.button-0 == go to home directory
- gmoccapy.h-button.button-1 == one directory level up
- gmoccapy.h-button.button-2 ==
- gmoccapy.h-button.button-2 ==
- gmoccapy.h-button.button-2 ==
- gmoccapy.h-button.button-5 == jump to directory as set in settings
- gmoccapy.h-button.button-6 ==
- gmoccapy.h-button.button-7 == select / ENTER
- gmoccapy.h-button.button-8 ==
- gmoccapy.h-button.button-9 == back

つまり、わずか 10hal ピンで 67 の反応があります。

これらのピンは、タッチパネルなしで画面を使用したり、パネルの周囲にハードウェアボタンを配置して過度の使用から画面を保護したりできるようになっています。



3.11.46.1 ベロシティとオーバーライド

gmoccapy のすべてのスライダーは、ハードウェアエンコーダーまたはハードウェアポテンショメーターに接続できます。

Note

gmoccapy 3 の場合、新しいコントロールが実装されたため、hal ピン名が変更されました。高速オーバーライドが実装されたため、最大速度は存在しなくなりました。 この変更は、多くのユーザーが要求したとおりに行われました。

エンコーダーを接続するために、次のピンがエクスポートされます。

- gmoccapy.jog.jog-velocity.counts = HAL_S32 Jog velocity
- gmoccapy.jog.jog-velocity.count-enable = HAL_BIT Must be True, to enable counts
- gmoccapy.feed.feed-override.counts = HAL_S32 feed override
- gmoccapy.feed.feed-override.count-enable = HAL_BIT Must be True, to enable counts
- gmoccapy.feed.reset-feed-override = HAL_BIT reset the feed override to 100 %
- gmoccapy.spindle.spindle-override.counts = HAL_S32 spindle override

- gmoccapy.spindle.spindle-override.count-enable = HAL_BIT Must be True, to enable counts
- gmoccapy.spindle.reset-spindle-override = HAL_BIT reset the spindle override to 100 %
- gmoccapy.rapid.rapid-override.counts = HAL_S32 Maximal Velocity of the machine
- gmoccapy.rapid.rapid-override.count-enable = HAL_BIT Must be True, to enable counts

ポテンショメータを接続するには、次の hal ピンを使用します。

- gmoccapy.jog.jog-velocity.direct-value = HAL_FLOAT To adjust the jog velocity slider
- gmoccapy.jog.jog-velocity.direct-value = HAL_FLOAT To adjust the jog velocity slider
- gmoccapy.feed.feed-override.direct-value = HAL_FLOAT To adjust the feed override slider
- gmoccapy.feed.feed-override.analog-enable = HAL_BIT Must be True, to allow analog inputs
- gmoccapy.spindle.spindle-override.direct-value = HAL_FLOAT To adjust the spindle override slider
- gmoccapy.spindle.spindle-override.analog-enable = HAL_BIT Must be True, to allow analog inputs
- gmoccapy.rapid.rapid-override.direct-value = HAL_FLOAT To adjust the max velocity slider
- gmoccapy.rapid.rapid-override.analog-enable = HAL_BIT Must be True, to allow analog inputs

さらに、gmoccapy 3 は、モーメンタリスイッチで新しいスライダーウィジェットを制御するための追加の hal ピンを提供します。 増加または減少の速度の値は、空き地ファイルで設定する必要があります。 将来のリリースでは、設定ページに統合される予定です。

- gmoccapy.spc_jog_vel.increase = HAL_BIT IN as long as True the value of the slider will increase
- gmoccapy.spc_jog_vel.decrease = HAL_BIT IN as long as True the value of the slider will decrease
- gmoccapy.spc_jog_vel.scale = HAL_FLOAT IN A value to scale the output value (Handy to change units/min to units/sec
- gmoccapy.spc_jog_vel.value = HAL_FLOAT OUT value of the widget
- gmoccapy.spc_jog_vel.scaled-value = HAL_FLOAT OUT scaled value of the widget
- gmoccapy.spc_feed.increase = HAL_BIT IN as long as True the value of the slider will increase
- gmoccapy.spc_feed.decrease = HAL_BIT IN as long as True the value of the slider will decrease

- gmoccapy.spc_feed.scale = HAL_FLOAT IN A value to scale the output value (Handy to change units/min to units/sec
- gmoccapy.spc_feed.value = HAL_FLOAT OUT value of the widget
- gmoccapy.spc_feed.scaled-value = HAL_FLOAT OUT scaled value of the widget
- gmoccapy.spc_spindle.increase = HAL_BIT IN as long as True the value of the slider will increase
- gmoccapy.spc_spindle.decrease = HAL_BIT IN as long as True the value of the slider will decrease
- gmoccapy.spc_spindle.scale = HAL_FLOAT IN A value to scale the output value (Handy to change units/min to units/sec
- gmoccapy.spc_spindle.value = HAL_FLOAT OUT value of the widget
- gmoccapy.spc_spindle.scaled-value = HAL_FLOAT OUT scaled value of the widget
- gmoccapy.spc_rapid.increase = HAL_BIT IN as long as True the value of the slider will increase
- gmoccapy.spc_rapid.decrease = HAL_BIT IN as long as True the value of the slider will decrease
- gmoccapy.spc_rapid.scale = HAL_FLOAT IN A value to scale the output value (Handy to change units/min to units/sec
- gmoccapy.spc_rapid.value = HAL_FLOAT OUT value of the widget
- gmoccapy.spc_rapid.scaled-value = HAL_FLOAT OUT scaled value of the widget

フロートピンは、スライダー値を設定するパーセンテージ値である 0.0～1.0 の値を受け入れます。

[警告]両方の接続タイプを使用する場合は、同じスライダーを両方のピンに接続しないでください。2 つの間の影響はテストされていません。 異なるスライダーを 1 つまたは他の hal 接続タイプに接続できます。

[重要]ジョグ速度はタートルボタンの状態によって異なり、モード（タートルまたはウサギ）によってスライダーのスケールが異なることに注意してください。 詳細については、ジョグ速度とタートルジョグハルピンもご覧ください。

Example

```
Spindle Override Min Value = 20 %
Spindle Override Max Value = 120 %
gmoccapy.analog-enable = 1
gmoccapy.spindle-override-value = 0.25
```

value to set = Min Value + (Max Value - Min Value) * gmoccapy.spindle-override-value
value to set = 20 + (120 - 20) * 0.25
value to set = 45 %

3.11.46.2 ジョグハルピン

INI ファイルで指定されたすべての軸にはジョグプラスピンとジョグマイナスピンがあるため、ハードウェアモーメンタリスイッチを使用して軸をジョグできます。

Note

この hal ピンの名前は gmoccapy2 で変更されました

標準の XYZ 構成では、次の hal ピンが使用可能になります。

- gmoccapy.jog.axis.jog-x-plus
- gmoccapy.jog.axis.jog-x-minus
- gmoccapy.jog.axis.jog-y-plus
- gmoccapy.jog.axis.jog-y-minus
- gmoccapy.jog.axis.jog-z-plus
- gmoccapy.jog.axis.jog-z-minus

4 軸 INI ファイルを使用する場合、2 つの追加ピンがあります

- gmoccapy.jog.jog-<your fourth axis letter >-plus
- gmoccapy.jog.jog-<your fourth axis letter >-minus

「C」軸の場合、次のように表示されます。

gmoccapy.jog.axis.jog-c-plus

gmoccapy.jog.axis.jog-c-minus

3.11.46.3 ジョグ速度とタートル-ジョグ HAL ピン

ジョグ速度は、対応するスライダーで選択できます。 タートルボタン（ウサギまたはタートルを表示するボタン）を切り替えると、スライダーのスケールが変更されます。 ボタンが表示されていない場合は、設定ページで無効になっている可能性があります。 ボタンにウサギのアイコンが表示されている場合、スケールは最小から最大のマシン速度です。 カメが表示されている場合、スケールはデフォルトで最大速度の 1/20 にしか達しません。 使用する仕切りは設定ページで設定できます。

したがって、タッチスクリーンを使用すると、より小さな速度を選択する方がはるかに簡単です。

gmoccapy は、カメとウサギのジョギングを切り替えるための hal ピンを提供します

- gmoccapy.jog.turtle-jog (Hal Bit In)

3.11.46.4 ジョグインクリメント HAL ピン

ジョグの増分は hal ピンを介して選択できるため、選択ハードウェアスイッチを使用して、使用する増分を選択できます。 INI ファイルで指定された増分には最大 10 個の hal ピンがあります。INI ファイルでさらに増分を指定すると、表示されないため、GUI から到達できなくなります。

hal に 6 つの増分がある場合、7 つのピンが得られます。jog-inc-0 は変更できず、継続的なジョギングを表します。

- gmoccapy.jog.jog-inc-0
- gmoccapy.jog.jog-inc-1
- gmoccapy.jog.jog-inc-2
- gmoccapy.jog.jog-inc-3
- gmoccapy.jog.jog-inc-4
- gmoccapy.jog.jog-inc-5
- gmoccapy.jog.jog-inc-6

gmoccapy は、選択したジョグ増分を出力するためのハーフパイントも提供します

- gmoccapy.jog.jog-increment

3.11.46.5 ハードウェアロック解除ピン

キースイッチを使用して設定ページのロックを解除できるようにするには、次のピンがエクスポートされます。

- gmoccapy.unlock-settings

ピンが高い場合、設定ページのロックが解除されます。 このピンを使用するには、設定ページでアクティブにする必要があります。

3.11.46.6 エラーピン

- gmoccapy.error
- gmoccapy.delete-message

gmoccapy.error は、エラーを示すためのビットアウトピンであるため、ライトが点灯したり、マシンが停止したりする可能性があります。 ピン gmoccapy.delete-message でリセットされます。 gmoccapy.delete-message は、最初のエラーを削除し、最後のエラーがクリアされた後、gmoccapy.error ピンを False にリセットします。

Note

メッセージまたはユーザー情報は gmoccapy.error ピンには影響しませんが、エラーが表示されない場合、gmoccapy.delete-message ピンは最後のメッセージを削除します。

3.11.46.7 ユーザーが作成したメッセージ HAL ピン

gmoccapy は、3 つの異なるユーザーメッセージを使用して、外部エラーに反応する場合があります。すべて HAL_BIT ピンです。

状態

- gmoccapy.messages.statustest

YesNo dialog

- gmoccapy.messages.yesnodialog
- gmoccapy.messages.yesnodialog-waiting
- gmoccapy.messages.yesnodialog-responce

Ok dialog

- gmoccapy.messages.okdialog
- gmoccapy.messages.okdialog-waiting

ユーザーが作成したメッセージを追加するには、[DISPLAY]セクションの INI ファイルにメッセージを追加する必要があります。 ここにいくつかの例があります。

```
MESSAGE_BOLDTEXT = LUBE SYSTEM FAULT
MESSAGE_TEXT = LUBE FAULT
MESSAGE_TYPE = okdialog
MESSAGE_PINNAME = lube-fault
MESSAGE_BOLDTEXT = NONE
MESSAGE_TEXT = X SHEAR PIN BROKEN
MESSAGE_TYPE = status
MESSAGE_PINNAME = pin
```

新しいピンを接続して入力するには、postguiHAL ファイルでこれを行う必要があります。 これは、HAL ファイルの他の場所の入力に信号が接続されている接続の例です。

```
net gmoccapy-lube-fault gmoccapy.messages.lube-fault
net gmoccapy-lube-fault-waiting gmoccapy.messages.lube-fault-waiting
net gmoccapy-pin gmoccapy.messages.pin
```

HAL ファイルと net コマンドの詳細については、HAL の基本を参照してください。

3.11.46.8 スピンドルフィードバックピン

スピンドルフィードバック用の 2 つのピンがあります

- gmoccapy.spindle_feedback_bar
- gmoccapy.spindle_at_speed_led

gmoccapy.spindle_feedback_bar は、スピンドル速度を示すフロート入力を受け入れます。
gmoccapy.spindle_at_speed_led は、スピンドルが高速の場合に主導される GUI を点灯させるためのビットインピンです。

3.11.46.9 プログラムの進捗情報を示すピン

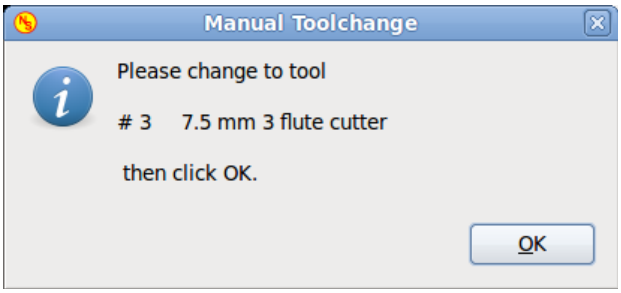
プログラムの進行状況に関する情報を提供する 3 つのピンがあります

- gmoccapy.program.lengthHAL_S32 プログラムの合計行数を示す
- gmoccapy.program.current-line HAL_S32 は、プログラムの現在の作業行を示します
- gmoccapy.program.progress HAL_FLOAT は、プログラムの進行状況をパーセンテージで示します

サブルーチンや大規模なリマッププロシージャを使用している場合は、値があまり正確でない可能性があります。また、ループによって異なる値が発生します。

3.11.46.10 ツール関連のピン

ツール変更ピンこのピンは、gmoccapy の内部ツール変更ダイアログを使用するために提供されています。これは、軸からわかるものと似ていますが、いくつかの変更が加えられているため、ツール番号 3 に変更するメッセージだけでなく、そのツールの説明も表示されます。 7.5 mm3 フルートカッターのように。 情報はツールテーブルから取得されるため、何を表示するかはユーザー次第です。



- gmoccapy.toolchange-numberHAL_S32 変更するツールの番号
- gmoccapy.toolchange-changeHAL_BIT ツールを変更する必要があることを示します
- gmoccapy.toolchange-変更された HAL_BIT 料金が変更されたことを示します

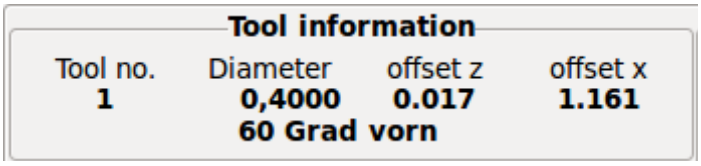
通常、手動で工具を交換する場合は、次のように接続します。

```
net tool-change gmoccapy.toolchange-change <= iocontrol.0.tool-change
net tool-changed gmoccapy.toolchange-changed <= iocontrol.0.tool-changed
net tool-prep-number gmoccapy.toolchange-number <= iocontrol.0.tool-prep-number
net tool-prep-loop iocontrol.0.tool-prepare <= iocontrol.0.tool-prepared
```

note

この接続は postguihal ファイルで行う必要があることに注意してください。

ツールオフセットピンこのピンを使用すると、ツール情報フレームに X および Z のアクティブなツールオフセット値を表示できます。 G43 が送信された後にのみアクティブになることを知っておく必要があります。



- gmoccapy.tooloffset-x
- gmoccapy.tooloffset-z

note

この接続は postguihal ファイルで行う必要があることに注意してください。

Note

tooloffset-x ラインはミルでは必要なく、些細な運動学のミルでは表示されません。

```
net tooloffset-x gmoccapy.tooloffset-x <= motion.tooloffset.x
net tooloffset-z gmoccapy.tooloffset-z <= motion.tooloffset.z
```

gmoccapy はオフセットを更新するために独自に処理し、ツールの変更後に G43 を送信しますが、自動モードではないことに注意してください。

重要

したがって、プログラムを作成すると、ツールを変更するたびに G43 を含める必要があります。

3.11.47 自動工具測定

Gmoccapy は、統合された自動ツール測定を提供します。 この機能を使用するには、いくつかの追加設定を行う必要があります。また、提供されている hal ピンを使用して、独自の ngc 再マップ手順で値を取得することもできます。

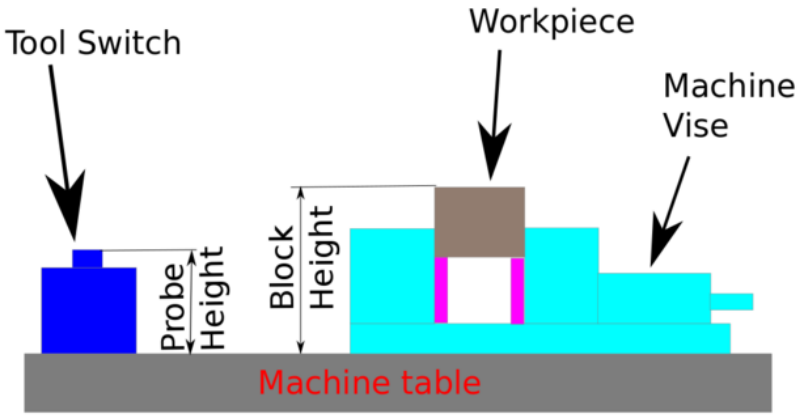
[重要]最初のテストを開始する前に、設定ページでプローブの高さとプローブの速度を入力することを忘れないでください！ 設定ページのツール測定を参照してください

工具測定ビデオをご覧になることもお勧めします。工具測定関連のビデオをご覧ください。

gmoccapy でのツール測定は、他の多くの GUI とは少し異なります。 次の手順に従う必要があります。

- X と Y のワークピースのタッチ
- ツールスイッチが配置されているベースからブロックの上面（チャックなどを含む）までのブロックの高さを測定します。
- ボタンブロックの高さを押して、測定値を入力します
- 自動モードに移動して、プログラムを開始します

ここに小さなスケッチがあります：



最初に与えられた工具交換で工具が測定され、オフセットはブロックの高さに合うように自動的に設定されます。 gmoccapy 方式の利点は、参照ツールが必要ないことです。

Note

プログラムには、最初にツールの変更が含まれている必要があります。 工具は以前に使用したことがある場合でも測定されるため、ブロックの高さが変更されても危険はありません。 あなたのチューブでそれを行う方法を示すいくつかのビデオがあります。

1.1.1.1 工具測定ピン

Gmoccapy は、工具測定用に 5 つのピンを提供しています。 ピンは主に gcode サブルーチンから読み取るために使用されるため、コードはさまざまな値に反応する可能性があります。

- gmoccapy.toolmeasurementHAL_BIT ツール測定を有効にするかどうか

- gmoccapy.blockheightHAL_FLOAT ワークの上面の測定値
- gmoccapy.probeheightHAL_FLOAT プローブスイッチの高さ
- gmoccapy.searchvelHAL_FLOAT ツールプローブスイッチを検索する速度
- gmoccapy.probevelHAL_FLOAT ツールの長さをプローブする速度

3.11.47.1 ツール測定 INI ファイルの変更

INI ファイルを変更して、以下を含めます。

RS274NGC セクション

```
[RS274NGC]
# Enables the reading of INI and HAL values from gcode
FEATURES=12
# is the sub, with is called when a error during tool change happens, not needed on every  -
machine configuration
ON_ABORT_COMMAND=O <on_abort> call
# The remap code
REMAP=M6 modalgroup=6 prolog=change_prolog ngc=change epilog=change_epilog
```

ツールセンサーセクションツールセンサーの位置とプロービング動作の開始位置。すべての値は絶対座標です。ただし、MAXPROBE は、相対動作で指定する必要があります。

```
[TOOLSENSOR]
X = 10
Y = 10
Z = -20
MAXPROBE = -20
```

位置変更セクションこれは意図的に TOOL_CHANGE_POSITION という名前ではありません-canon はその名前を使用し、そうでない場合は干渉します。 工具交換コマンドを出す前に機械を動かす位置。 すべての値は絶対座標です。

```
[CHANGE_POSITION]
X = 10
Y = 10
Z = -2
```

Python セクション Python プラグインは、インタプリタとタスクを提供します。

```
[PYTHON]
# The path to start a search for user modules
PATH_PREPEND = python
```

```
# The start point for all.
TOPLEVEL = python/toplevel.py
```

3.11.47.2 必要なファイル

次のファイルを設定ディレクトリにコピーする必要があります

まず、config フォルダーにディレクトリ python を your_linuxcnc-dev_directory / configs / sim / gmoccapy / python から作成し、toplevel.py を config_dir / python フォルダーにコピーします。 remap.py を config_dir / python フォルダーにコピーします stdglue.py を config_dir / python フォルダーにコピーします。

your_linuxcnc-dev_directory / configs / sim / gmoccapy / macros から、on_abort.ngc を SUBROUTINE_PATH で指定されたディレクトリにコピーします。RS274NGC セクションを参照してください。 your_linuxcnc-ev_directory / configs / sim / gmoccapy / macros から change.ngc をコピーして SUBROUTINE_PATH として指定されたディレクトリは、RS274NGC セクションを参照してください。 エディターで change.ngc を開き、次の行（49 および 50）のコメントを解除します。

```
F #<_hal[gmoccapy.probevel]>
G38.2 Z-4
```

ニーズに合わせてこのファイルを変更することをお勧めします。

3.11.47.3 必要な Hal 接続

次のように、hal ファイルのツールプローブを接続します。

```
net probe motion.probe-input <= <your_input_pin>
```

行は次のようになります。

```
net probe motion.probe-input <= parport.0.pin-15-in
```

postgui.hal ファイルに以下を追加します。

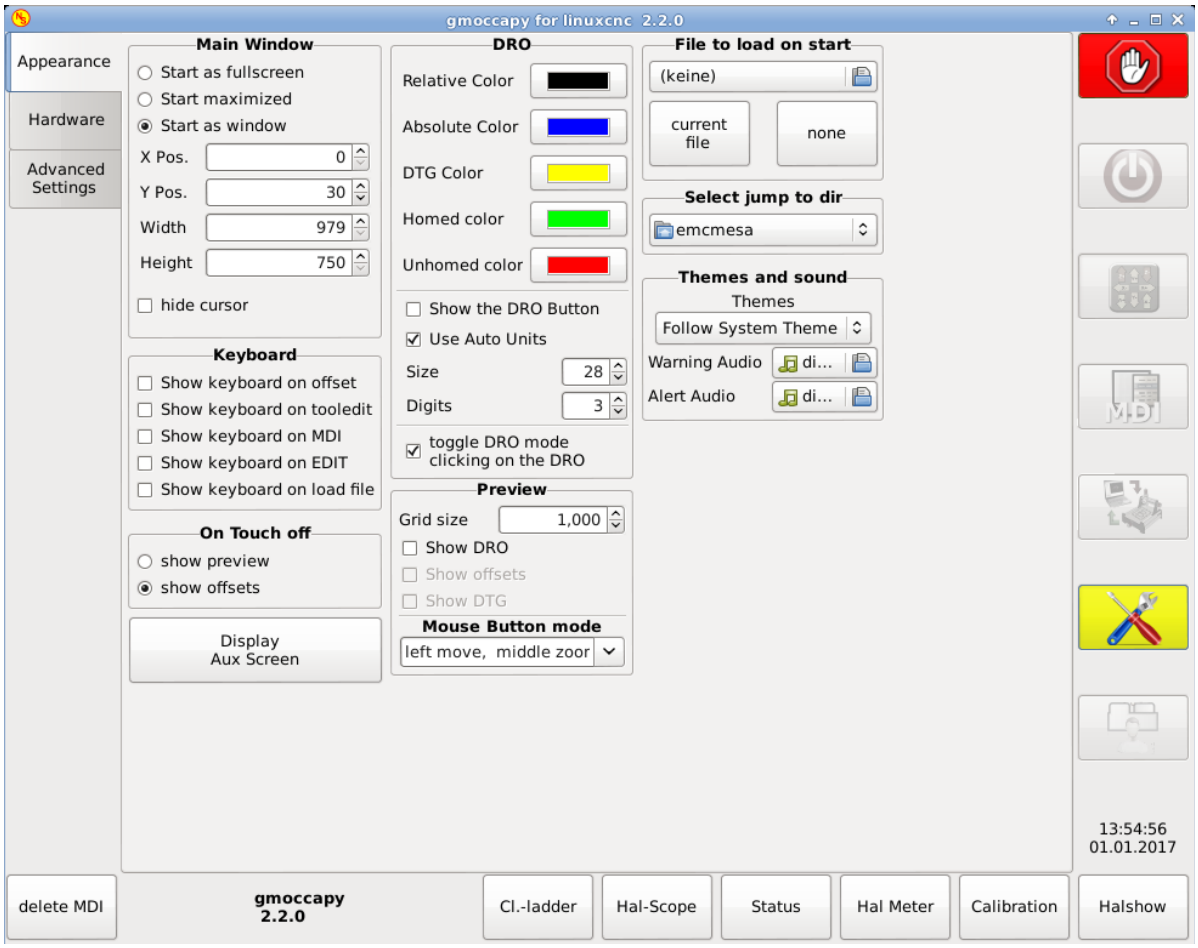
```
# The next lines are only needed if the pins had been connected before
unlinkp iocontrol.0.tool-change
unlinkp iocontrol.0.tool-changed
unlinkp iocontrol.0.tool-prep-number
unlinkp iocontrol.0.tool-prepared
# link to gmoccapy toolchange, so you get the advantage of tool description on change -
dialog
net tool-change gmoccapy.toolchange-change <= iocontrol.0.tool-change
```

```
net tool-changed gmoccapy.toolchange-changed <= iocontrol.0.tool-changed
net tool-prep-number gmoccapy.toolchange-number <= iocontrol.0.tool-prep-number
net tool-prep-loop iocontrol.0.tool-prepare <= iocontrol.0.tool-prepared
```

3.11.48 設定ページ

ページに入るには、クリックしてロック解除コードを入力する必要があります。デフォルトでは witch は 123 です。 この時点で変更する場合は、非表示の設定ファイルを編集する必要があります。詳細については、表示セクションを参照してください。

このページは、次のように表示されます。



このページは、次の 3 つのメインタブに分かれています。

1.1.1.1 外観

このタブには、次のオプションがあります。

MainWindow

ここで、GUI の開始方法を選択できます。 これの主な理由は、ユーザーがコードに触れることなく開始オプションを簡単に設定できるようにしたいという願望でした。

3 つのオプションがあります。

- フルスクリーンで開始
- 最大化を開始
- ウィンドウとして開始

ウィンドウとして開始を選択すると、位置とサイズを設定するためのスピンボックスがアクティブになります。 一度設定すると、GUI はその場所で、選択したサイズで毎回起動します。 それでも、ユーザーはマウスを使用してサイズと位置を変更できますが、それは設定に影響を与えません。

カーソルを非表示にすると、カーソルを非表示にできます。これは、タッチスクリーンを使用する場合に非常に便利です。

キーボード

チェックボックスを使用すると、ユーザーは、オンボードキーボードをすぐに表示するか、MDI モードに入るとき、オフセットページ、ツール編集ウィジェットに入るとき、または編集モードでプログラムを開くかを選択できます。 一番下のボタンリストのキーボードボタンはこの設定の影響を受けないため、ボタンを押すことでキーボードを表示または非表示にできます。 デフォルトの動作は、チェックボックスによって設定されます。

デフォルトは：

> Note
>
> このセクションが重要でない場合は、仮想キーボードをインストールしていません。+オンボードおよびマッチボックスキーボードがサポートされています。

- オフセットでキーボードを表示= True
- tooledit でキーボードを表示= False
- MDI でキーボードを表示= True
- EDIT = True でキーボードを表示する
- ロードファイルにキーボードを表示= False

キーボードのレイアウトが正しくない場合、つまり X をクリックすると Z が表示されますが、ロケール設定に関連して、レイアウトが正しく設定されていません。 オンボードの場合、次の内容の小さなバッチファイルで解決できます。

```
#!/bin/bash
setxkbmap -model pc105 -layout de -variant basic
```

「de」の文字はドイツ語用です。ロケール設定に従って設定する必要があります。 LinuxCNC を起動する前にこのファイルを実行するだけで、ローカルフォルダにスターターを追加することもできます。

```
./config/autostart
```

起動時にレイアウトが自動的に設定されるようにします。

マッチボックスキーボードの場合は、独自のレイアウトを作成する必要があります。ドイツ語のレイアウトの場合は、フォーラムで質問してください。

オンタッチオフ

対応する下部のボタンをクリックしてタッチオフモードに入る場合は、プレビュータブまたはオフセットページタブを表示するオプションを指定します。

- ショープレビュー
- オフセットを表示

ノートブックのタブが表示されているので、どのような場合でも両方のビューを切り替えることができます。

DRO オプション

さまざまな DRO 状態の背景色を選択するオプションがあります。 したがって、1 型 2 色覚（赤/緑の弱さ）に苦しんでいるユーザーは適切な色を選択することができます

デフォルトでは、背景は次のとおりです。

- 相対モード=黒
- 絶対モード=青
- 移動距離=黄色

DRO の前景色は、次の方法で選択できます。

- homed color = green
- unhomed color = red

プレビューで dro を表示

DRO はプレビューウィンドウに表示されます+

オフセットを表示+オフセットはプレビューウィンドウに表示されます+

DTG を表示

移動距離はプレビューウィンドウに表示されます+

Note

DRO をクリックすると、DRO モード（絶対、相対、移動距離）を変更できます。 * DRO の左側の文字をクリックすると、ポップアップウィンドウで軸の値を設定でき、タッチオフボタンボタンに目を通す必要がないため、値の設定が簡単になります。 数字（DRO の右側）をクリックすると、上記のように DRO モードが切り替わります。 *

サイズ

DRO フォントのサイズを設定できます。デフォルトは 28 です。より大きな画面を使用する場合は、サイズを 56 まで増やすことができます。4 軸を使用する場合、DRO フォントのサイズは値の 3/4 になります。 、スペース上の理由から。 +

数字

DRO の桁数を 1 から 5 まで設定します。

> **Note**
>
> インペリアルは、そのメトリックより 1 桁多く表示されます。 したがって、インペリアルマシン単位を使用していて、桁の値を 1 に設定すると、メートル法では桁がまったく取得されません。

DRO モードを切り替えます

アクティブでない場合、DRO をマウスでクリックしても何も実行されません。

デフォルトでは、このチェックボックスはアクティブになっているため、DRO をクリックするたびに、DRO の読み取り値が実際の値から DTG を基準にした値（移動距離）に切り替わります。

それでも、軸の文字をクリックすると、ポップアップダイアログが開き、軸の値を設定します+

プレビュー

グリッドサイズプレビューウィンドウのグリッドサイズを設定します。 残念ながら、マシンの単位がメートル法であっても、サイズはインチで設定する必要があります。 将来のリリースで修正されることを望んでいます。

【注意】グリッドは透視図では表示されません。

DRO を表示

プレビューウィンドウにも DRO が表示され、フルサイズのプレビューで自動的に表示されます

Show DTG は、Show DRO がアクティブであり、フルサイズのプレビューではない場合にのみ、プレビューに DTG（終点までの直接距離）も表示します。

[オフセットを表示]は、プレビューウィンドウにオフセットを表示します。

[注]このオプションのみをオンにし、他のオプションをオフのままにすると、オフセットページがフルサイズでプレビューされます。

マウスボタンモードこのコンボボックスでは、マウスのボタンの動作を選択して、プレビュー内で回転、移動、またはズームできます。

- 左回転、中央移動、右ズーム
- 左ズーム、中央移動、右回転

- 左移動、中央回転、右ズーム
- 左ズーム、中央回転、右移動
- 左移動、中央ズーム、右回転
- 左回転、中央ズーム、右移動

デフォルトは左移動、中央ズーム、右回転です。

マウスホイールは、すべてのモードでプレビューをズームします。

> **Tip**
>
> プレビューで要素を選択すると、選択した要素が回転の中心点と見なされ、自動モードでは対応するコード行が強調表示されます。

起動時にロードするファイル

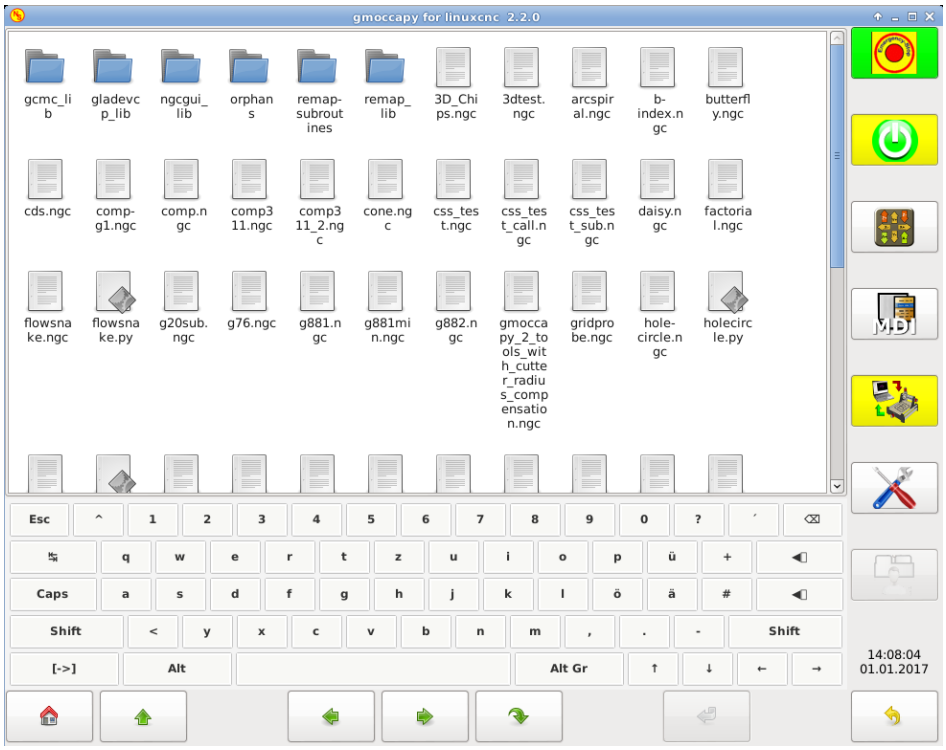
起動時にロードするファイルを選択します。 他の GUI では、ユーザーが INI ファイルの編集を余儀なくされたため、これを変更するのは非常に面倒でした。

起動時にロードするファイルを選択します。 ファイルがロードされている場合は、現在のボタンを押すことで設定でき、起動時にプログラムがロードされないようにすることができます。[なし]ボタンを押すだけです。

ファイル選択画面では、INI ファイルで設定したフィルターが使用されます。フィルターが指定されていない場合は、ngc ファイルのみが表示されます。 [DISPLAY] PROGRAM_PREFIX の INI 設定に従ってパスが設定されます

dir にジャンプ

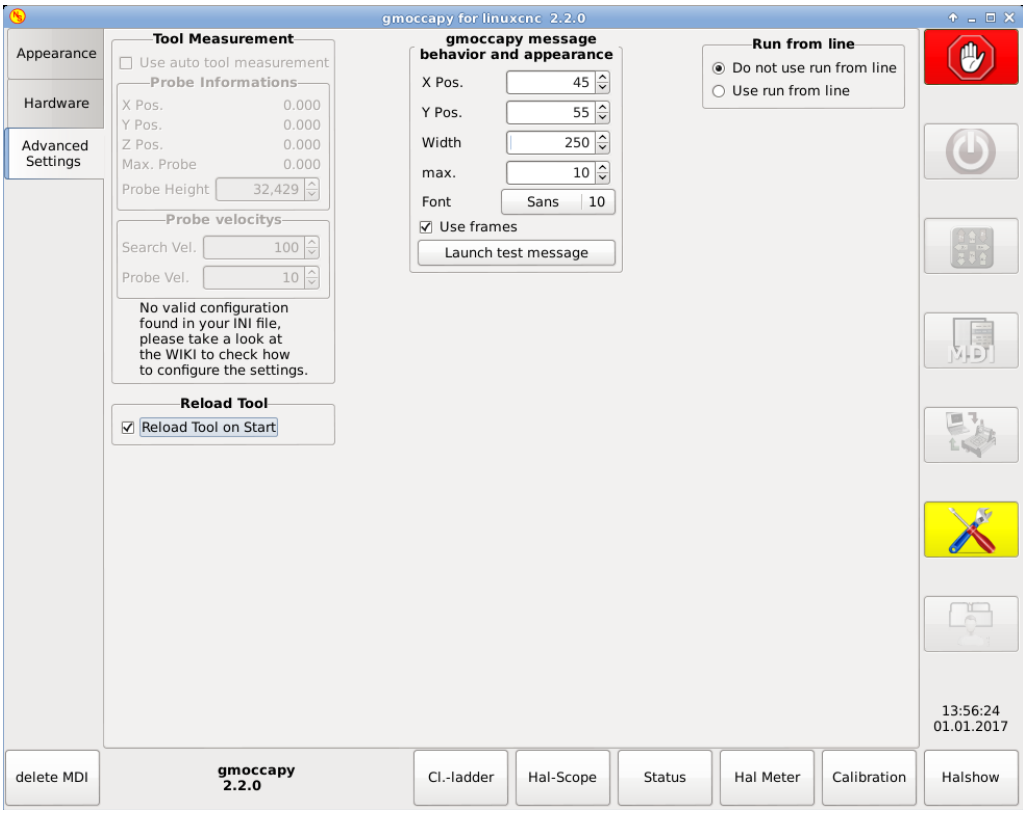
ここで、ファイル選択ダイアログの対応するボタンを押した場合にジャンプするディレクトリを設定できます。



テーマとサウンド

　これにより、ユーザーは、適用するデスクトップテーマと、再生するエラーおよびメッセージの音を選択できます。 デフォルトでは、「FollowSystemTheme」が設定されています。

3.11.48.1 ハードウェア



ハードウェア MPG スケール

MPG ホイールを接続するさまざまなハルピンについて、適用する個々のスケールを選択できます。 これの主な理由は、hal 接続を介してこれを解決するための私自身のテストであり、非常に複雑な hal ファイルになりました。 100 ipr の MPG ホイールを使用していて、最大 vel を 14000 から 2000 mm / min に減速したいとしていると想像してください。これには、12000 インパルスが必要で、ホイールが 120 回転します。 または、500 ipr の MPG ホイールを使用していて、スピンドルオーバーライドウィッチの制限を 50〜120%に設定したい他のユーザーは、70 インパルス以内で最小から最大に移動します。つまり、1/4 回転すらしません。

デフォルトでは、すべてのスケールは計算を使用して設定されます。

```
(MAX - MIN)/100
```

キーボードショートカット

キーボードボタンを使用してマシンをジョギングしたいユーザーもいれば、これを決して許可しないユーザーもいます。 したがって、誰もがそれらを使用するかどうかを選択できます。 キーボードジョギングはオペレーターと機械に重大なリスクをもたらすため、使用することはお勧めしません。

デフォルトでは、キーボードショートカットは使用されません。

旋盤を使用する場合は、ショートカットが異なりますのでご注意ください。 旋盤のセクションを参照してください

- 左矢印または NumPad_Left = X マイナス
- 右矢印または NumPad_Right = X プラス
- 上矢印または NumPad_Up = Y plus
- 下矢印または NumPad_Down = Y マイナス
- PageUp または NumPad_Page_Up = Z plus
- PageDown または NumPad_Page_Down = Z マイナス
- F1 = Estop（キーボードショートカットが無効になっている場合でも機能します）
- F2 =マシンオン
- F3 =手動モード
- F5 = MDI モード
- ESC =中止

AUTO モードでは、次のキーショートカットを使用できます* R または r =プログラムの実行* P または p =プログラムの一時停止* S または s =プログラムの再開* Control および R または r は、ロードされたファイルをリロードします

メッセージ処理用の追加のキーがあります。メッセージの動作と外観を参照してください。

- WINDOWS =最後のメッセージを削除する
- <STRG> <SPACE> =すべてのメッセージを削除する

オプションのロックを解除する

設定ページのロックを解除するには、次の 3 つのオプションがあります。

- ロック解除コードを使用します（ユーザーは入るためにコードを与える必要があります）
- ロック解除コードを使用しないでください（セキュリティチェックはありません）
- hal ピンを使用してロックを解除します（設定のロックを解除するには、ハードウェアピンをハイにする必要があります。ハードウェアロック解除ピンを参照してください）

デフォルトはロック解除コードを使用することです（デフォルト= 123）

スピンドル

開始 RPM は、スピンドルが開始され、S 値が設定されていない場合に使用される rpm を設定します。

> Note
>
> この値は、INI の[DISPLAY] DEFAULT_SPINDLE_SPEED の設定に従って事前設定されます。 設定ページで設定を変更した場合、その値はその時点からデフォルトになり、INI ファイルは変更されません。

MIN および MAX 設定を使用して、メイン画面の INFO フレームに表示されるスピンドルバーの制限を設定します。 間違った値を与えてもエラーはありません。 最大 2000 を与え、スピンドルが 4000 rpm を作る場合、2000rpm より高い速度ではバーレベルだけが間違っています。

```
default values are
MIN = 0
MAX = 6000
```

タートルジョグ

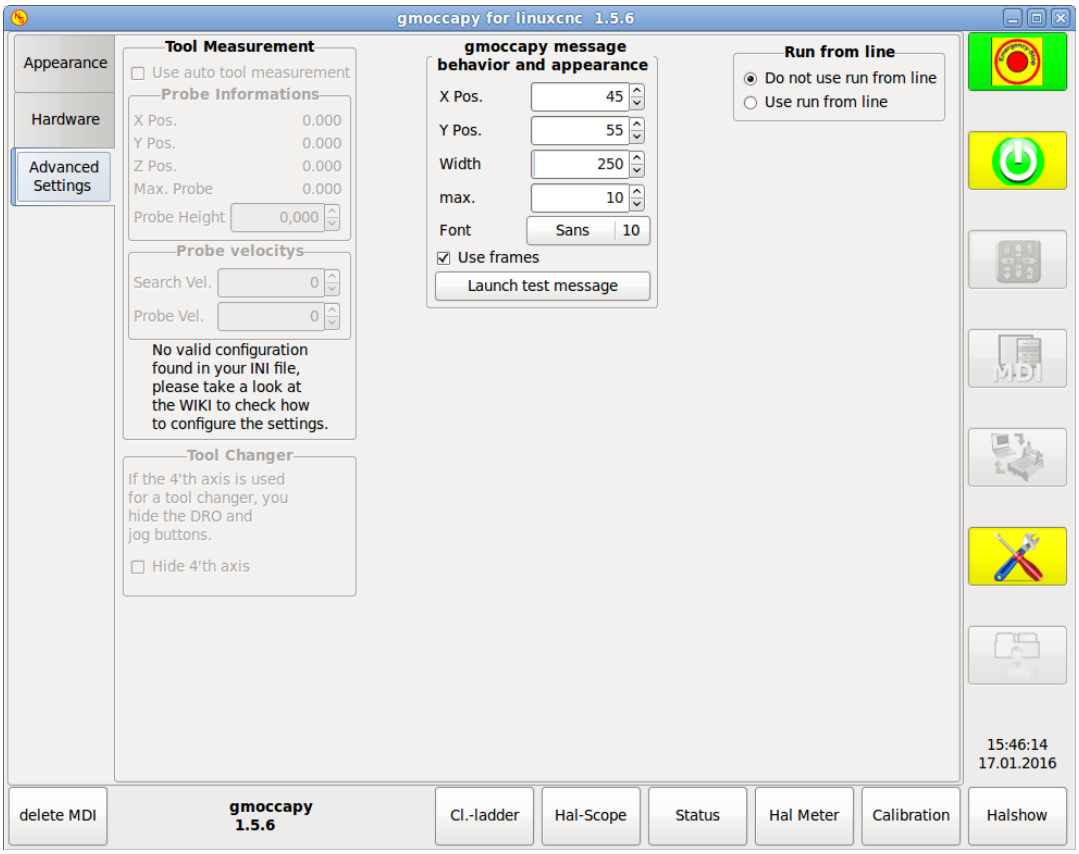
この設定は、ジョグ速度に影響を与えます。

- タートルジョグボタンを非表示にすると、ジョグ速度スライダーの右側のボタンが非表示になります。このボタンを非表示にすると、ウサギのアイコンが表示されるように注意してください。そうしないと、を使用して計算されたタートルジョグ速度よりも速くジョギングできなくなります。 タートルジョギングファクター。
- タートルジョグ係数は、タートルジョグモードに適用するスケールを設定します。 係数を 20 に設定すると、タートルモード（ボタンを押してタートルを表示）の場合、タートルの最大ジョグ速度はマシンの最大速度の 1/20 になります。

Note

このボタンは、タートルジョグハルピンを使用してアクティブにできます。

3.11.48.2 高度な設定



Note

この部分が敏感でない場合、ツール測定を使用するための有効な INI ファイル構成がありません。

自動工具測定を確認してください

- 自動工具測定を使用する：チェックすると、各工具交換後に工具測定が実行され、結果が工具テーブルに保存され、変更後に G43 が実行されます。

プローブ情報次の情報は INI ファイルから取得されたものであり、絶対座標で指定する必要があります

- XPos。 =ツールスイッチの X 位置
- Y 位置 =ツールスイッチの Y 位置
- Z 位置 =ツールスイッチの Z 位置、この座標にすばやく移動します
- 最大 プローブ=は連絡先を検索する距離です。連絡先が指定されていない場合、エラーが発生します。 距離は、Z 位置から移動を開始する相対座標で指定する必要があるため、下に移動するには負の値を指定する必要があります。

- プローブの高さ=はプローブスイッチの高さであり、測定できます。 プローブスイッチが配置されているベースをタッチオフして、ゼロに設定するだけです。 次に、ツールを変更し、tool_offset_z 値を監視します。これは、ここに入力する必要のある最高値です。

プローブ速度

- Vel を検索します。 =ツールスイッチを検索する速度。接触後、ツールは再び上昇し、プローブ vel を使用して再びプローブに向かって移動するため、より良い結果が得られます。
- プローブベル。 =スイッチへの 2 回目の移動の速度です。より良いタッチ結果を得るには、速度を遅くする必要があります（sim モードでは、これは macros / change.ngc でコメント化されています。そうでない場合、ユーザーはプローブボタンを 2 回クリックする必要があります。 ）

オンにすると、スピンドルのツールが設定ファイルの変更ごとに保存され、起動時に最後にマウントされたツールをリロードできるようになります。 ツールは、MDI コマンドの実行が許可されていないため、すべての軸が原点復帰した後にロードされます。 NO_FORCE_HOMING を使用する場合、必要な all_homed_signal が発行されないため、この機能を使用することはできません。

メッセージの動作と外観

これにより、メッセージまたはエラーテキストを表示する小さなポップアップウィンドウが表示されます。動作は、1 つの軸が使用するものと非常に似ています。 特定のメッセージを削除するには、閉じるボタンをクリックします。最後のメッセージを削除する場合は、キーボードの WINDOWS キーを押すか、<STRG> <SPACE>が付いているメッセージをすべて削除します。

あなたはいくつかのオプションを設定することができます：

- X Pos =画面の左上隅からピクセル単位でカウントされた X 単位のメッセージの左上隅の位置。
- Y Pos =画面の左上隅からピクセル単位でカウントされた Y 単位のメッセージの左上隅の位置。
- 幅=メッセージボックスの幅
- max = 1 で表示する最大メッセージ。これを 10 に設定すると、11 番目のメッセージで最初のメッセージが削除されるため、最後の 10 個のメッセージのみが表示されます。
- フォント=メッセージの表示に使用するフォントとサイズ
- フレームを使用=チェックボックスをオンにすると、各メッセージがフレームに表示されるため、メッセージを区別するのがはるかに簡単になります。 ただし、もう少しスペースが必要になります。
- ボタン起動テストメッセージは、想定どおりに機能し、メッセージが表示されるため、エラーを生成することなく設定の変更を確認できます。

行から実行オプション行からの実行を許可または禁止できます。 これにより、対応するボタンが非表示（グレー表示）に設定されるため、ユーザーはこのオプションを使用できなくなります。 デフォルトは、回線からの実行を無効にします。

警告

LinuxCNC は開始行の前のコード内の前の行を処理しないため、run fromline を使用することはお勧めしません。 したがって、エラーまたはクラッシュが発生する可能性が非常に高くなります。

3.11.49 旋盤固有のセクション

INI ファイルで LATHE = 1 が指定されている場合、GUI はその外観を旋盤の特別なニーズに変更します。 主に Y 軸が非表示になり、ジョグボタンの配置が異なります。

図 4-48

図 4-16

ご覧のとおり、R DRO の背景は黒で、DDRO は灰色です。 これは、アクティブな G コード G7 または G8 に応じて変わります。 アクティブモードは黒い背景で表示されます。これは、表示されている画像では G8 がアクティブであることを意味します。

標準画面との次の違いは、ジョグボタンの位置です。 X と Z の場所が変わり、Y はなくなりました。 X +ボタンと X-ボタンは、通常の旋盤または戻る工具旋盤に応じて場所が変わることに注意してください。

また、キーボードの動作も変更されます。

通常の旋盤：

- 左矢印または NumPad_Left = Z マイナス
- 右矢印または NumPad_Right = Z プラス
- 上矢印または NumPad_Up = X マイナス
- 下矢印または NumPad_Down = X plus

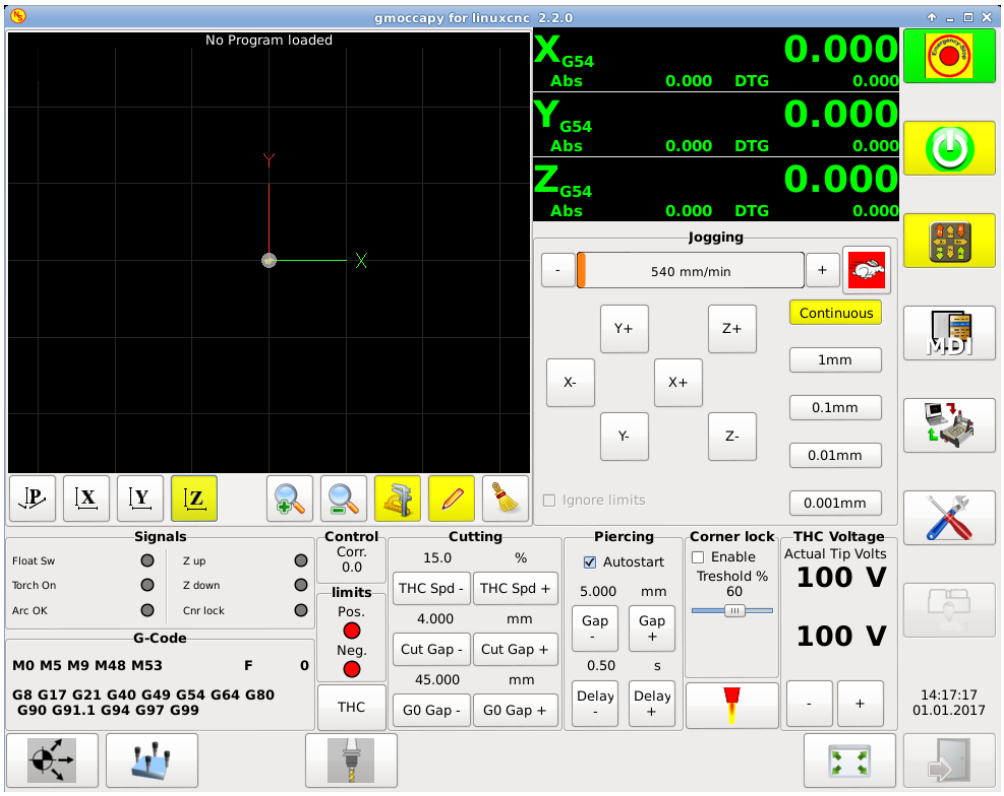
バックツール旋盤：

LinuxCNC V2.8.1-84-g81a16a1fd, 2021-03-29
左矢印または NumPad_Left = Z マイナス

- 右矢印または NumPad_Right = Z プラス
- 上矢印または NumPad_Up = X plus
- 下矢印または NumPad_Down = X マイナス

工具情報フレームには Z オフセットだけでなく、X オフセットも表示され、工具テーブルにはすべての旋盤関連情報が表示されます。

3.11.50 プラズマ特異的セクション



マリウスによって維持されている、実際に成長している非常に優れた WIKI があります。Plasmawiki ページを参照してください。

3.11.51 YouTube のビデオ

これは、gmoccapy の動作を示すビデオです。残念ながら、ビデオには最新バージョンの gmoccapy が表示されていませんが、使用方法は今後あまり変更されません。 私はできるだけ早くビデオを実現しようとします。

1.1.1.1 基本的な使用法

https://www.youtube.com/watch?v=O5B-s3uiI6g

3.11.51.1 シミュレートされたジョグホイール

http://youtu.be/ag34SGxt97o

3.11.51.2 設定ページ

https://www.youtube.com/watch?v=AuwhSHRJoiI

3.11.51.3 シミュレートされたハードウェアボタン

Deutsch = http://www.youtube.com/watch?v=DTqhY-MfzDE

English = http://www.youtube.com/watch?v=ItVWJBK9WFA

3.11.51.4 ユーザータブ

http://www.youtube.com/watch?v=rG1zmeqXyZI

3.11.51.5 工具測定ビデオ

Auto Tool Measurement Simulation = http://youtu.be/rrkMw6rUFdk

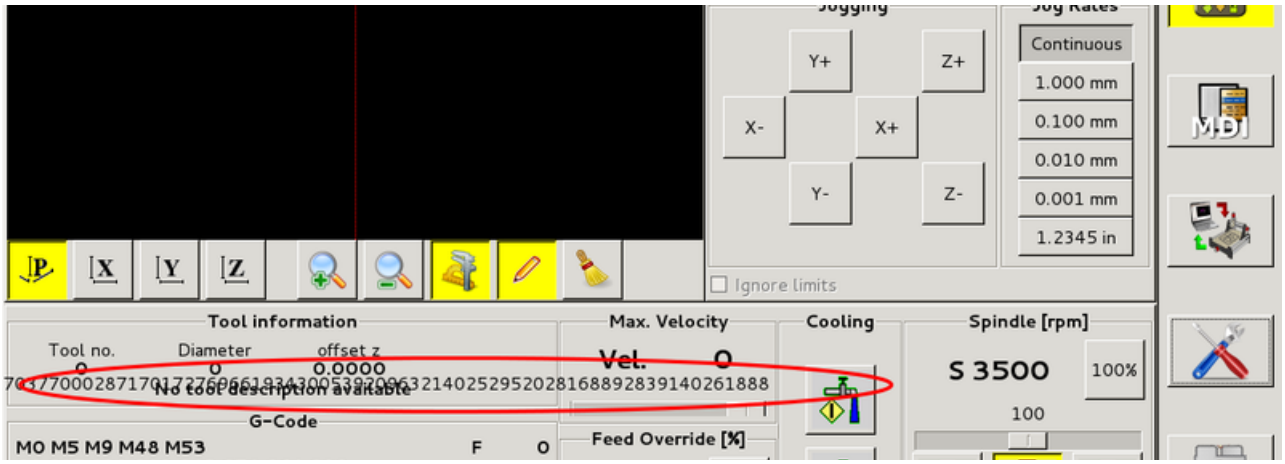
Auto Tool Measurement Screen = http://youtu.be/Z2ULDj9dzvk

Auto Tool Measurement Machine = http://youtu.be/1arucCaDdX4

3.11.52 既知の問題

1.1.1.1 情報エリアの奇妙な数字

gmoccapy の情報領域に次のような奇妙な数字が表示された場合：



古いバージョンの StepConfWizard を使用して構成ファイルを作成しました。 MAX_LINEAR_VELOCITY = xxx という名前の[TRAJ]の下の INI ファイルに間違ったエントリを作成しました。 そのエントリを MAX_VELOCITY = xxx に変更します

3.11.52.1 マクロを終わらせない

次のように、移動せずにマクロを使用する場合：

```
o<zeroxy> sub
G92.1
G92.2
G40
G10 L20 P0 X0 Y0
o<zeroxy> endsub
m2
```

インタプリタはその状態を IDLE に変更する必要があるため、gmoccapy はマクロの終わりを認識しませんが、マクロはインタプリタを新しい状態に設定することさえしません。 これを回避するには、G4P0.1 ラインを追加して必要な信号を取得します。 正しいマクロは次のとおりです。

```
o<zeroxy> sub
G92.1
G92.2
G40
G10 L20 P0 X0 Y0
G4 P0.1
o<zeroxy> endsub
m2
```

3.12 NGCGUI

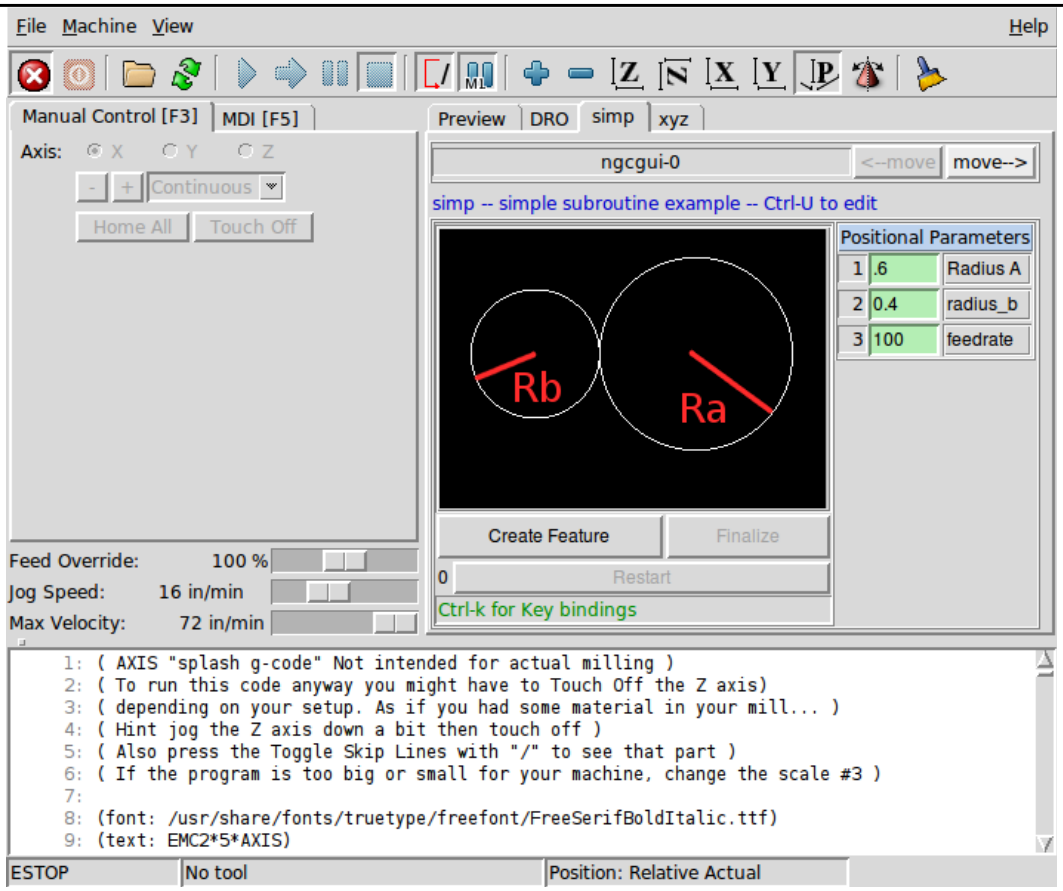


図 4-17

3.12.1 概要

- NGCGUI は、サブルーチンを操作するための Tcl アプリケーションです。 これにより、LinuxCNC との会話型インターフェイスを使用できます。 サブルーチンを実行するために必要な順序でサブルーチンを編成し、サブルーチンを 1 つのファイルに連結して、完全なパートプログラムにすることができます。
- NGCGUI は、スタンドアロンアプリケーションとして実行することも、軸 GUI の複数のタブページに埋め込むこともできます。
- PYNGCGUI は、ngcgui の代替の Python 実装です。
- PYNGCGUI は、スタンドアロンアプリケーションとして実行することも、gladevcp アプリケーション axis、touchy、gscreen、および gmoccapy の埋め込みをサポートする任意の GUI にタブページ（独自の複数のサブルーチンタブのセットを含む）として埋め込むこともできます。

NGCGUI または PYNGCGUI の使用：

- INI ファイルで指定されたサブルーチンごとにタブページが用意されています
- カスタムタブを使用して、新しいサブルーチンタブページをその場で追加できます
- 各サブルーチンタブページには、すべてのサブルーチンパラメータの入力ボックスがあります

- 入力ボックスには、サブルーチンファイル内の特別なコメントで識別されるデフォルト値とラベルを付けることができます。
- サブルーチン呼び出しを連結して、複数ステップのプログラムを形成できます。
- ngcgui 規則に準拠する任意の単一ファイル G コードサブルーチンを使用できます
- 変数のタグ付けに関する ngcgui 規則に準拠する任意の gcmc（G コード-メタコンパイラ）プログラムを使用できます。（gcmc 実行可能ファイルは個別にインストールする必要があります。http：//www.vagrearg.org/content/gcmc を参照してください）

Note

NGCGUI と PYNGCGUI は同じ関数を実装し、両方ともいくつかの ngcgui 固有の規則に準拠する.ngc ファイルと.gcmc ファイルを処理します。 このドキュメントでは、NGCGUI という用語は一般的にいずれかのアプリケーションを指します。

3.12.2 デモンストレーション構成

LinuxCNC 構成ピッカーによって提供されるサンプル構成の sim ディレクトリーには、いくつかのデモンストレーション構成があります。 構成ピッカーは、システムのメインメニューにあります。

CNC > LinuxCNC

axis、touchy、gscreen、および gmoccapy の例が含まれています。 これらの例は、3 軸（XYZ）デカルト構成（ミルなど）と旋盤（XZ）セットアップの両方を示しています。 いくつかの例は、タッチスクリーンシステムでのポップアップキーボードの使用を示し、他の例は、gcmc（G コードメタコンパイラ）アプリケーション用に作成されたファイルの使用を示しています。 微妙な例は、gladevcp バックプロットビューア（gremlin_view）の組み込みも示しています。

最も単純なアプリケーションは次のとおりです。

Sample Configurations/sim/axis/ngcgui /ngcgui_simple

gcmc の互換性を示す包括的な例は次のとおりです。

Sample Configurations/sim/axis/ngcgui/ngcgui_gcmc

gladevcp アプリとして埋め込まれ、gcmc を使用する包括的な例は次のとおりです。

Sample Configurations/sim/gscreen/ngcgui/pyngcgui_gcmc

sim 構成の例では、G コードサブルーチン（.ngc）ファイルと G コードエタコンパイラ（.gcmc）ファイルの例を提供するライブラリファイルを使用しています。

- nc_files / nccgui_lib

–arc1.ngc-カッター半径補正を使用した基本アーク

– arc2.ngc-中心、オフセット、幅、角度で指定されたアーク（arc1 を呼び出します）

–backlash.ngc-ダイヤルゲージで軸のバックラッシュを測定するルーチン

–db25.ngc-DB25 プラグカットアウトを作成します

– gosper.ngc-再帰デモ（flowsnake）

–helix.ngc-らせんまたは D 穴の切断

–helix_rtheta.ngc-半径と角度で配置されたらせんまたは D 穴

–hole_circle.ngc-円上の等間隔の穴

–ihex.ngc-内部六角形

–iquad.ngc-内部四辺形

–ohex.ngc-六角形の外側

–oquad.ngc-四辺形の外側

– qpex_mm.ngc-qpockets のデモ（mm ベース）

– qpex.ngc-qpockets のデモ（インチベース）

–qpocket.ngc-四辺形ポケット

–rectangle_probe.ngc-長方形の領域をプローブします

–simp.ngc-2 つの円を作成する簡単なサブルーチンの例

–slot.ngc-2 つのエンドポイントを接続するためのスロット

–xyz.ngc-ボックス形状に制約されたマシンエクササイザー

- nc_files / nccgui_lib / lathe

–g76base.ngc-g76 スレッディングの GUI

– g76diam.ngc-大径、小径で指定されたねじ切り

–id.ngc-内径を穴あけします

–od.ngc-外径を回転させます

–taper-od.ngc-外径のテーパーを回します

- nc_files / gcmc_lib

–drill.gcmc-長方形パターンで穴を開ける

–square.gcmc-gcmc ファイルの変数タグの簡単なデモ

–star.gcmc-関数と配列を示す gcmc デモ

デモンストレーションを試すには、sim 構成を選択し、linuxCNC プログラムを起動します。

軸 GUI を使用している場合は、E-Stop、Machine Power、HomeAll の順に押します。 ngcgui タブを選択し、空の空白に適切な値を入力して、[フィーチャの作成]、[ファイナライズ]の順に押します。 最後に[実行]ボタンを押して、実行を確認します。 さまざまなタブページから複数の機能や機能を作成してみてください。

1 つのファイルに連結された複数のサブルーチンを作成するには、各タブに移動して空白を埋め、[フィーチャの作成]を押してから、矢印キーを使用して、それらを整理するために必要なタブを移動します。 次に、[ファイナライズ]を押し、プロンプトに応答して作成します

他の GUI も同様の機能を備えていますが、ボタンと名前が異なる場合があります。

Note

デモンストレーション構成は、提供されている例のほんの一部のタブページを作成します。 カスタムタブのある GUI は、ライブラリのサンプルサブルーチン、または linuxCNC サブルーチンパスにある場合はユーザーファイルを開くことができます。

特別なキーバインディングを表示するには、ngcgui タブページ内をクリックしてフォーカスを取得し、Control-k を押します。

デモンストレーションサブルーチンは、ディストリビューションに含まれているシミュレートされたマシン構成で実行する必要があります。 ユーザーは、実際のマシンで実行する前に、プログラムの動作と目的を常に理解する必要があります。

3.12.3 ライブラリの場所

deb パッケージからインストールされた linuxCNC インストールでは、ngcgui のシミュレーション構成は、次の目的でユーザーが使用できない LinuxCNC ライブラリへのシンボリックリンクを使用します。

- nc_files / nccgui_libngcgui 互換のサブファイル
- nc_files / ngcgui_lib / lathengcgui 互換の旋盤サブファイル
- nc_files / gcmc_libngcgui-gcmc 互換プログラム
- nc_files / nccgui_lib / utilitysubs ヘルパーサブルーチン
- nc_files / nccgui_lib / mfiles ユーザー M ファイル

これらのライブラリは、linuxCNC（および ngcgui）で使用される検索パスを指定する ini ファイルアイテムによって配置されます。

```
[RS274NGC]
SUBROUTINE_PATH = ../../nc_files/ngcgui_lib:../../nc_files/gcmc_lib:../../nc_files/  -
ngcgui_lib/utilitysubs
```

```
USER_M_PATH = ../../nc_files/ngcgui_lib/mfiles
```

Note

これらは、検索パッチで使用されるディレクトリを指定する長い行です（複数行に続くことはありません）。 ディレクトリ名はコロン（:)で区切られます。 ディレクトリ名の間にスペースを入れないでください。

ユーザーは、独自のサブルーチンと M ファイル用に新しいディレクトリを作成し、それらを検索パスに追加できます。 たとえば、ユーザーは次のコマンドを使用してターミナルからディレクトリを作成できます。

```
mkdir /home/myusername/mysubs
mkdir /home/myusername/mymfiles
```

次に、システム提供のファイルを作成するか、これらのユーザー書き込み可能なディレクトリにコピーします。 たとえば、ユーザーが gcguicompatible を作成する場合があります

名前の付いたサブファイル：

```
/home/myusername/mysubs/example.ngc
```

新しいディレクトリのファイルを使用するには、ini ファイルを編集して、新しいサブファイルを含め、検索パスを拡張する必要があります。 この例の場合：

```
[RS274NGC]
...
SUBROUTINE_PATH =
/home/myusername/mysubs:../../nc_files/ngcgui_lib:../../nc_files/gcmc_lib  -
:../../nc_files/ngcgui_lib/utilitysubs
USER_M_PATH = /home/myusername/mymfiles:../../nc_files/ngcgui_lib/mfiles
[DISPLAY]
...
NGCGUI_SUBFILE = example.ngc
...
```

LinuxCNC（および ngcgui）は、検索パスでディレクトリを検索するときに最初に見つかったファイルを使用します。 この動作により、パス検索の前半で見つかったディレクトリに同じ名前のサブファイルを配置することで、ngcgui_lib サブファイルを置き換えることができます。 詳細については、インテグレータマニュアルの INI の章を参照してください。

3.12.4 スタンドアロンの使用法

1.1.1.1 スタンドアロン NGCGUI

使用方法については、ターミナルに入力してください。

```
ngcgui --help
Usage:
ngcgui --help | -?
ngcgui [Options] -D nc_files_directory_name
ngcgui [Options] -i LinuxCNC_inifile_name
ngcgui [Options]
Options:
[-S subroutine_file]
[-p preamble_file]
[-P postamble_file]
[-o output_file]
[-a autosend_file] (autosend to axis default:auto.ngc)
[--noauto] (no autosend to axis)
[-N | --nom2] (no m2 terminator (use %))
[--font [big|small|fontspec]] (default: "Helvetica -10 normal")
[--horiz|--vert] (default: --horiz)
[--cwidth comment_width] (width of comment field)
[--vwidth varname_width] (width of varname field)
[--quiet] (fewer comments in outfile)
[--noiframe] (default: frame displays image)
```

Note

スタンドアロンアプリケーションとして、ngcgui は複数回呼び出すことができる単一のサブルーチンファイルを処理します。 複数のスタンドアロン ngcgui アプリケーションを個別に起動できます。

3.12.4.1 スタンドアロン PYNGCGUI

使用方法については、ターミナルに入力してください。

```
pyngcgui --help
Usage:
pyngcgui [Options] [sub_filename]
Options requiring values:
[-d | --demo] [0|1|2] (0: DEMO standalone toplevel)
(1: DEMO embed new notebook)
(2: DEMO embed within existing notebook)
[-S | --subfile sub_filename]
[-p | --preamble preamble_filename]
[-P | --postamble postamble_filename]
[-i | --ini inifile_name]
```

```
[-a | --autofile auto_filename]
[-t | --test testno]
[-K | --keyboardfile glade_file] (use custom popupkeyboard glade file)
Solo Options:
[-v | --verbose]
[-D | --debug]
[-N | --nom2] (no m2 terminator (use %))
[-n | --noauto] (save but do not automatically send result)
[-k | --keyboard] (use default popupkeybaord)
[-s | --sendtoaxis] (send generated ngc file to axis gui)
Notes:
A set of files is comprised of a preamble, subfile, postamble.
The preamble and postamble are optional.
One set of files can be specified from cmdline.
Multiple sets of files can be specified from an inifile.
If --ini is NOT specified:
search for a running linuxCNC and use its inifile
```

note

スタンドアロンアプリケーションとして、pyngcgui は ini ファイル（または実行中の linuxCNC アプリケーション）を読み取って、複数のサブファイルのタブページを作成できます。

3.12.5 NGCGUI の埋め込み

1.1.1.1 Axis への NGCGUI の埋め込み

[DISPLAY]セクションには次の INI ファイル項目があります。（必要な追加項目については、以下の追加セクションを参照してください）

- TKPKG = Ngcgui1.0-NGCGUI パッケージ
- TKPKG = Ngcguittt1.0-彫刻用のテキストを生成するための TrueType トレーサーパッケージ（オプション、TKPKG = Ngcgui に従う必要があります）。
- TTT = truetype-tracer-truetype トレーサープログラムの名前（ユーザー PATH に含まれている必要があります）
- TTT_PREAMBLE = in_std.ngc-オプションで、ttt で作成されたサブファイルに使用されるプリアンブルのファイル名を指定します。（代替：mm_std.ngc）

Note

オプションの TrueType トレーサー項目は、アプリケーション truetype-tracer を使用する ngcgui 互換のタブページを指定するために使用されます。 truetype-tracer アプリケーションは、個別にインストールし、ユーザー PATH に配置する必要があります。

3.12.5.1 PYNGCGUI を GUI の gladevcp タブページとして埋め込む

次の INI ファイルアイテムは、[DISPLAY]セクションにあり、axis、gscreen、または touchygui で使用できます。（必要な追加項目については、以下の追加セクションを参照してください）

EMBED_アイテム

```
EMBED_TAB_NAME = Pyngcgui - name to appear on embedded tab
EMBED_TAB_COMMAND = gladevcp -x {XID} pyngcgui_axis.ui - invokes gladevcp
EMBED_TAB_LOCATION = name_of_location - where the embeded page is located
```

Note

EMBED_TAB_LOCATION 指定子は、軸 gui には使用されません。 pyngcgui は軸に埋め込むことができますが、ngcgui を使用すると（TKPKG = Ngcgui 1.0 を使用）統合がより完全になります。 他の GUI に EMBED_TAB_LOCATION を指定するには、INI 構成の章の DISPLAY セクションを参照してください。

Note

TrueType Tracer gui フロントエンドは、現在 gladevcp アプリケーションでは使用できません。

3.12.5.2 ngcgui または pyngcgui に必要な追加の INI ファイルアイテム

次の INI ファイル項目は、ngcgui または pyngcgui のいずれかを埋め込むすべての GUI の[DISPLAY]セクションにあります。

- NGCGUI_FONT = Helvetica -12 normal-フォント名、サイズ、normal | bold を指定します
- NGCGUI_PREAMBLE = in_std.ngc-サブルーチンの前に追加されるプリアンブルファイル。 複数の一般的なサブルーチン呼び出しを連結する場合、このプリアンブルは 1 回だけ追加されます。 mm ベースのマシンの場合は、mm_std.ngc を使用します
- NGCGUI_SUBFILE = filename1.ngc-filename1 サブルーチンからタブを作成します
- NGCGUI_SUBFILE = filename2.ngc-filename2 サブルーチンからタブを作成します
- 。。。 NS
- NGCGUI_SUBFILE = ""-検索パス内の任意のサブルーチンを開くことができるカスタムタブを作成します
- NGCGUI_OPTIONS = opt1opt2。。。 -NGCGUI オプション

- nonew-新しいカスタムタブの作成を禁止する
- noremove-タブページの削除を禁止します
- noauto-自動送信なし（makeFile を使用してから、保存または手動で送信）
- noiframe-内部画像なし、別のトップレベルウィジェットに画像を表示
- nom2-m2 で終了せず、%ターミネータを使用します。 このオプションは、m2 終端のすべての副作用を排除します

- GCMC_INCLUDE_PATH = dirname1：dirname2-gcmc インクルードファイルのディレクトリを検索します

これは、NGCGUI を Axis に埋め込む例です。 サブルーチンは、[RS274NGC] SUBROUTINE_によって指定されたディレクトリにある必要があります。一部のサンプルサブルーチンは他のサブルーチンを使用するため、SUBROUTINE_PATH ディレクトリに依存関係がある場合はそれを確認してください。 一部のサブルーチンは、[RS274NGC] USER_M_PATH で指定されたディレクトリにある必要があるカスタム Mfile を使用する場合があります。

Gcode-meta-compiler（gcmc）には、次のようなステートメントを含めることができます。include（"filename.inc.gcmc"）；デフォルトでは、gcmc には現在のディレクトリが含まれています。これは、linuxCNC の場合、linuxCNCini ファイルを含むディレクトリになります。 GCMC_INCLUDE_PATH アイテムを使用して、gcmc 検索順序の前に追加のディレクトリを追加できます。

サンプル軸-GUI ベースの INI

```
[RS274NGC]
...
SUBROUTINE_PATH = ../../nc_files/ngcgui_lib:../../ngcgui_lib/utilitysubs
USER_M_PATH = ../../nc_files/ngcgui_lib/mfiles
[DISPLAY]
TKPKG = Ngcgui 1.0
TKPKG = Ngcguittt 1.0
# Ngcgui must precede Ngcguittt
NGCGUI_FONT = Helvetica -12 normal
# specify filenames only, files must be in [RS274NGC]SUBROUTINE_PATH
NGCGUI_PREAMBLE = in_std.ngc
NGCGUI_SUBFILE = simp.ngc
NGCGUI_SUBFILE = xyz.ngc
NGCGUI_SUBFILE = iquad.ngc
NGCGUI_SUBFILE = db25.ngc
NGCGUI_SUBFILE = ihex.ngc
NGCGUI_SUBFILE = gosper.ngc
# specify "" for a custom tab page
NGCGUI_SUBFILE = ""
```

```
#NGCGUI_SUBFILE = "" use when image frame is specified if
# opening other files is required
# images will be put in a top level window
NGCGUI_OPTIONS =
#NGCGUI_OPTIONS = opt1 opt2 ...
# opt items:
# nonew -- disallow making a new custom tab
# noremove -- disallow removing any tab page
# noauto -- no auto send (makeFile, then manually send)
# noiframe -- no internal image, image on separate top level
GCMC_INCLUDE_PATH = /home/myname/gcmc_includes
TTT = truetype-tracer
TTT_PREAMBLE = in_std.ngc
PROGRAM_PREFIX = ../../nc_files
```

Note

上記は完全な軸の GUIINI ではありません。表示されている項目は ngcgui で使用されているものです。LinuxCNC では、完全な INI ファイルを作成するために多くの追加アイテムが必要です。

3.12.5.3 Truetype トレーサー

Ngcgui_ttt は、truetype-tracer（v4）のサポートを提供します。 これは、ユーザーがテキストを入力し、フォントやその他のパラメーターを選択した後、新しい ngcgui タブページを作成できるようにする軸タブページを作成します。（Truetype-tracer は個別にインストールする必要があります）。

軸に ngcgui_ttt を埋め込むには、ngcgui アイテムに加えて次のアイテムを指定します。

```
Item: [DISPLAY]TKPKG = Ngcgui_ttt version_number
Example: [DISPLAY]TKPKG = Ngcgui_ttt 1.0
Note: Mandatory, specifies loading of ngcgui_ttt in an axis tab page named ttt.
Must follow the TKPKG = Ngcgui item.
Item: [DISPLAY]TTT = path_to_truetype-tracer
Example: [DISPLAY]TTT = truetype-tracer
Note: Optional, if not specified, attempt to use /usr/local/bin/truetype-tracer  -
.
Specify with absolute pathname or as a simple executable name
in which case the user PATH environment will used to find the program.
Item: [DISPLAY]TTT_PREAMBLE = preamble_filename
Example: [DISPLAY]TTT_PREAMBLE = in_std.ngc
Note: Optional, specifies filename for preamble used for ttt created subfiles.
```

3.12.5.4 INI ファイルパスの仕様

Ngcgui は、linuxCNC 検索パスを使用してファイルを検索します。

検索パスは、次のように指定された標準ディレクトリで始まります。

```
[DISPLAY]PROGRAM_PREFIX = directory_name
```

次のように指定された複数のディレクトリが続きます。

```
[RS274NGC]SUBROUTINE_PATH = directory1_name:directory1_name:directory3_name ...
```

ディレクトリは、絶対パスまたは相対パスとして指定できます。

```
Example: [DISPLAY]PROGRAM_PREFIX = /home/myname/linuxcnc/nc_files
Example: [DISPLAY]PROGRAM_PREFIX = ~/linuxcnc/nc_files
Example: [DISPLAY]PROGRAM_PREFIX = ../../nc_files
```

「/」で始まる絶対パスは、完全なファイルシステムの場所を指定します。 「～/」で始まるパスは、ユーザーのホームディレクトリから始まるパスを指定します。 「～username /」で始まるパスは、ユーザー名のホームディレクトリで始まるパスを指定します。

相対パス相対パスは、INI ファイルを含むディレクトリである起動ディレクトリに基づいています。 相対パスを使用すると、構成の再配置が容易になりますが、Linux パス指定子について十分に理解している必要があります。

```
./d0 is the same as d0, e.g., a directory named d0 in the startup -
directory
../d1 refers to a directory d1 in the parent directory
../../d2 refers to a directory d2 in the parent of the parent directory
../../../d3 etc.
```

複数のディレクトリは、コロンで区切ることにより、[RS274NGC] SUBROUTINE_PATH で指定できます。 次の例は、複数のディレクトリの形式を示し、相対パスと絶対パスの使用法を示しています。

複数のディレクトリの例：

```
[RS274NGC]SUBROUTINE_PATH =
../../nc_files/ngcgui_lib:../../nc_files/ngcgui_lib/utilitysubs -
:/tmp/tmpngc‘
```

これは 1 つの長い行です。複数の行に続けないでください。 linuxCNC または ngcgui、あるいはその両方がファイルを検索するとき、検索で最初に見つかったファイルが使用されます。

LinuxCNC（および ngcgui）は、ngcgui サブファイル内から呼び出されるヘルパールーチンを含むすべてのサブルーチンを検索できる必要があります。 上記の例に示されているように、ユーティリティサブを別のディレクトリに配置すると便利です。

ディストリビューションには、ngcgui_lib ディレクトリと、プリアンブル、サブファイル、ポストアンブル、およびヘルパーファイルのデモファイルが含まれています。 ファイルの動作を変更するには、任意のファイルをコピーして、検索パスの前の部分に配置します。 最初に検索されるディレクトリは[DISPLAY]

PROGRAM_PREFIX です。 このディレクトリを使用できますが、専用のディレクトリを作成して、[RS274NGC] SUBROUTINE_PATH の先頭に配置することをお勧めします。

次の例では、/ home / myname / linuxcnc / mysubs 内のファイルは、.. / .. / nc_files / ngcgui_lib 内のファイルの前にあります。

ユーザーディレクトリの追加例：

```
[RS274NGC]SUBROUTINE_PATH =
/home/myname/linuxcnc/mysubs:../../nc_files/ngcgui_lib:../../  -
nc_files/ngcgui_lib/utilitysubs‘
```

新規ユーザーは、ngcgui 要件と互換性があるように構造化されていないファイルを誤って使用しようとする可能性があります。 Ngcgui は、ファイルがその規則に従ってコーディングされていない場合、多数のエラーを報告する可能性があります。 ngcgui 互換のサブファイルは、その目的専用のディレクトリに配置し、プリアンブル、ポストアンブル、およびヘルパーファイルは、サブファイルとして使用しないように別々のディレクトリに配置することをお勧めします。 サブファイルとしての使用を目的としていないファイルには、特別なコメント「（not_a_subfile）」を含めることができます。これにより、ngcgui は関連するメッセージとともにそれらを自動的に拒否します。

3.12.5.5 NGCGUI の使用に関する INI ファイルアイテムの詳細の概要

```
Item: [RS274NGC]SUBROUTINE_PATH = dirname1:dirname2:dirname3 ...
Example: [RS274NGC]SUBROUTINE_PATH = ../../nc_files/ngcgui_lib:../../nc_files/  -
ngcgui_lib/utilitysubs
Note: Optional, but very useful to organize subfiles and utility files
Item: [RS274NGC]USER_M_PATH = dirname1:dirname2:dirname3 ...
Example: [RS274NGC]USER_M_PATH = ../../nc_files/ngcgui_lib/mfiles
Note: Optional, needed to locate custom user mfiles
Item: [DISPLAY]EMBED_TAB_NAME = name to display on embedded tab page
Example: [DISPLAY]EMBED_TAB_NAME = Pyngcgui
Note: The entries: EMBED_TAB_NAME,EMBED_TAB_COMMAND,EMBED_TAB_LOCATION
define an embedded application for several linuxCNC guis
Item: [DISPLAY]EMBED_TAB_COMMAND = programname followed by arguments
Example: [DISPLAY]EMBED_TAB_COMMAND = gladevcp -x {XID} pyngcgui_axis.ui
Note: For gladevcp applications, see the <<cha:glade-vcp,GladeVCP Chapter>>
Item: [DISPLAY]EMBED_TAB_LOCATION = name_of_location
Example: [DISPLAY]EMBED_TAB_LOCATION = notebook_main
Note: See example INI files for possible locations
Not required for the axis gui
Item: [DISPLAY]PROGRAM_PREFIX = dirname
Example: [DISPLAY]PROGRAM_PREFIX = ../../nc_files
```

Note: Mandatory and needed for numerous linuxCNC functions
It is the first directory used in the search for files
item: [DISPLAY]TKPKG = Ngcgui version_number
Example: [DISPLAY]TKPKG = Ngcgui 1.0
Note: Required only for axis gui embedding, specifies loading of ngcgui axis -
tab pages
Item: [DISPLAY]NGCGUI_FONT = font_descriptor
Example: [DISPLAY]NGCGUI_FONT = Helvetica -12 normal
Note: Optional, font_descriptor is a tcl-compatible font specifier
with items for fonttype -fontsize fontweight
Default is: Helvetica -10 normal
Smaller font sizes may be useful for small screens
Larger font sizes may be helpful for touch screen applications
Item: [DISPLAY]NGCGUI_SUBFILE = subfile_filename
Example: [DISPLAY]NGCGUI_SUBFILE = simp.ngc
Example: [DISPLAY]NGCGUI_SUBFILE = square.gcmc
Example: [DISPLAY]NGCGUI_SUBFILE = ""
Note: Use one or more items to specify ngcgui-compatible
subfiles or gcmc programs that require a tab page on startup.
A "Custom" tab will be created when the filename is "".
A user can use a "Custom" tab to browse the file system
and identify preamble, subfile, and postamble files.
Item: [DISPLAY]NGCGUI_PREAMBLE = preamble_filename
Example: [DISPLAY]NGCGUI_PREAMBLE = in_std.ngc
Note: Optional, when specified, the file is prepended to a subfile.
Files created with "Custom" tab pages use the preamble specified
with the page.
Item: [DISPLAY]NGCGUI_POSTAMBLE = postamble_filename
Example: [DISPLAY]NGCGUI_POSTAMBLE = bye.ngc
Note: Optional, when specified, the file is appended to a subfiles.
Files created with "Custom" tab pages use the postamble specified
with the page.
Item: [DISPLAY]NGCGUI_OPTIONS = opt1 opt2 ...
Example: [DISPLAY]NGCGUI_OPTIONS = nonew noremove
Note: Multiple options are separated by blanks.
By default, ngcgui configures tab pages so that:
1) a user can make new tabs
2) a user can remove tabs (except for the last remaining one)
3) finalized files are automatically sent to linuxCNC
4) an image frame (iframe) is made available to display

an image for the subfile (if an image is provided)
5) the ngcgui result file sent to linuxCNC is terminated with
an m2 (and incurs m2 side-effects)
The options nonew, noremove, noauto, noiframe, nom2 respectively
disable these default behaviors.
By default, if an image (.png,.gif,jpg,pgm) file
is found in the same directory as the subfile, the
image is displayed in the iframe. Specifying
the noiframe option makes available additional buttons
for selecting a preamble, subfile, and postamble and
additional checkboxes. Selections of the checkboxes
are always available with special keys:
Ctrl-R Toggle "Retain values on Subfile read"
Ctrl-E Toggle "Expand subroutine"
Ctrl-a Toggle "Autosend"
(Ctrl-k lists all keys and functions)
If noiframe is specified and an image file is found,
the image is displayed in a separate window and
all functions are available on the tab page.
The NGCGUI_OPTIONS apply to all ngcgui tabs except that the
nonew, noremove, and noiframe options are not applicable
for "Custom" tabs. Do not use "Custom" tabs if you want
to limit the user's ability to select subfiles or create
additional tab pages.
Item: [DISPLAY]GCMC_INCLUDE_PATH = dirname1:dirname2:...
Example: [DISPLAY]GCMC_INCLUDE_PATH =
/home/myname/gcmc_includes:/home/myname/ -
gcmc_includes2
Note: Optional, each directory will be included when gcmc is invoked
using the option: --include dirname

3.12.6 NGCGUI 互換性のためのファイル要件

1.1.1.1 単一ファイル Gcode（.ngc）サブルーチン要件

NGCGUI 互換のサブファイルには、単一のサブルーチン定義が含まれています。 サブルーチンの名前はファイル名と同じである必要があります（.ngc サフィックスは含まれません）。 LinuxCNC は、名前付きまたは番号付きのサブルーチンをサポートしていますが、NGCGUI と互換性があるのは名前付きサブルーチンのみです。 詳細については、O コードの章を参照してください。

コメント以外の最初の行はサブステートメントである必要があります。 コメント以外の最後の行は、endsub ステートメントである必要があります。

Examp.ngc：

```
(info: info_text_to_appear_at_top_of_tab_page)
; comment line beginning with semicolon
( comment line using parentheses)
o<examp> sub
BODY_OF_SUBROUTINE
o<examp> endsub
; comment line beginning with semicolon
( comment line using parentheses)
```

サブルーチンの本体は、サブルーチン呼び出しで予期される各位置パラメーターのローカル名前付きパラメーターを定義する一連のステートメントで開始する必要があります。 これらの定義は、＃1 で始まり、最後に使用されたパラメーター番号で終わる連続している必要があります。 これらのパラメーターごとに定義を提供する必要があります（省略なし）。

パラメータの番号付け

```
#<xparm> = #1
#<yparm> = #2
#<zparm> = #3
```

LinuxCNC は、＃1 から＃30 の範囲のすべての番号付きパラメーターを呼び出しパラメーターと見なすため、ngcgui は、この範囲内のパラメーターが発生した場合の入力ボックスを提供します。 サブルーチン内の他の場所では、番号付きパラメーター＃1 から＃30 の使用を避けることをお勧めします。 すべての内部変数には、ローカルの名前付きパラメーターを使用することをお勧めします。

各定義ステートメントには、オプションで特別なコメントとパラメーターのデフォルト値を含めることができます。

ステートメントプロトタイプ

```
#<vname> = #n (=default_value)
or
#<vname> = #n (comment_text)
or
#<vname> = #n (=default_value comment_text)
```

パラメータの例

```
#<xparm> = #1 (=0.0)
#<yparm> = #2 (Ystart)
#<zparm> = #3 (=0.0 Z start setting)
```

default_value が指定されている場合、起動時にパラメーターの入力ボックスに入力されます。

comment_text が含まれている場合、パラメーター名の代わりに入力を識別するために使用されます。

グローバル名前付きパラメーターグローバル名前付きパラメーターと ngcgui に関する注意事項：

（グローバルな名前付きパラメーターには、#<_ someglobalname>のように、名前の先頭にアンダースコアがあります）

多くのプログラミング言語と同様に、グローバルの使用は強力ですが、予期しない結果を招くことがよくあります。 LinuxCNC では、既存のグローバル名前付きパラメーターはサブルーチンの実行時に有効になり、サブルーチンはグローバル名前付きパラメーターを変更または作成できます。

グローバルな名前付きパラメーターを使用してサブルーチンに情報を渡すことはお勧めしません。そのような使用には、維持が困難な明確に定義されたグローバルコンテキストの確立と維持が必要になるためです。 サブルーチン入力として番号付きパラメーター＃1 から＃30 を使用することで、幅広い設計要件を満たすのに十分なはずです。

入力グローバル名前付きパラメーターは推奨されませんが、linuxCNC サブルーチンは、結果を返すためにグローバル名前付きパラメーターを使用する必要があります。 ngcgui 互換のサブファイルは GUI の使用を目的としているため、戻り値は一般的な要件ではありません。 ただし、ngcgui は、グローバルな名前付きパラメーターを返すサブルーチンのテストツールとして役立ち、ngcgui 互換のサブファイルは、グローバルな名前付きパラメーターで結果を返すユーティリティサブルーチンファイルを呼び出すのが一般的です。

これらの使用法をサポートするために、ngcgui は、名前にコロン（:)文字を含むグローバルな名前付きパラメーターを無視します。 名前にコロン（:)を使用すると、ngcgui がこれらのパラメーターのエントリボックスを作成できなくなります。

グローバル名前付きパラメーター

```
o<examp> sub
...
#<_examp:result> = #5410 (return the current tool diameter)
...
o<helper> call [#<x1>] [#<x2>] (call a subroutine)
#<xresult> = #<_helper:answer> (immediately localize the helper global result)
#<_helper:answer> = 0.0 (nullify global named parameter used by subroutine)
...
o<examp> endsub
```

上記の例では、ユーティリティサブルーチンは helper.ngc という名前の別のファイルにあります。 ヘルパールーチンは、#<_ helper：answer という名前のグローバルな名前付きパラメーターで結果を返します。

良い習慣として、呼び出し元のサブファイルは、サブファイルの他の場所で使用するために結果をすぐにローカライズし、結果を返すために使用されるグローバルな名前付きパラメーターは、グローバルコンテキストの他の場所での不注意な使用を軽減するために無効にされます。 （0.0 の無効化値は常に良い選択であるとは限りません）。

Ngcgui は、サブファイルおよび複数のサブファイルの複数の機能の作成と連結をサポートします。 ngcgui がサブルーチン内でテストできる特別なグローバルパラメータを挿入するように、サブファイルが実行時にそれらの順序を決定することが役立つ場合があります。 パラメータの名前は#<_ feature：>です。 その値は 0 の値で始まり、追加された機能ごとに増分されます。

追加機能特別な情報コメントは、ngcgui 互換のサブファイルのどこにでも含めることができます。 形式は次のとおりです。

```
(info: info_text)
```

info_text は、軸の ngcgui タブページの上部近くに表示されます。

サブファイルとしての使用を目的としていないファイルには、特別なコメントを含めることができるため、ngcgui は関連するメッセージとともにそれらを自動的に拒否します。

```
(not_a_subfile)
```

オプションの画像ファイル（.png、.gif、.jpg、.pgm）をサブファイルに添付できます。 画像ファイルは、サブファイルで使用されるパラメータを明確にするのに役立ちます。 画像ファイルはサブファイルと同じディレクトリにあり、同じ名前に適切な画像サフィックスが付いている必要があります。 サブファイル example.ngc には、画像ファイル examp.png を付けることができます。 Ngcgui は、最大幅 320 ピクセル、最大高さ 240 ピクセルのサイズにサブサンプリングすることにより、大きな画像のサイズを変更しようとします。

ngcgui 互換のサブファイルを作成するために必要な規則は、LinuxCNC の汎用サブルーチンファイルとしての使用を妨げるものではありません。

LinuxCNC ディストリビューションには、LinuxCNC サブルーチンの機能と ngcgui の使用法を説明するために、ngcgui 互換のサブファイルとユーティリティファイルの両方を含むライブラリ（ngcgui_lib ディレクトリ）が含まれています。 別のライブラリ（gcmc_lib）は、Gcode メタコンパイラ（gcmc）のサブルーチンファイルの例を提供します。

追加のユーザーが送信したサブルーチンは、フォーラムのサブルーチンセクションにあります。

3.12.6.1 Gcode-メタコンパイラ（.gcmc）ファイルの要件

Gcode-meta-compiler（gcmc）のファイルは、ngcgui によって読み取られ、ファイルでタグ付けされた変数の入力ボックスを作成します。 ファイルの機能が完成すると、ngcgui はファイルを入力として gcmc コンパイラに渡し、コンパイルが成功すると、結果の gcode ファイルが linuxCNC に送信されて実行されます。 結果のファイルは、単一ファイルのサブルーチンとしてフォーマットされます。 .gcmc ファイルと.ngc ファイルは ngcgui によって混在させることができます。

ngcgui に含めるために識別された変数は、gcmc コンパイラーへのコメントとして表示される行でタグ付けされます。

可変タグ形式の例

```
//ngcgui: varname1 =
//ngcgui: varname2 = value2
```

```
//ngcgui: varname3 = value3, label3;
```

例：

```
//ngcgui: zsafe =
//ngcgui: feedrate = 10
//ngcgui: xl = 0, x limit
```

これらの例では、varname1 の入力ボックスにはデフォルトがなく、varname2 の入力ボックスにはデフォルトの value2 があり、varname 3 の入力ボックスにはデフォルトの値 3 とラベル label3（varname3 ではなく）があります。 デフォルト値は数字でなければなりません。

gcmc ファイルの有効な行を簡単に変更できるようにするために、別のタグ行形式を使用できます。 代替形式では、末尾のセミコロン（;）と末尾のコメントマーカー（//）が無視されます。この規定により、多くの場合、.gcmc ファイルの既存の行に// ngcgui：タグを追加するだけで済みます。

代替可変タグ形式

```
//ngcgui: varname2 = value2;
//ngcgui: varname3 = value3; //, label3;
```

例：

```
//ngcgui: feedrate = 10;
//ngcgui: xl = 0; //, x limit
```

タブページの上部に表示される情報行は、オプションで次のタグが付けられた行に含めることができます。

情報タグ

```
//ngcgui: info: text_to_appear_at_top_of_tab_page
```

必要に応じて、次のタグが付けられた行を使用して、オプションを gcmc コンパイラに渡すことができます。

オプションラインタグフォーマット

```
//ngcgui: -option_name [ [=] option_value]
```

例：

```
//ngcgui: -I
//ngcgui: --imperial
//ngcgui: --precision 5
//ngcgui: --precision=6
```

gcmc のオプションは、terminal コマンドで使用できます。

```
gcmc –help
```

gcmc プログラムは、デフォルトでメートル法を使用します。 モードは、オプション設定でインチに設定できます。

```
//ngcgui: --imperial
```

プリアンブルファイルを使用すると、gcmc ファイルで使用されるモードと競合するモード（g20 または g21）を設定できます。 gcmc プログラムモードが有効になっていることを確認するには、.gcmc ファイルに次のステートメントを含めます。

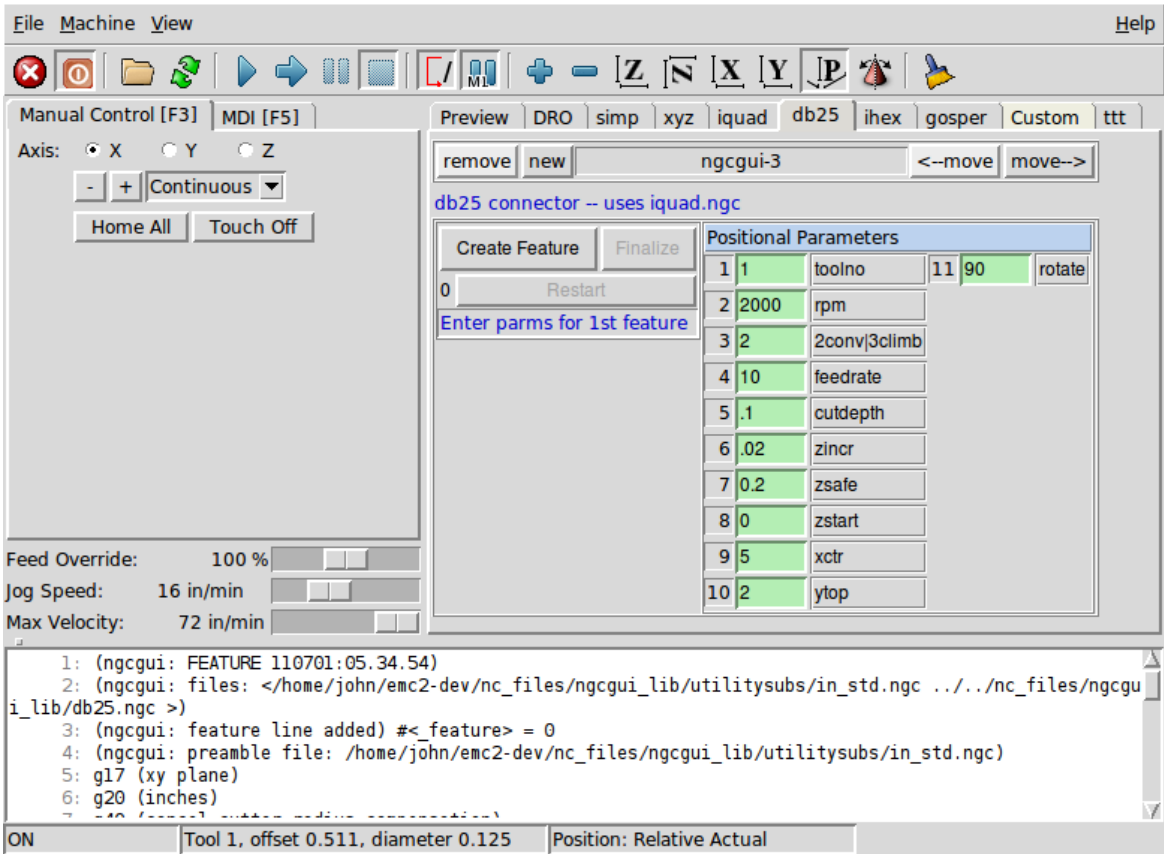
```
include("ensure_mode.gcmc")
```

次に、ini ファイル内の gcmcinclude_files に適切なパスを指定します。次に例を示します。

```
[DISPLAY]
GCMC_INCLUDE_PATH = ../../nc_files/gcmc_lib
```

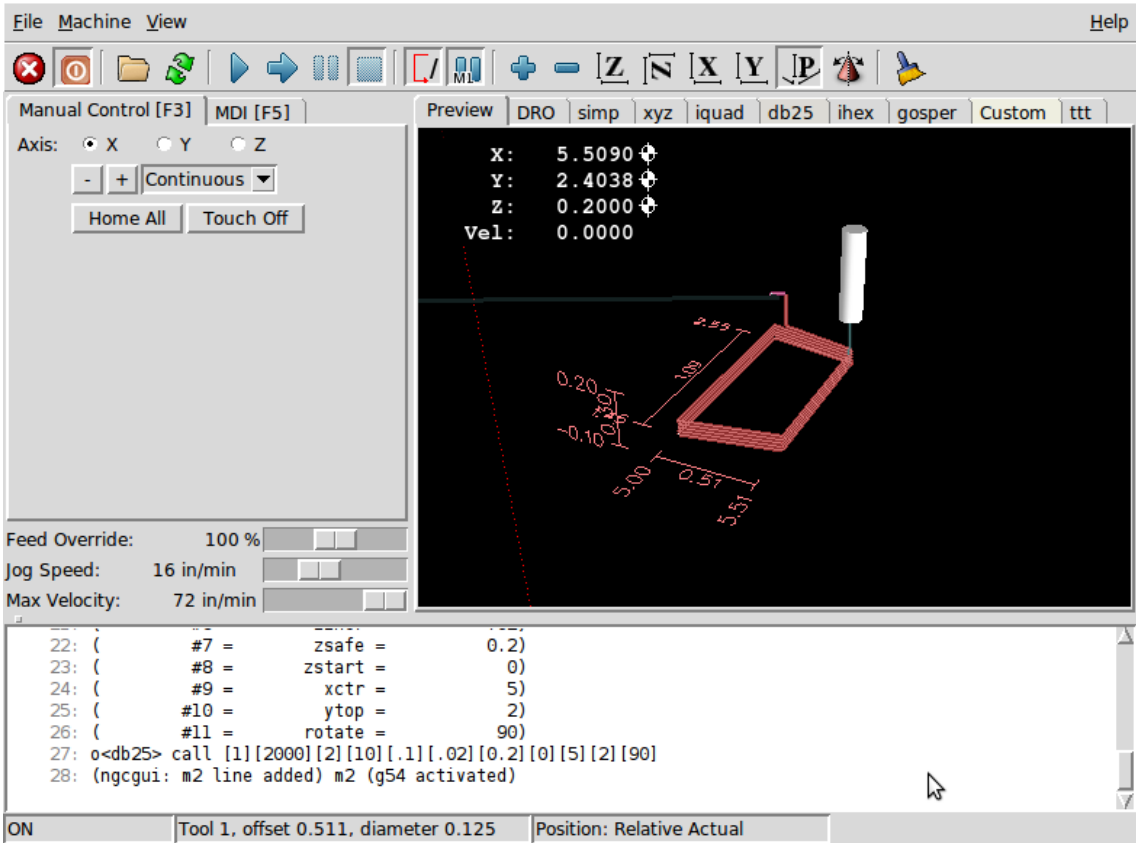
3.12.7 DB25 の例

以下に DB25 サブルーチンを示します。 最初の写真では、各変数の空白を埋める場所がわかります。

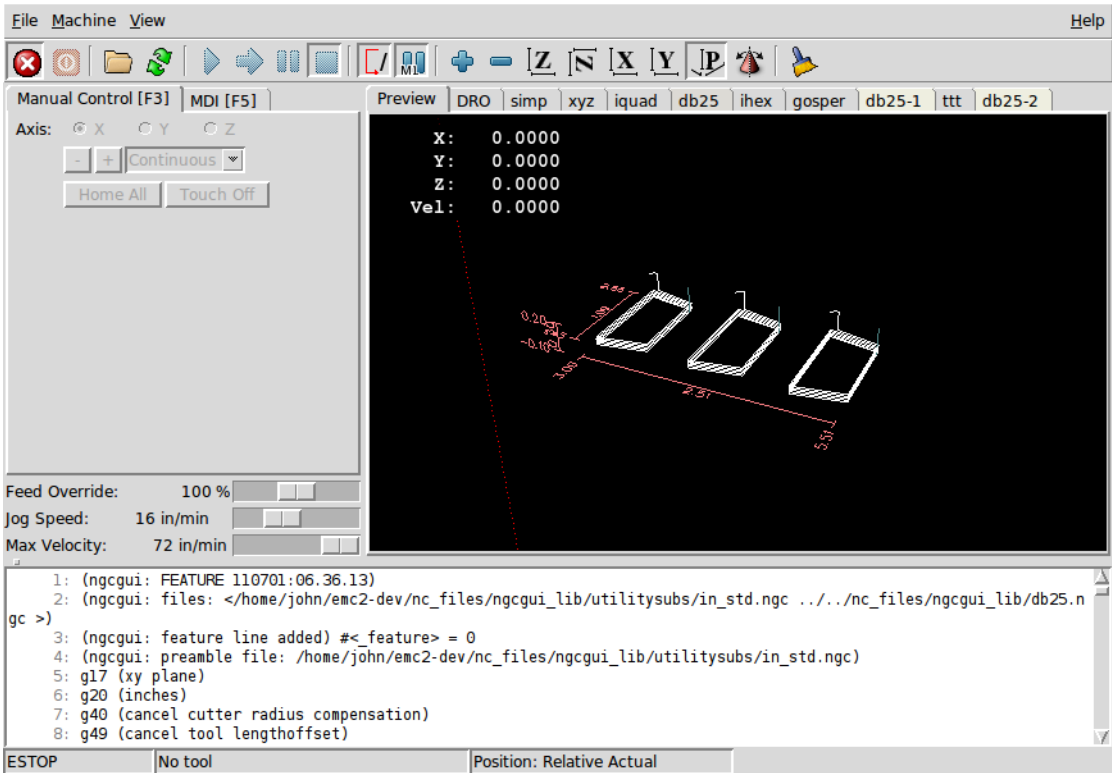


この写真は、DB25 サブルーチンのバックプロットを示しています。

LinuxCNC V2.8.1-84-g81a16a1fd, 2021-03-29



この写真は、新しいボタンとカスタムタブを使用して、1つのプログラムで3つのDB25カットアウトを作成する方法を示しています。



3.13 Touchy GUI

Touchy は、マシンのコントロールパネルで使用することを目的とした LinuxCNC のユーザーインターフェイスであるため、キーボードやマウスは必要ありません。

タッチスクリーンでの使用を目的としており、ホイール/ MPG およびいくつかのボタンとスイッチと組み合わせて機能します。

[ハンドホイール]タブには、ホイール/ MPG 入力のフィードオーバーライド、スピンドルオーバーライド、最大速度、およびジョギング機能を選択するためのラジオボタンがあります。 軸選択用のラジオボタンとジョギング用の増分も提供されます。

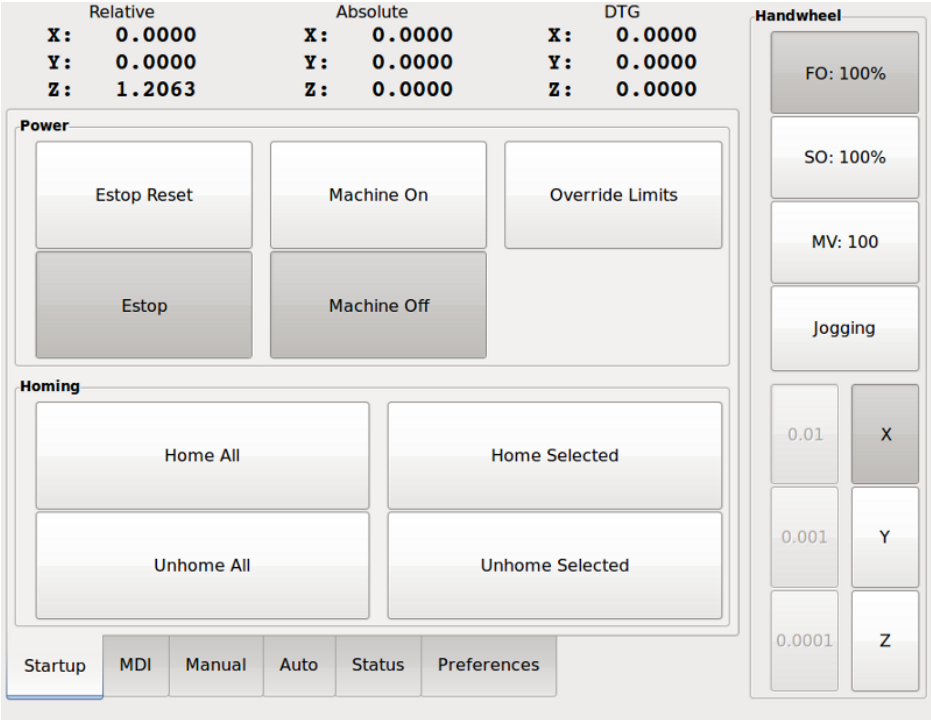


図 4-18

3.13.1 パネル構成

1.1.1.1 HAL 接続

Touchy では、コントロールを接続するために、構成ディレクトリ（ini ファイルが存在するディレクトリ）に touchy.hal という名前のファイルを作成する必要があります。 Touchy は、独自のピンを接続に使用できるようにした後、このファイルの HAL コマンドを実行します。

HAL ファイルと net コマンドの詳細については、HAL の基本を参照してください。

Touchy には、ホイールのジョギングを制御するためにモーションコントローラーに接続するための出力ピンがいくつかあります。

- touchy.jog.wheel.increment、これは axis.N.jog-scale ピンの各軸 N に接続されます。
- touchy.jog.wheel.N。これは axis.N.jog に接続されます-各軸 N に対して有効にします。

N は軸番号 0〜8 を表します。

- touchy.wheel-counts に接続するだけでなく、wheelcounts も axis.N.jog-countsfor each axis N に接続する必要があります。HAL コンポーネント ilowpass を使用してホイールジョギングをスムーズにする場合は、必ず軸のみをスムーズにしてください。 N.jog-counts であり、touchy.wheel-counts ではありません。

必要な管理

- HAL ピン touchy.abort に接続された中止ボタン（瞬間的な接触）
- touchy.cycle-start に接続されたサイクルスタートボタン（瞬間的な接触）
- 上記のように touchy.wheel-counts とモーションピンに接続されたホイール/ MPG
- touchy.single-block に接続されたシングルブロック（トグルスイッチ）

オプションのコントロール

- 連続ジョグの場合、各軸に 1 つのセンターオフ双方向モーメンタリトグル（または 2 つのモーメンタリボタン）があり、touchy.jog.continuous.x.negative、touchy.jog.continuous.x.positive などにフックされています。
- クイルアップボタンが必要な場合（Z を最高速度でトラベルのトップにジョギングするため）、touchy.quill-up に接続された瞬間的なボタン。

オプションのパネルランプ

- touchy.jog.active は、パネルのジョギングコントロールがライブになっていることを示します
- touchy.status-indicator は、マシンが G コードを実行しているときにオンになり、マシンが実行されているが一時停止/フィードホールドにあるときに点滅します。

3.13.1.1 あらゆるセットアップに推奨

- estop チェーンに配線された Estop ボタン

3.13.2 設定

1.1.1.1 Touchy を有効にする

Touchy を使用するには、ini ファイルの[DISPLAY]セクションで、表示セレクター行を DISPLAY = touchy に変更します。

3.13.2.1 4.4.2.2 設定

Touchy を初めて起動するときは、[設定]タブを確認してください。 タッチスクリーンを使用している場合は、最良の結果を得るためにポインタを非表示にするオプションを選択してください。

StatusWindow は固定の高さで、固定フォントのサイズによって設定されます。 これは、システム/環境設定/外観/フォント/詳細で構成されている GnomeDPI の影響を受ける可能性があります。 画面の下部が途切れる場合は、DPI 設定を下げてください。

他のすべてのフォントサイズは、[設定]タブで変更できます。

3.13.2.2 マクロ

Touchy は、MDI インターフェイスを使用して O-word マクロを呼び出すことができます。 これを構成するには、ini ファイルの[TOUCHY]セクションで、1 つ以上の MACRO 行を追加します。 それぞれの形式である必要があります

MACRO = increment xinc yinc

この例では、increment はマクロの名前であり、xinc と yinc という名前の 2 つのパラメーターを受け入れます。

次に、マクロを、increment.ngc という名前のファイル、PROGRAM_PREFIX ディレクトリ、または SUBROUTINE_PATH の任意のディレクトリに配置します。

次のようになります。

```
O<increment> sub
G91 G0 X#1 Y#2
G90
O<increment> endsub
```

サブの名前は、大文字と小文字を含めて、ファイル名とマクロ名と完全に一致することに注意してください。

Touchy の[MDI]タブの[マクロ]ボタンを押してマクロを呼び出すと、xinc と yinc の値を入力できます。 これらは、それぞれ＃1 および＃2 としてマクロに渡されます。 空のままにしたパラメーターは、値 0 として渡されます。

複数の異なるマクロがある場合は、マクロボタンを繰り返し押してそれらを循環させます。

この簡単な例では、xinc に-1 を入力してサイクル開始を押すと、G0 の高速移動が呼び出され、1 ユニットが左に移動します。

このマクロ機能は、エッジ/ホールプロービングやその他のセットアップタスク、および特別に作成された gcode プログラムを必要とせずにパネルから実行できるホールミリングやその他の簡単な操作に役立ちます。

3.14 Gscreen

3.14.1 イントロ

Gscreen は、LinuxCNC を制御するためのカスタム画面を表示するためのインフラストラクチャです。 Gscreen は GladeVCP から多額の借用をしています。 Glade- VCP は、GTK ウィジェットエディター GLADE を使用して、ポイントアンドクリックで仮想コントロールパネル（VCP）を構築します。 Gscreen は、これを Python プログラミングと組み合わせて、CNC マシンを実行するための GUI 画面を作成します。

さまざまなボタンやステータス LED が必要な場合は、Gscreen をカスタマイズできます。 Gscreen は、コントロールとインジケーターを追加するために使用される GladeVCP をサポートします。 Gscreen をカスタマイズす

るには、Glade エディターを使用します。 Gscreen は、右側にカスタムパネルを追加したり、完全に編集可能なカスタムタブを追加したりすることに制限されていません。

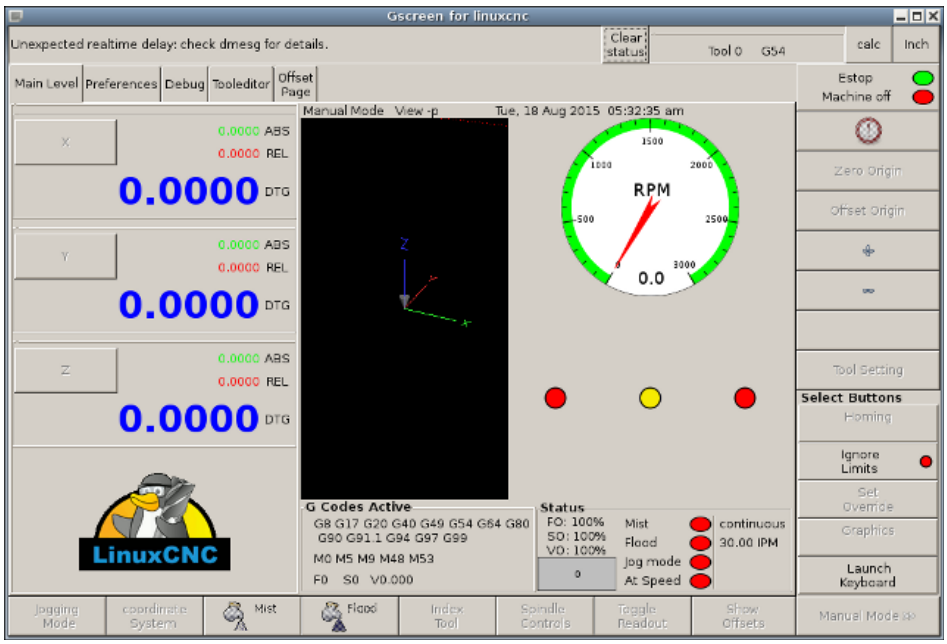


図 4-19

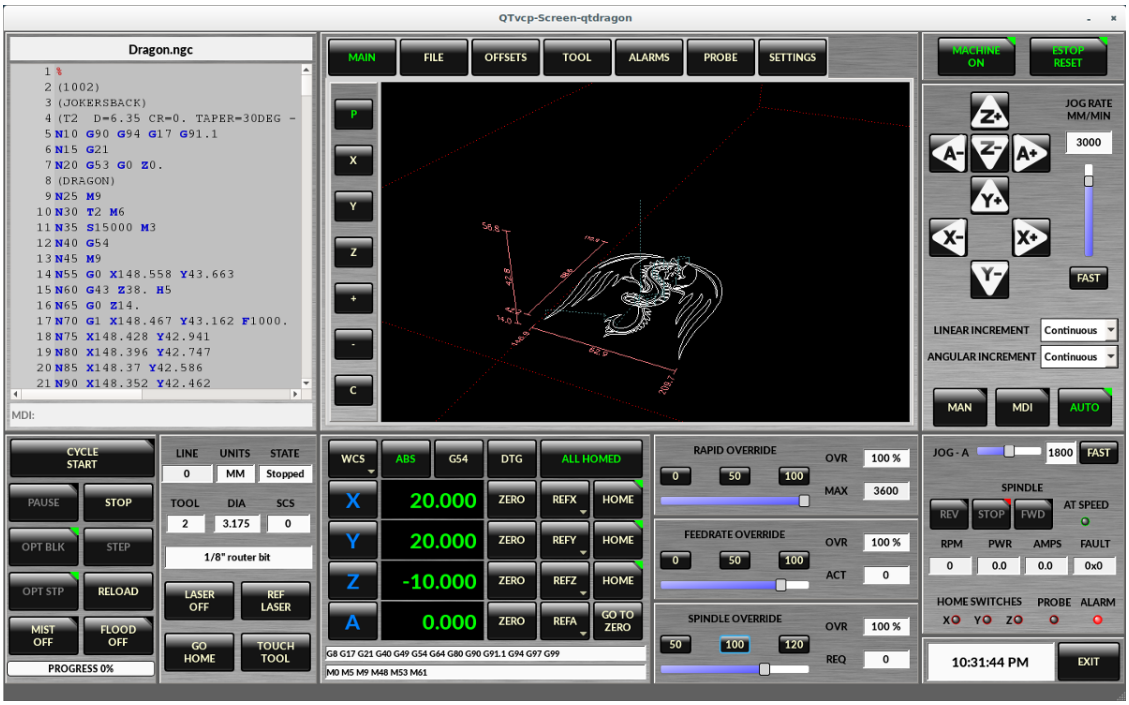


図 4-20

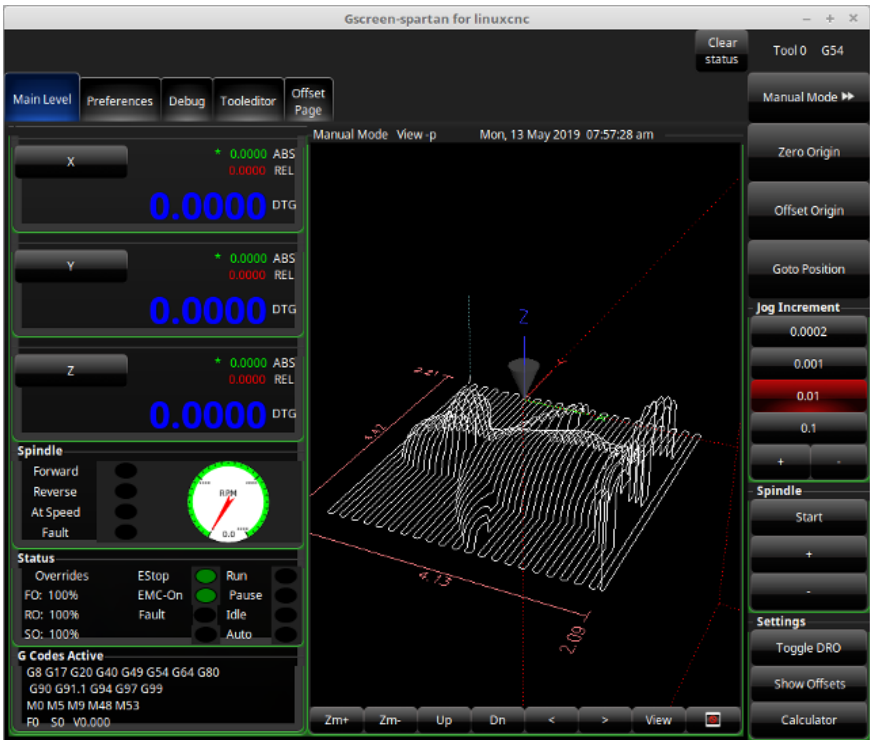

Gscreen-spartan for linuxcnc
Clear status
Tool 0 G54
Main Level
Preferences
Debug
Tooleditor
Offset Page
Manual Mode
Manual Mode View -p
Mon, 13 May 2019 07:57:28 am
Zero Origin
Offset Origin
Goto Position
Jog Increment
0.0002
0.001
0.01
0.1
Spindle
Start
Settings
Toggle DRO
Show Offsets
Calculator
X
Y
Z
0.0000 ABS
0.0000 REL
0.0000 DTG
Spindle
Forward
Reverse
At Speed
Fault
Status
Overrides
FO: 100%
RO: 100%
SO: 100%
EStop
EMC-On
Fault
Run
Pause
Idle
Auto
G Codes Active
G8 G17 G20 G40 G49 G54 G64 G80
G90 G91.1 G94 G97 G99
M0 M5 M9 M48 M53
F0 S0 V0.000
Zm+
Zm-
Up
Dn
<
>
View

図 4-21

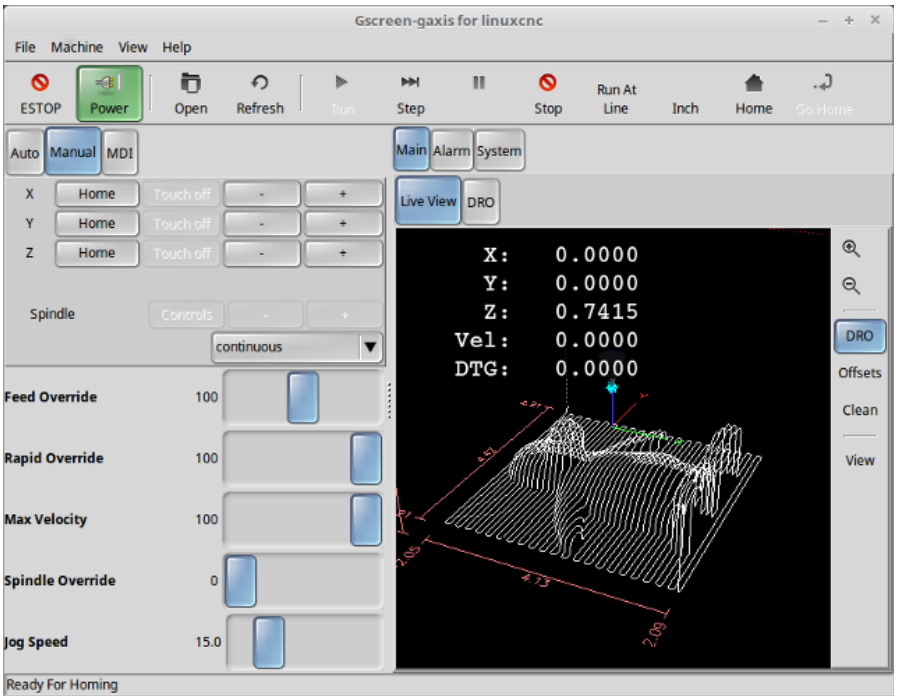

Gscreen-gaxis for linuxcnc
File Machine View Help
ESTOP
Power
Open
Refresh
Run
Step
Stop
Run At Line
Inch
Home
Go Home
Auto
Manual
MDI
Main
Alarm
System
X Home Touch off
Y Home Touch off
Z Home Touch off
Live View
DRO
X: 0.0000
Y: 0.0000
Z: 0.7415
Vel: 0.0000
DTG: 0.0000
Spindle Controls
continuous
Feed Override 100
Rapid Override 100
Max Velocity 100
Spindle Override 0
Jog Speed 15.0
DRO
Offsets
Clean
View
Ready For Homing

図 4-22

図 4-23

Gscreen は、Glade（エディター）、PyGTK（ウィジェットツールキット）、および GladeVCP（LinuxCNC の Glade および PyGTK への接続）に基づいています。 GladeVCP には、LinuxCNC 専用に追加されたいくつかの特別なウィジェットとアクションがあります。ウィジェットは、PyGTK ツールキットのボタン、スライダー、ラベルなどに使用される一般的な名前です。

1.1.1.1 グレイドファイル

Glade ファイルは、画面のレイアウトとウィジェットを説明する XML 標準で編成されたテキストファイルです。 PyGTK はこのファイルを使用して、これらのウィジェットを実際に表示して反応します。 Glade エディターを使用すると、このファイルの作成と編集が比較的簡単になります。GTK2 ウィジェットを使用する Glade3.8.0 エディターを使用する必要があります。 Linux の新しいバージョンは、GTK3 ウィジェットを使用する最新の Glade エディターを使用します。 その場合、リポジトリから古い Glade エディターをダウンロードする必要があります。

3.14.1.1 PyGTK

PyGTK は、GTK への Python バインディングです。 GTK はビジュアルウィジェットのツールキットであり、C でプログラミングされています。PyGTK は Python を使用して GTK とバインドします。

3.14.2 GladeVCP

GladeVCP は、LinuxCNC、HAL、PyGTK、Glade をすべて結合します。 LinuxCNC にはいくつかの特別なウィジェットが必要なので、GladeVCP がそれらを提供します。 多くは、既存の PyGTK ウィジェットに対する単なる HAL 拡張です。 GladeVCP は、Glade ファイルに記述されている特別なウィジェットの HAL ピンを作成します。 GladeVCP では、ウィジェットと対話するための Python コマンドを追加して、ウィジェットがデフォルトの形式では利用できないことを実行できるようにすることもできます。GladeVCP パネルを作成できる場合は、Gscreen をカスタマイズできます。

1.1.1.1 概要

カスタマイズを追加するために個別にまたは組み合わせて使用できる 2 つのファイルがあります。 ローカル空き地ファイルとハンドラーファイル。 通常、Gscreen はストック Glade ファイルと、場合によってはハンドラーファイルを使用します（サンプルスキンを使用している場合）。Gscreen を指定して、ローカルの Glade ファイルとハンドラーファイルを使用できます。 Gscreen は、選択した構成のすべての構成ファイルを保持するフォルダーを調べます。

ローカル Glade ファイル存在する場合、ストック Glade ファイルの代わりに、構成フォルダー内のローカル Glade ファイルがロードされます。 ローカル Glade ファイルを使用すると、デフォルトの画面ではなく、カスタマイズしたデザインを使用できます。 INI ファイルには、ベース名を設定するためのスイッチがあります。-cname は、Gscreen が MYNAME.glade と MYNAME_handler.py を検索するようにします。

Gscreen に、Glade ファイルをロードするだけで、内部信号をファイルに接続しないように指示できます。 これにより、gscreen は GTK ビルダーで保存された Glade ファイルをロードできます。これは、完全にカスタムの画面を表示できることを意味しますが、ハンドラーファイルを使用する必要もあります。 Gscreen は、Glade ファイルを使用してウィジェットを定義するため、ウィジェットを表示して操作できます。 それらの多くには特定の名前があり、他の人には一般的な名前が付けられた Glade があります。 ウィジェットが表示されるが変更されない場合は、一般名で問題ありません。 ウィジェットを制御または操作する必要がある場合は、目的のある名前が付けられます（すべての名前は一意である必要があります）。 ウィジェットには、GLADE エディターでウィジェット用に定義されたシグナルを含めることもできます。 与えられる信号と呼び出すメソッドを定義します。

ストックスキンの変更ウィジェットの名前を変更すると、Gscreen がウィジェットを見つけられない場合があります。 このウィジェットが Python コードから参照されている場合、せいぜいウィジェットが機能しなくなり、最悪の場合、Gscreen のデフォルト画面をロードするときにエラーが発生します。エディタで定義された多くの信号を使用せず、Python コードで定義します。 シグナルのあるウィジェットを移動（カットアンドペースト）すると、シグナルはコピーされません。 手動で再度追加する必要があります。

ハンドラーファイルハンドラーファイルは、Gscreen がデフォルトのルーチンに追加する Python コードを含むファイルです。ハンドラーファイルを使用すると、Gscreen を適切に変更しなくても、デフォルトを変更したり、Gscreen スキンにロジックを追加したりできます。新しい関数を Gscreen の関数と組み合わせて、必要に応じて動作を変更できます。 Gscreen のすべての機能を完全にバイパスして、まったく異なる動作をする場合は作成できます。存在する場合、gscreen_handler.py（または INI スイッチを使用している場合は MYNAME_handler.py）という名前のハンドラーファイルがロードおよび登録されます。Gscreen はハンドラーファイルを検索し、見つかった場合は特定の関数名を検索して代わりに呼び出します。デフォルトのものの。ウィジェットを追加する場合は、Glade エディターからシグナル呼び出しを設定して、ハンドラーファイルに記述したルーチンを呼び出すことができます。このようにして、カスタム動作を行うことができます。ハンドラールーチンは、独自のコードを実行する前または後に、Gscreen のデフォルトルーチンを呼び出すことができます。このようにして、サウン

ドの追加などの追加の動作に取り組むことができます。 GladeVCP ハンドラーファイルの基本については、GladeVCP の章を参照してください。 Gscreen は非常によく似た手法を使用しています。

テーマ Gscreen は、PyGTK ツールキットを使用して画面を表示します。 Pygtk は、GTK にバインドされている Python 言語です。 GTK はテーマをサポートしています。テーマは、画面上のウィジェットのルックアンドフィールを変更する方法です。たとえば、ボタンやスライダーの色やサイズは、テーマを使用して変更できます。 Web 上には多くの GTK2 テーマがあります。テーマをカスタマイズして、特定の名前付きウィジェットのビジュアルを変更することもできます。これにより、テーマファイルと空き地ファイルがより緊密に結び付けられます。一部のサンプル画面スキンを使用すると、ユーザーはシステム上の任意のテーマを選択できます。サンプルの gscreen は一例です。設定ファイルに同じ名前のテーマをロードするものもあります。サンプルの gscreen-gaxis は一例です。これは、テーマフォルダを INI ファイルと HAL ファイルがある config フォルダに置き、次の名前を付けることで実行されます。SCREENNAME_theme（SCREENNAME はファイルのベース名です。例：gaxis_theme）このフォルダ内には、gtk-2.0 という別のフォルダがあります。テーマファイルです。このファイルを追加すると、Gscreen は起動時にデフォルトでこのテーマになります。 gscreen-gaxis には、特定の名前付きウィジェットを検索し、それらの特定のウィジェットの視覚的動作を変更するサンプルカスタムテーマがあります。 Estop ボタンと machine-on ボタンは、他のボタンとは異なる色を使用して、目立つようにしています。これは、テーマの gtkrc ファイルに特定のコマンドを追加することにより、ハンドラーファイルで特定の名前を付けることによって行われます。 GTK2 テーマに関する情報（サンプルテーマはピックスマップテーマエンジンを使用）については、以下を参照してください。

GTK テーマ

Pixmap テーマエンジン

3.14.2.1 GladeVCP パネルを作成する

Gscreen は、それを制御する Python コードを備えた、非常に複雑な GladeVCP パネルです。 カスタマイズするには、Glade エディターに Glade ファイルをロードする必要があります。

インストールされている LinuxCNCUbuntu10.04 に LinuxCNC2.6 +がインストールされている場合は、アプリケーションメニューまたはターミナルから Glade エディターを起動するだけです。 Linux の新しいバージョンでは、Glade 3.8.0〜3.8.6 をインストールする必要があります（自分でコンパイルする必要がある場合があります）。

RIP コンパイル済みコマンド LinuxCNC のソースバージョンからコンパイル済みを使用して、ターミナルを開き、Linux-CNC フォルダーの先頭に移動します。 を入力して環境を設定します。 ./scripts/rip-environment が空き地に入り、ターミナルに無視できる警告が表示され、エディターが開くはずです。 在庫の GscreenGlade ファイルは次の場所にあります：src / emc / usr_intf / gscreen /サンプルスキンは/ share / gscreen / skins /にあります。これを構成フォルダーにコピーする必要があります。 または、構成フォルダーに保存して、クリーンシートの Glade ファイルを作成することもできます。

これで、ストックの Glade ファイルが読み込まれ、編集できるようになりました。 最初に気付くのは、Gscreen がボタンのすべてのボックスを非表示にし、モードに応じてボタンを変更するなど、いくつかのトリックを使用しているように、エディターで表示されているように見えないことです。 ノートブックについても同じ

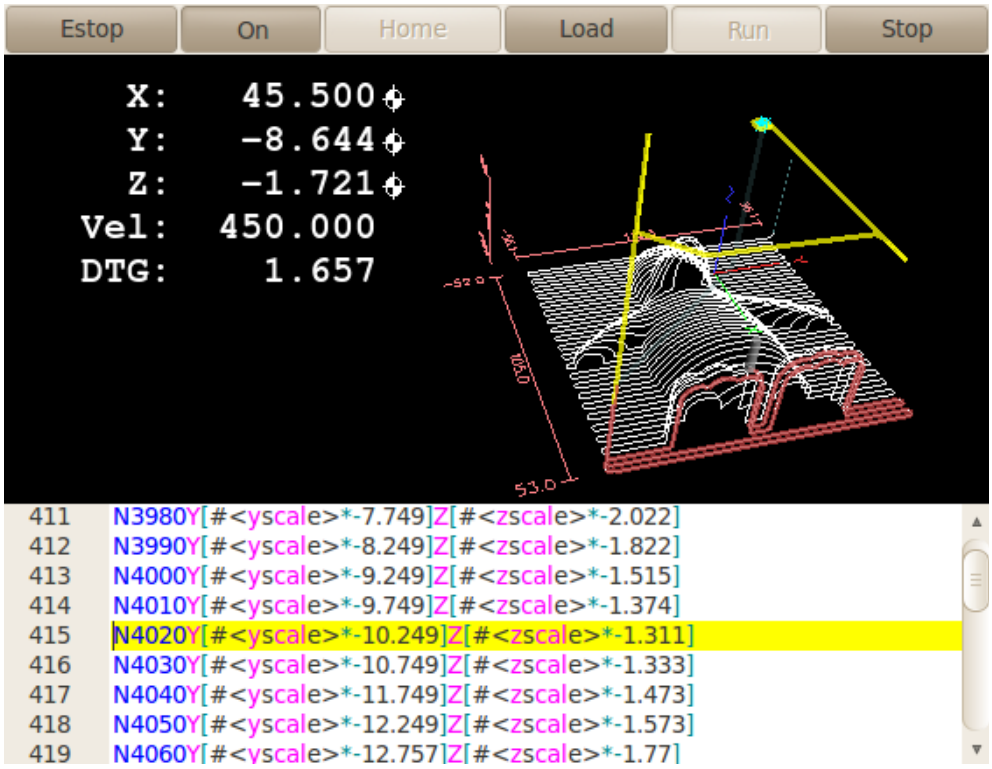
ことが言えます。一部の画面では、タブが表示されていないノートブックを使用しています。 エディタでページを変更するには、それらのタブを一時的に表示する必要があります。

変更を加える場合は、ウィジェットを追加してからウィジェットを減算し、画面を適切に機能させてオブジェクトを非表示にする方がはるかに簡単です。これは、エラーが発生せずに表示を変更する 1 つの方法です。 これは常に機能するとは限りません。一部のウィジェットは再び表示されるように設定されます。 Gscreen の通常のウィジェットの名前を変更することは、Python コードを変更しないとうまく機能しない可能性がありますが、名前を維持したままウィジェットを移動することは通常は機能します。

Gscreen は、Python コードの追加を回避するために、GladeVCP ウィジェットを可能な限り活用します。 GladeVCP ウィジェットについて学ぶことは前提条件です。 既存のウィジェットが必要な機能を提供する場合は、Python コードを追加する必要はなく、Glade ファイルを構成フォルダーに保存するだけです。 もっとカスタムなものが必要な場合は、Python プログラミングを行う必要があります。 親ウィンドウの名前は window1 である必要があります。 Gscreen はこの名前を想定しています。

カスタム画面オプションを使用する場合は、LinuxCNC を更新するときに（必要に応じて）修正する必要があることを忘れないでください。

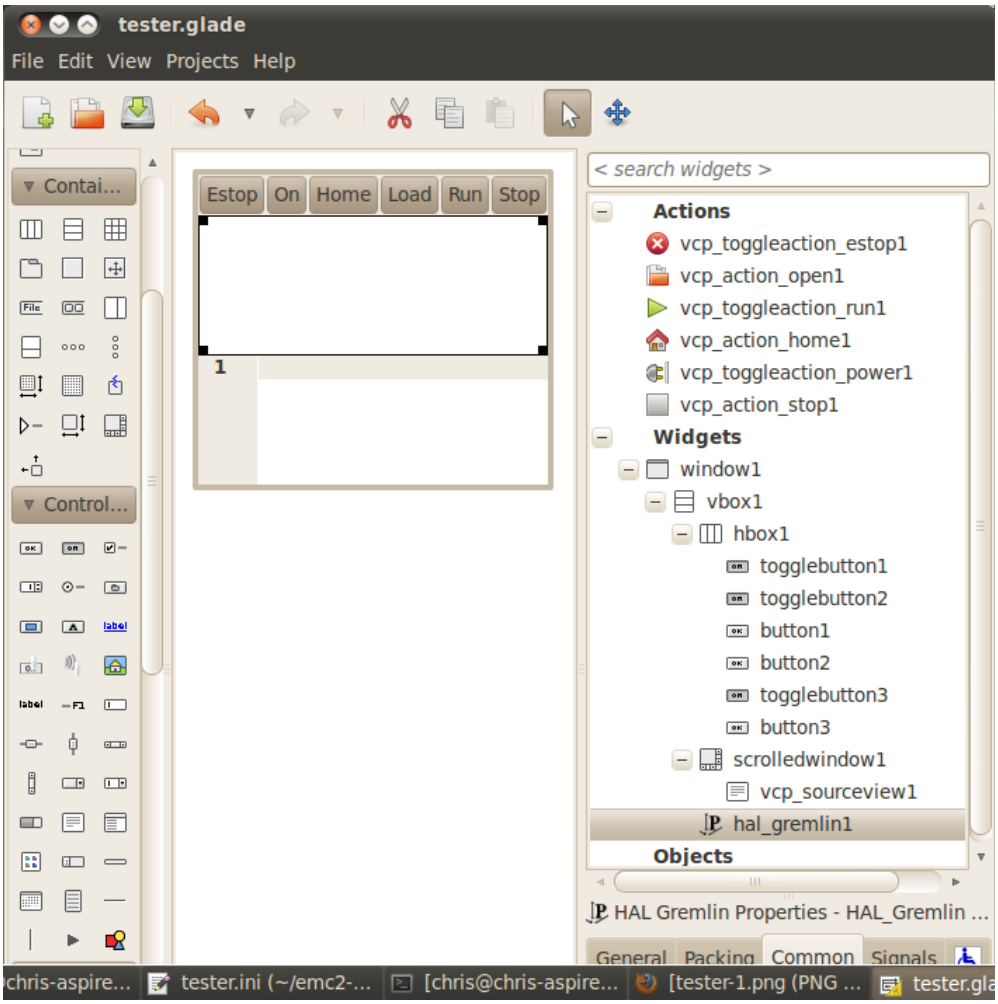
3.14.3 シンプルなクリーンシートカスタム画面の作成



シンプルで使いやすい画面を作りましょう。 これを Glade エディターでビルドします（RIP パッケージを使用している場合は、.scripts / rip-environment を使用した後にターミナルから実行します）。

注意事項：

- トップレベルウィンドウはデフォルト名である window1-と呼ばれる必要があります-Gscreen はこれに依存しています。
- 右クリックしてアクションを追加し、[トップレベルウィジェットとして追加]を選択すると、ウィンドウに視覚的なものは何も追加されませんが、右端のアクションリストに追加されます。 右上に表示されているものをすべて追加します。
- アクションを追加した後、ボタンを機能させるには、ボタンをアクションにリンクする必要があります（以下を参照）。
- グレムリンウィジェットにはデフォルトのサイズがないため、要求されたサイズを設定すると便利です（以下を参照）。
- sourceview ウィジェットはウィンドウ全体を使用しようとするため、スクロールされたウィンドウに追加するとこれがカバーされます（これは例ですでに実行されています）。
- ウィンドウが大きくなるとボタンが拡大しますが、これは醜いので、ボタンが入っているボックスを拡大しないように設定します（以下を参照）。
- 使用するボタンの種類は、使用する VCP_action によって異なります。たとえば、vcp_toggle_action には通常、トグルボタンが必要です（今のところ例に従ってください）。
- この例のボタンは、HAL ボタンではなく通常のボタンです。 HAL ピンは必要ありません。



この画面では、VCP_actions を使用して、必要なアクションを LinuxCNC に伝達しています。 これにより、ハンドラーファイルに Python コードを追加しなくても、標準の関数を使用できます。 estoptoogle ボタンを estop アクションにリンクしましょう estop トグルボタンを選択し、[全般]タブで[関連アクション]を探して、その横にあるボタンをクリックします。 次に、トグル estop アクションを選択します。 これで、ボタンをクリックすると、estop のオンとオフが切り替わります。 [全般]タブで、ボタンのラベルのテキストを変更して、ボタンの機能を説明できます。 すべてのボタンに対してこれを行います。

グレムリンウィジェットを選択し、[共通]タブをクリックして、要求された高さを 100 に設定し、その横にあるチェックボックスをクリックします。

ボタンを保持している水平ボックスをクリックします。 [パッキング]タブをクリックし、[いいえ]まで展開をクリックします。

tester.glade として保存し、sim / gscreen / gscreen_custom /フォルダーに保存します。 LinuxCNC を起動し、クリックして sim / gscreen gscreen_custom /をクリックして起動します。 すべてがうまくいけば、画面がポップアップし、ボタンが機能します。これは、tester.ini が gscreen に tester.glade と tester_handler.py を探してロードするように指示するために機能します。 tester_handler.py ファイルはそのフォルダーに含まれており、画面を表示するだけでコード化されています。 特別なウィジェットは LinuxCNC と直接通信するため、引き続き便利なことができます。 画面のニーズが利用可能な特別なウィジェットでカバーされている場合、これは画面を構築するために必要な範囲です。 もっと何かが必要な場合は、関数呼び出しを追加するだけで、動作を制限するための

多くのトリックを利用できます。 独自の Python コードをコーディングして、必要なものを正確にカスタマイズします。 しかし、それはハンドラーファイルについて学ぶことを意味します。

3.14.4 ハンドラーファイルの例

Gscreen がハンドラーファイルをチェックする特別な関数があります。 これらをハンドラーファイルに追加すると、Gscreen は、gscreen の内部の同じ名前の関数の代わりにそれらを呼び出します。

- initialize_preferences（self）：新しい設定ルーチンをインストールできます。
- initialize_keybindings（self）新しいキーバインディングルーチンをインストールできます。 ほとんどの場合、これを実行したくない場合は、個々のキーバインド呼び出しをオーバーライドする必要があります。 任意の関数を呼び出すキーバインディングをさらに追加することもできます。
- initialize_pins（self）：HAL ピンを作成/初期化します
- connect_signals（self、handlers）：完全に異なる画面を使用している場合はデフォルトの Gscreen を追加する必要があります。そうしないと、gscreen はそこにないウィジェットに信号を接続しようとします。 Gscreen のデフォルト関数は、self.gscreen.connect_signals（handlers）を使用して呼び出されます。画面に信号を追加するだけで、デフォルトの関数を最初に呼び出してから、信号を追加する場合。 信号が単純な場合（ユーザーデータが渡されない場合）、Glade エディターで Glade 信号の選択を使用することもできます。
- initialize_widgets（self）：これを使用して任意のウィジェットを設定できます。 Gscreen は通常、self.gscreen.initialize_widgets（）を呼び出します。これは、実際には多くの個別の関数を呼び出します。 これらのウィジェットのいくつかを組み込みたい場合は、それらの関数を直接呼び出すだけです。 または、self.gscreen.init_show_windows（）を追加して、ウィジェットが表示されるようにします。 次に、必要に応じて、新しいウィジェットを初期化/調整します。
- initialize_manual_toolchange（self）：手動ツールチェンジシステムの完全な刷新を可能にします。
- set_restart_line（self.line）：
- timer_interrupt（self）：割り込みルーチンの再定義を完了できます。これは、periodic（）を呼び出し、linuxcnc.status からのエラーをチェックするために使用されます。
- check_mode（self）：画面がどのモードにあるかを確認するために使用されます。list[] 0 -manual 1- mdi 2- auto 3-jog を返します。
- on_tool_change（self、widget）：これを使用して、手動のツール変更ダイアログをオーバーライドできます。これは、gscreen.toolchange が状態を変更したときに呼び出されます。
- dialog_return（self、dialog_widget、displaytype、pinname）：これを使用して、ユーザーメッセージまたは手動のツール変更ダイアログをオーバーライドします。 ダイアログが閉じられたときに呼び出されます。
- period（self）：これは（デフォルトでは 100）ミリ秒ごとに呼び出されます。 ウィジェット/ HAL ピンを更新するために使用します。 その後、定期的に Gscreen を呼び出すこともできます。

self.gscreen.update_position（）、またはパスを追加して何も更新しないようにします。 Gscreen の update_position（）は、実際には多くの個別の関数を呼び出します。 これらのウィジェットのいくつかを組み込みたい場合は、それらの関数を直接呼び出すだけです。

このファイルで呼び出す独自の関数を追加することもできます。 通常、ウィジェットにシグナルを追加して関数を呼び出します。

1.1.1.1 キーバインディング関数の追加

テスターの例は、キーボードコマンドに応答する場合にさらに役立ちます。

これを設定しようとする keybindings（）と呼ばれる関数があります。

完全にオーバーライドすることはできますが、オーバーライドしませんでした。ただし、いくつかのことを前提としています。

estop トグルボタンが button_estop の呼び出しであり、F1 キーがそれを制御していることを前提としています。

電源ボタンが button_machine_on と呼ばれ、F2 キーがそれを制御することを前提としています。

これらは、Glade エディターのボタンの名前を一致するように変更することで簡単に修正できます。

ただし、代わりに、標準の呼び出しをオーバーライドして、独自の呼び出しを追加します。

次のコマンドをハンドラーファイルに追加します。

```
# override Gscreen Functions
# keybinding calls
def on_keycall_ESTOP(self,state,SHIFT,CNTRL,ALT):
if state: # only if pressed, not released
self.widgets.togglebutton1.emit('activate')
self.gscreen.audio.set_sound(self.data.alert_sound)
self.gscreen.audio.run()
return True # stop progression of signal to other widgets
def on_keycall_POWER(self,state,SHIFT,CNTRL,ALT):
if state:
self.widgets.togglebutton2.emit('activate')
return True
def on_keycall_ABORT(self,state,SHIFT,CNTRL,ALT):
if state:
self.widgets.button3.emit('activate')
```

```
return True
```

これで、同じ名前の Gscreen の関数呼び出しをオーバーライドし、ハンドラーファイルでそれらを処理しました。 ここで、Glade エディターで使用した名前でウィジェットを参照します。

また、Estop が変更されたときに音を出すための組み込みの gscreen 関数を追加しました。

Gscreen の組み込み関数を呼び出す場合は、self.gscreen。[FUNCTION NAME]（）を使用する必要があることに注意してください。self。[FUNCTION NAME]（）を使用すると、ハンドラーファイル内の関数が呼び出されます。

F4 が押されたときにハルメーターをロードする別のキーバインディングを追加しましょう。

def initialize_widgets（self）の下のハンドラーファイルで：次のように変更します。

```
def initialize_widgets(self):
self.gscreen.init_show_windows()
self.gscreen.keylookup.add_conversion('F4','TEST','on_keycall_HALMETER')
```

次に、これらの関数を HandlerClass クラスの下に追加します。

```
def on_keycall_HALMETER(self,state,SHIFT,CNTRL,ALT):
if state:
self.gscreen.on_halmeter()
return True
```

これにより、F4 が押されたときに gscreen が on_keycall_HALMETER を呼び出すように指示するキーバインディング変換が追加されます。

次に、関数をハンドルファイルに追加して、Gscreen 組み込み関数を呼び出して halmeter を開始します。

3.14.4.1 Linuxcnc の状態ステータス

モジュール Gstat は、100 ミリ秒ごとに linuxcnc の状態をポーリングし、状態が変化するとコールバックメッセージをユーザー関数に送信します。 メッセージを登録して、特定の状態変化に対応することができます。

例として、linuxcnc が新しいファイルをロードするときにファイルがロードされたメッセージを取得するように登録します。

まず、モジュールをインポートしてインスタンス化する必要があります。

ハンドラーファイルのインポートセクションに以下を追加します。

```
from hal_glib import GStat
GSTAT = GStat()
```

def __init __（self）の下のハンドラーファイルで：次を追加します。

```
GSTAT.connect('file-loaded', self.update_filepath)
```

次に、HandlerClass に次の関数を追加します。

```
self.update_filepath(self, obj, path):
self.widgets.my_path_label.set_text(path)
```

linuxcnc が新しいファイルをロードすると、Gstat はコールバックメッセージを関数 update_filepath に送信します。 この例では、GLADE ファイルでそのパス名（my_path_label という名前のラベルがあると想定）でラベルを更新します。

3.14.4.2 ジョギングキー

画面ボタンのジョギングを行うための特別なウィジェットはないため、Python コードを使用して行う必要があります。 connect_signals 関数の下に、次のコードを追加します。

```
for i in('x','y','z'):
self.widgets[i+'neg'].connect("pressed", self['jog_'+i],0,True)
self.widgets[i+'neg'].connect("released", self['jog_'+i],0,False)
self.widgets[i+'pos'].connect("pressed", self['jog_'+i],1,True)
self.widgets[i+'pos'].connect("released", self['jog_'+i],1,False)
self.widgets.jog_speed.connect("value_changed",self.jog_speed_changed)
```

HandlerClass クラスの下に次の関数を追加します。

```
def jog_x(self,widget,direction,state):
self.gscreen.do_key_jog(_X,direction,state)
def jog_y(self,widget,direction,state):
self.gscreen.do_key_jog(_Y,direction,state)
def jog_z(self,widget,direction,state):
self.gscreen.do_key_jog(_Z,direction,state)
def jog_speed_changed(self,widget,value):
self.gscreen.set_jog_rate(absolute = value)
```

最後に、各軸の GLADE ファイルに 2 つのボタンを追加します。1 つは正の方向のジョギング用、もう 1 つは負の方向のジョギング用です。

これらのボタンにそれぞれ xneg、xpos、yneg、ypos zneg、zpos という名前を付けます。

SpeedControl ウィジェットを GLADE ファイルに追加し、jog_speed という名前を付けます

3.14.5 Gscreen の起動

Gscreen は、実際には、カスタム GladeVCP ファイルをロードして操作するための単なるインフラストラクチャです。

1. Gscreen は、開始されたオプションを読み取ります。
2. Gscreen はデバッグモードを設定し、オプションのスキン名を設定します。
3. Gscreen は、構成フォルダーにローカル XML、ハンドラー、ロケールファイルがあるかどうかを確認します。 デフォルトのもの（share / gscreen / skins /内）の代わりにそれらを使用します（2 つの別々の画面が表示される場合があります）。
4. メイン画面が読み込まれ、翻訳が設定されます。 存在する場合、2 番目の画面が読み込まれ、翻訳が設定されます。
5. オプションのオーディオは、利用可能な場合は初期化されます。
6. INI ファイルの一部を読み取って、単位と軸の数/タイプを初期化します。
7. Python の HAL へのバインドを初期化して、Gscreen 名でユーザースペースコンポーネントを構築します。
8. GladeVCP の makepin は、XML ファイルを解析して HAL ウィジェットの HAL ピンを構築し、LinuxCNC に接続されたウィジェットを登録するために呼び出されます。
9. 構成フォルダー内のローカルハンドラーファイルをチェックするか、スキンフォルダーのストックファイルを使用します。
10. ハンドラーファイルがある場合、gscreen はそれを解析し、関数呼び出しを Gscreen の名前空間に登録します。
11. Glade は、gscreen およびハンドラーファイル内の関数へのすべてのシグナル呼び出しを照合/登録します。
12. Gscreen は、INI ファイルでオプション設定ファイル名をチェックします。それ以外の場合は、.gscreen_preferences =を使用します。
13. Gscreen は、ハンドラーファイルに設定関数呼び出し（initialize_preferences（self））があるかどうかを確認します。ない場合は、ストック Gscreen のものを使用します。
14. Gscreen は、classicladder リアルタイムコンポーネントをチェックします。
15. Gscreen はシステム全体の GTK テーマをチェックします。
16. Gscreen は、INI ファイルからジョギングの増分を収集します。
17. Gscreen は、INI ファイルから角度ジョギングの増分を収集します。

18. Gscreen は、INI からデフォルトおよび最大ジョギングレートを収集します。
19. Gscreen は、INI の TRAJ セクションから任意の軸の最大速度を収集します。
20. Gscreen は、角度軸があるかどうかを確認してから、INI ファイルからデフォルト速度と最大速度を収集します。
21. Gscreen は、INI からすべてのオーバーライド設定を収集します。
22. Gscreen は、INI ファイルから旋盤構成かどうかを確認します。
23. Gscreen は、INI から tool_table、tool editor、および param ファイルの名前を検索します。
24. Gscreen は、ハンドラーファイルでキーバインディング関数（initialize_keybindings（self））をチェックするか、Gscreen ストック関数を使用します。
25. Gscreen は、ハンドラーファイルでピン関数（initialize_pins（self））をチェックするか、Gscreen ストック関数を使用します。
26. Gscreen は、ハンドラーファイルで manual_toolchange 関数（initialize_manual_toolchange（self））をチェックするか、Gscreen ストック関数を使用します。
27. Gscreen は、ハンドラーファイルで connect_signals 関数（initialize_connect_signals（self））をチェックするか、Gscreen ストック関数を使用します。
28. Gscreen は、ウィジェット関数のハンドラーファイル（initialize_widgets（self））をチェックするか、Gscreen ストックファイルを使用します。
29. INI ファイルで指定された画面設定メッセージ。
30. Gscreen は、Gscreen HAL コンポーネントがピンの作成を終了し、準備ができていることを HAL に通知します。 画面にターミナルウィジェットがある場合は、すべての Gscreen ピンがそれに印刷されます。
31. Gscreen は、INI ファイルに基づいて表示サイクル時間を設定します。
32. Gscreen は、ハンドラーファイルで timer_interupt（self）関数呼び出しをチェックします。それ以外の場合は、Gscreen のデフォルトの関数呼び出しを使用します。

3.14.6 INI 設定

[DISPLAY]見出しの下：

```
DISPLAY = gscreen -c tester
options:
-d debugging on
-v verbose debugging on
```

-c スイッチを使用すると、スキンを選択できます。 Gscreen は、Glade ファイルとハンドラーファイルがこれと同じ名前を使用していることを前提としています。 オプションの 2 番目の画面は 2 と同じ名前になります

（例：tester2.glade）2 番目のハンドラーファイルは許可されていません。 存在する場合にのみロードされます。 Gscreen は、最初に起動された LinuxCNC 構成ファイルでファイルを検索し、次に systen スキンフォルダーで検索します。

3.14.7 ユーザーダイアログメッセージ

この機能は、ポップアップダイアログメッセージを画面に表示するために使用されます。

これらは INI ファイルで定義され、HAL ピンによって制御されます。

太字は一般的にタイトルです。

テキストはその下にあり、通常は長くなります。

クリックしない限り、詳細は非表示になります。

pinname は、HAL ピンのベース名です。

type は、yes / no、ok、または status メッセージのいずれであるかを指定します。

ステータスメッセージは、ステータスバーと通知ダイアログに表示されます。

ユーザーの介入は必要ありません。

ok メッセージでは、ユーザーは[ok]をクリックしてダイアログを閉じる必要があります。

ok メッセージには、ダイアログを起動するための 1 つの HAL ピンと、応答を待機していることを示すための 1 つの HAL ピンがあります。

はい/いいえメッセージでは、ユーザーがダイアログを閉じるために[はい]または[いいえ]ボタンを選択する必要があります。

はい/いいえメッセージには 3 つの hal ピンがあります。1 つはダイアログを表示するため、1 つは待機用、もう 1 つは応答用です。

これがサンプルの INI コードです。 [DISPLAY]の見出しの下にあります。

```
# This just shows in the status bar and desktop notify popup.
MESSAGE_BOLDTEXT = NONE
MESSAGE_TEXT = This is a statusbar test
MESSAGE_DETAILS = STATUS DETAILS
MESSAGE_TYPE = status
MESSAGE_PINNAME = statustest
# This will pop up a dialog that asks a yes no question
MESSAGE_BOLDTEXT = NONE
MESSAGE_TEXT = This is a yes no dialog test
MESSAGE_DETAILS = Y/N DETAILS
MESSAGE_TYPE = yesnodialog
```

```
MESSAGE_PINNAME = yndialogtest
# This pops up a dialog that requires an ok response and it shows in the status bar and
# the destop notify popup.
MESSAGE_BOLDTEXT = This is the short text
MESSAGE_TEXT = This is the longer text of the both type test. It can be longer then the -
status bar text
MESSAGE_DETAILS = BOTH DETAILS
MESSAGE_TYPE = okdialog status
MESSAGE_PINNAME = bothtest
```

1.1.1.1 変更のためにストックハンドラー/グレードファイルをコピーする

ストック画面を使用したいが、ハンドラーファイルを変更する場合は、ストックファイルを構成ファイルフォルダーにコピーする必要があります。 Gscreen はこれを認識し、コピーされたファイルを使用します。

しかし、元のファイルはどこにありますか？ RIP linuxcnc を使用している場合、サンプルスキンは/ share / gscreen / skins / SCREENNAME にあります。インストールされているバージョンの linuxcnc は、使用されているディストリビューションに応じて、わずかに異なる場所にあります。

場所を見つける簡単な方法は、ターミナルを開いて、使用したい sim 画面を開始することです。

ターミナルでは、ファイルの場所が印刷されます。

INI の gscreen ロードラインに-d スイッチを追加すると役立つ場合があります。

サンプルは次のとおりです。

```
chris@chris-ThinkPad-T500 ~/emc-dev/src $ linuxcnc
LINUXCNC - 2.7.14
Machine configuration directory is
'/home/chris/emc-dev/configs/sim/gscreen/gscreen_custom'
Machine configuration file is 'industrial_lathe.ini'
Starting LinuxCNC...
Found file(lib): /home/chris/emc-dev/lib/hallib/core_sim.hal
Note: Using POSIX non-realtime
Found file(lib): /home/chris/emc-dev/lib/hallib/sim_spindle_encoder.hal
Found file(lib): /home/chris/emc-dev/lib/hallib/axis_manualtoolchange.hal
Found file(lib): /home/chris/emc-dev/lib/hallib/simulated_home.hal
**** GSCREEN WARNING: no audio alerts available - Is python-gst0.10 libray installed?
**** GSCREEN INFO ini: /home/chris/emc-dev/configs/sim/gscreen/gscreen_custom/ -
industrial_lathe.ini
**** GSCREEN INFO: Skin name = industrial
```

```
**** GSCREEN INFO: Using SKIN glade file from /home/chris/emc-dev/share/gscreen/skins/
-
industrial/industrial.glade ****
**** GSCREEN INFO: No Screen 2 glade file present
**** GSCREEN INFO: handler file path:
['/home/chris/emc-dev/share/gscreen/skins/industrial/  -
industrial_handler.py']
```

この線：

```
**** GSCREEN INFO: handler file path:
['/home/chris/emc-dev/share/gscreen/skins/industrial/  -
industrial_handler.py']
```

ストックファイルが存在する場所を示します。 このファイルを config フォルダーにコピーします。

これは、Glade ファイルでも同じように機能します。

3.15 TkLinuxCNC GUI

3.15.1 序章

TkLinuxCNC は、LinuxCNC の最初のグラフィカルフロントエンドの 1 つです。 これは Tcl で記述されており、表示に Tk ツールキットを使用します。 Tcl で記述されているため、非常に移植性が高くなります（多数のプラットフォームで実行されます）。 図のように、別のバックプロットウィンドウを表示できます。

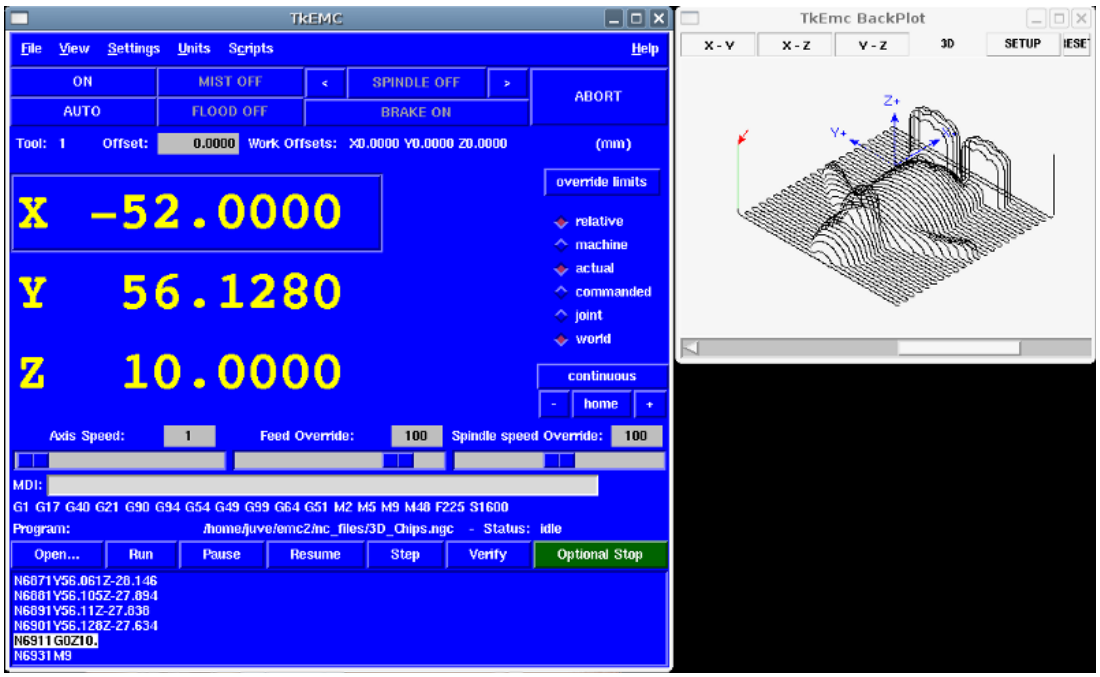


図 4-24

3.15.2 入門

LinuxCNC のフロントエンドとして TkLinuxCNC を選択するには、.ini ファイルを編集します。 [DISPLAY]セクションで、DISPLAY 行を次のように変更します

```
DISPLAY = tklinuxcnc
```

次に、LinuxCNC を起動し、その ini ファイルを選択します。 サンプル構成 sim / tklinuxcnc / tklinuxcnc.ini は、フロントエンドとして TkLinuxCNC を使用するように既に構成されています。

1.1.1.1 TkLinuxCNC との典型的なセッション

1. LinuxCNC を起動し、構成ファイルを選択します。
2. 非常停止状態をクリアし、マシンの電源をオンにします（F1 を押してから F2 を押します）。
3. 各軸をホームします。
4. ミリングするファイルをロードします。
5. ミーリングするストックをテーブルに置きます。
6. ジョギングして再度原点復帰するか、軸名を右クリックしてオフセット値を入力することにより、各軸に適切なオフセットを設定します。1)
   注 1)これらのアクションの一部では、LinuxCNC が現在実行されているモードを変更する必要がある場合があります。
7. プログラムを実行します。
8. 同じファイルを再度ミリングするには、手順 6 に戻ります。別のファイルをミリングするには、手順 4 に戻ります。完了したら、LinuxCNC を終了します。

3.15.3 TkLinuxCNC ウィンドウの要素

TkLinuxCNC ウィンドウには、次の要素が含まれています。

- さまざまなアクションを実行できるメニューバー
- 現在の作業モード、スピンドルの開始/停止、およびその他の関連する I / O を変更できる一連のボタン
- さまざまなオフセット関連ディスプレイのステータスバー
- 座標表示エリア
- ジョギング速度、フィードオーバーライド、およびスピンドル速度オーバーライドを制御する一連のスライダー。これらの設定を増減できます。
- 手動データ入力テキストボックス MDI

- アクティブなGコード、Mコード、FワードおよびSワードを含むステータスバーの表示
- 通訳関連ボタン
- ロードされたファイルのGコードソースを示すテキスト表示領域

1.1.1.1 メインボタン

左から右へ、ボタンは次のとおりです。

- マシンの有効化：ESTOP> ESTOP RESET> ON
- ミストクーラントを切り替えます
- スピンドル速度を下げる
- 主軸方向を設定します。主軸オフ>主軸前進。 スピンドルリバース
- スピンドル速度を上げる
- アボート

次に2行目：

- 動作モード：MANUAL> MDI> AUTO
- フラッドクーラントを切り替えます
- スピンドルブレーキ制御を切り替えます

3.15.3.1 オフセット表示ステータスバー

オフセット表示ステータスバーには、現在選択されている工具（Txx M6で選択）、工具長オフセット（アクティブな場合）、および作業オフセット（座標を右クリックして設定）が表示されます。

3.15.3.2 座標表示エリア

ディスプレイの主要部分には、ツールの現在の位置が表示されます。 位置の読み取りの色は、軸の状態によって異なります。 軸がホームにない場合、軸は黄色の文字で表示されます。 家に帰ると、緑色の文字で表示されます。 現在の軸にエラーがある場合、TkLinuxCNCは赤い文字を使用してそれを示します。（たとえば、ハードウェアリミットスイッチが作動した場合）。

これらの番号を正しく解釈するには、右側のラジオボックスを参照してください。 位置が機械の場合、表示される数値は機械座標系にあります。 相対の場合、表示される数値はオフセット座標系にあります。 さらに下の選択肢は、実際のものまたは命令することができます。 実際はエンコーダからのフィードバック（サーボマシンを使用している場合）を指し、コマンドはモーターに送信される位置コマンドを指します。 これらの値は、いくつかの理由で異なる可能性があります。次のエラー、不感帯、エンコーダ分解能、またはステップサイズ。 たとえば、ミルでX 0.0033への移動をコマンドしたが、ステッピングモーターの1ステップが0.00125の場合、コマンド位置は0.0033になりますが、実際の位置は0.0025（2ステップ）または0.00375（3ステップ）になります。

別のラジオボタンのセットを使用すると、ジョイントビューとワールドビューのどちらかを選択できます。 これらは通常のタイプのマシン（例：トリビアルキネマティクス）ではほとんど意味がありませんが、ロボットや

スチュワートプラットフォームなどのトリビアルキネマティクスを備えたマシンでは役立ちます。（キネマティクスの詳細については、インテグレーターマニュアルを参照してください）。

バックプロットマシンが移動すると、バックプロットと呼ばれる軌跡が残ります。 View！Backplot を選択すると、バックプロットウィンドウを開始できます。

3.15.3.3 自動運転

制御用のボタン TkLinuxCNC の下部にあるボタンは、プログラムの実行を制御するために使用されます。プログラムをロードするために開く、エラーがないか確認するために確認する、実際の切断を開始するために実行する、実行中に停止するために一時停止する、再開する すでに一時停止しているプログラムを再開し、ステップでプログラムを 1 行進め、オプションの停止でオプションの停止スイッチを切り替えます（ボタンが緑色の場合、M1 が検出されるとプログラムの実行が停止します）。

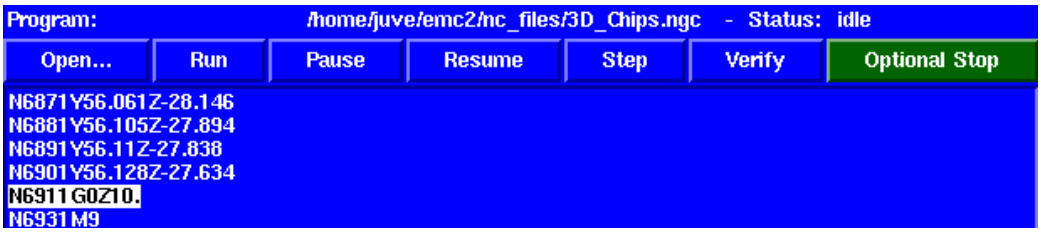


図 4-25

テキストプログラム表示領域プログラムの実行中は、現在実行中の行が白で強調表示されます。 テキストディスプレイは自動的にスクロールして現在の行を表示します。

3.15.3.4 手動制御

暗黙のキー TkLinuxCNC を使用すると、マシンを手動で移動できます。 このアクションはジョギングとして知られています。 まず、移動する軸をクリックして選択します。 次に、目的の動作方向に応じて、[+]または[-]ボタンをクリックして押したままにします。 最初の 4 つの軸は、キーボードの矢印キー（X および Y）、PAGEUP および PAGEDOWN キー（Z）、および[および]キー（A / 4 番目）によっても移動できます。

[連続]を選択すると、ボタンまたはキーが押されている間、モーションが続行されます。 別の値を選択した場合、ボタンがクリックされるか、キーが押されるたびに、マシンは表示された距離を正確に移動します。 使用可能な値は、1.0000、0.1000、0.0100、0.0010、0.0001 です。

ホームキーまたはホームキーを押すと、選択した軸がホームになります。 構成によっては、軸の値を絶対位置 0.0 に設定するだけの場合もあれば、ホームスイッチを使用してマシンを特定のホームロケーションに移動させる場合もあります。 詳細については、ホーミングの章を参照してください。

オーバーライド制限を押すと、マシンは一時的に.ini ファイルで定義された制限の外でジョギングすることが許可されます。 （注：[制限の上書き]がアクティブな場合、ボタンは赤い色で表示されます）。

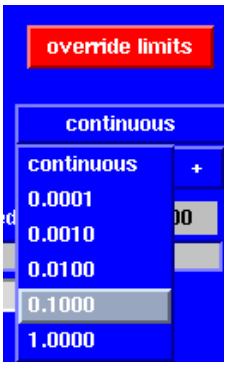


図 4-26

スピンドルグループ最初の行のボタンは、スピンドルが回転する方向を選択します：反時計回り、停止、時計回り。 その横にあるボタンを使用すると、ユーザーは回転速度を増減できます。 2 列目のボタンを使用すると、スピンドルブレーキを作動または解放できます。 マシンの構成によっては、このグループのすべての項目が影響を与えるとは限りません。

クーラントグループ 2 つのボタンを使用すると、ミストクーラントとフラッドクーラントのオンとオフを切り替えることができます。 マシンの構成によっては、このグループのすべてのアイテムが表示されるとは限りません。

3.15.3.5 コード入力

手動データ入力（MDI とも呼ばれます）を使用すると、G コードプログラムを一度に 1 行ずつ手動で入力できます。 マシンの電源がオンになっておらず、MDI モードに設定されていない場合、コード入力コントロールは使用できません。

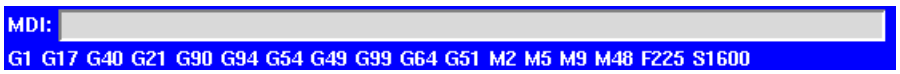


図 4-27

MDI：これにより、実行する g コードコマンドを入力できます。 Enter キーを押してコマンドを実行します。 アクティブな G コードこれは、インタプリタでアクティブなモーダルコードを示します。 たとえば、G54 は、入力されたすべての座標に G54 オフセットが適用されることを示します。

3.15.3.6 ジョグ速度

このスライダーを動かすことで、ジョグの速度を変えることができます。 上記の数値は、軸の単位/秒を示しています。 番号の付いたテキストボックスをクリックできます。 クリックするとポップアップウィンドウが表示され、番号を入力できます。

3.15.3.7 フィードのオーバーライド

このスライダーを動かすことにより、プログラムされた送り速度を変更することができます。 たとえば、プログラムが F60 を要求し、スライダーが 120%に設定されている場合、結果の送り速度は 72 になります。番号の付いたテキストボックスをクリックできます。 クリックするとポップアップウィンドウが表示され、番号を入力できます。

3.15.3.8 スピンドル速度オーバーライド

スピンドル速度オーバーライドスライダーは、フィードオーバーライドスライダーとまったく同じように機能しますが、スピンドル速度を制御します。 プログラムが S500（スピンドル速度 500 RPM）を要求し、スライダーが 80%に設定されている場合、結果のスピンドル速度は 400RPM になります。 このスライダーには、ini ファイルで定義されている最小値と最大値があります。 それらが欠落している場合、スライダーは 100%でスタックします。 番号の付いたテキストボックスをクリックできます。 クリックするとポップアップウィンドウが表示され、番号を入力できます。

3.15.4 キーボードコントロール

TkLinuxCNC のほとんどすべてのアクションは、キーボードを使用して実行できます。 MDI モードでは、ショートカットの多くは使用できません。

次の表に、最も頻繁に使用されるキーボードショートカットを示します。

表 4.2 最も一般的なキーボードショートカット

キーストローク	実行されたアクション
F1	緊急停止の切り替え
F2	マシンのオン/オフを切り替える
`、1 .. 9、0	フィードオーバーライドを 0%から 100%に設定します
X、`	最初の軸をアクティブにする
Y、1	2 番目の軸をアクティブにします
Z、2	3 番目の軸をアクティブにします
A、3	4 番目の軸をアクティブにします
Home	アクティブな軸をホームに送信
Left, Right	ジョグ第 1 軸
Up, Down	ジョグ第 2 軸
Pg Up, Pg Dn	ジョグ第 3 軸
[、]	ジョグ第 4 軸
ESC	実行を停止します

3.16 PlasmaC ユーザーガイド

3.16.1 ライセンス

PlasmaC とそれに関連するすべてのソフトウェアは、GPLv2 でリリースされています。

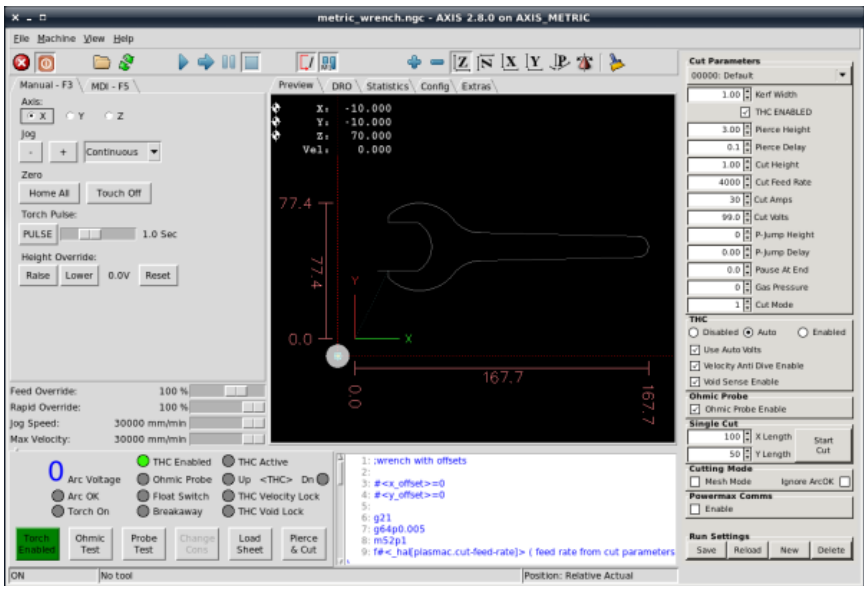
3.16.2 序章

PlasmaC は、LinuxCNCv2.8 以降に搭載されるプラズマ切断構成です。 これには、HAL コンポーネントに加えて、Axis と Gmoccapy の両方の GUI 構成が含まれています。 AxisGUI を縦向きモードで表示するオプションもあります。Axis<machine_name> .ini ファイルを参照してください。

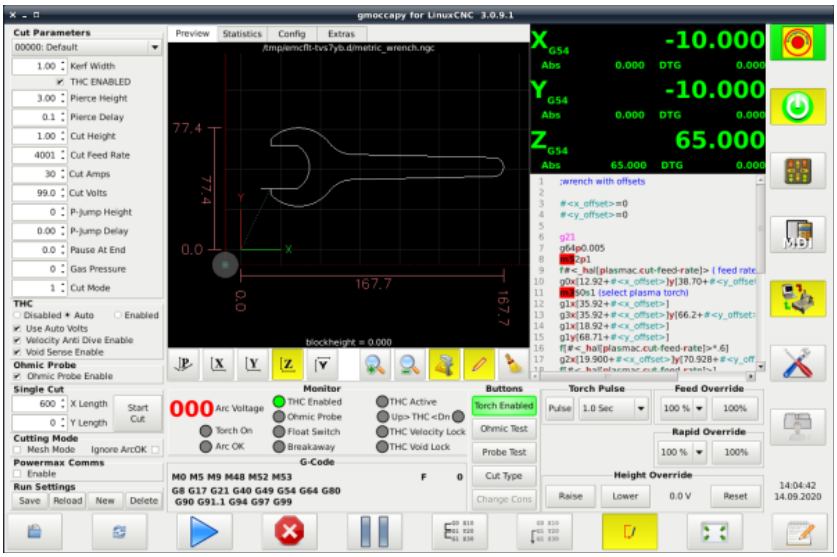
Axis と Gmoccapy を切り替えるには新しい構成を作成する必要があるため、Linux-CNC のインストールプロセスを開始する前に、PlasmaC コンポーネントがロードされた目的の LinuxCNC GUI（Axis または Gmoccapy）を選択する必要があります。

AxisGUI と GmoccapyGUI の両方の上にロードされた PlasmaC のスクリーンショットの例を以下に示します。

AXIS:



GMOCCAPY：



PlasmaC コンポーネントは、プラズマ構成の要件を満たすのに十分なハードウェア I / O ピンがあれば、LinuxCNC でサポートされているすべてのハードウェアで実行する必要があります。

ユーザーが CNC プラズマ切断に不慣れな場合は、CNC プラズマ切断の一般的な紹介である Plasma CNCPrimer ドキュメントを読むことをお勧めします。

Note

特に記載のない限り、このガイドでは、ユーザーが最新バージョンの PlasmaC を使用していることを前提としています。 PlasmaC の更新通知は、https：//forum.linuxcnc.org/plasmac/37233-plasmac-updates に掲載されています。構成パネルで現在の PlasmaC のバージョンを確認するには、PlasmaC のバージョンを参照してください。 バージョン番号が存在しない場合、PlasmaC のユーザーのバージョンは v0.121 より前です。 PlasmaC の更新については、PlasmaC の更新を参照してください。

3.16.3 LinuxCNC のインストール

LinuxCNC（デフォルトで PlasmaC を含む）をインストールするための推奨される方法は、以下に説明する ISO イメージを使用することです。

Note

LinuxCNC をさまざまな Linux ディストリビューションにインストールして実行することは可能ですが、それはこのユーザーガイドの範囲を超えています。 ユーザーが推奨以外の Linux ディストリビューションをインストールする場合は、最初に優先 Linux ディストリビューションをインストールしてから、必要な依存関係とともに LinuxCNCv2.8 以降をインストールする必要があります。

1.1.1.1 ユーザーが Linux をインストールしていない場合

インストール手順は、http：//linuxcnc.org/docs/devel/html/getting-started/getting-linuxcnc.html で入手できます。

これらの手順に従うと、LinuxCNC の現在の安定したブランチ（v2.8）を備えたマシンが生成されます。

3.16.3.1 ユーザーが LinuxCNCv2.7 を搭載した Linux を使用している場合

アップグレード手順は、http：//linuxcnc.org/docs/devel/html/getting-started/updating-linuxcnc.html で入手できます。

これらの手順に従うと、LinuxCNC の現在の安定したブランチ（v2.8）を備えたマシンが生成されます。

3.16.3.2 作業ベースマシン構成の作成

Note

「基本マシン構成」とは、「I / O 要件」セクションに示されているプラズマ接続のない完全に機能するシステムを意味します。 すべての軸が機能し、最高のパフォーマンスが得られるように調整されている必要があり、X、Y、および Z 軸のすべてのホームスイッチとリミットスイッチ（取り付けられている場合）が正しく動作している必要があります。

重要

現時点では、I / O 要件のセクションに示されているプラズマ接続を追加しないでください。これらの I / O は、後でコンフィギュレーターを使用した PlasmaC 構成中に追加されます。

いくつかの推奨設定：

- Z MINIMUM_LIMIT は、float_switch_travel と超過移動許容値を考慮して、スラットの上部のすぐ下にある必要があります。 たとえば、ユーザーのフロートスイッチがアクティブになるまでに 4mm（0.157 "）かかる場合は、Z の最小値を 5mm（0.197"）に加えて、最低スラットの下のオーバーランの許容値（以下の式を使用して計算）を設定します。
- Z MAXIMUM_LIMIT は、ユーザーが Z 軸を移動させたい最高値である必要があります（Z HOME_OFFSET より低くすることはできません）。
- Z HOME は、上限値より約 5mm（0.196 "）低く設定する必要があります。
- フローティングヘッド-フローティングヘッドを使用し、プロービング中にオーバーランを許容するのに十分な動きがあることをお勧めします。 オーバーランは、次の式を使用して計算できます。

o = 0.5 $\times$ a $\times$ (v $\div$ a)^2

ここで、to =オーバーラン、a =単位/秒 2 の加速度、v =単位/秒の速度。

メートル法の例：Z 軸の MAX_ACCELERATION が 600mm / s2、MAX_VELOCITY が 60mm / s の場合、オーバーランは 3mm になります。

帝国の例：Z 軸の MAX_ACCELERATION が 24in / s2、MAX_VELOCITY が 2.4in / s の場合、オーバーランは 0.12in になります。

プローブの主な方法としてオームプローブを使用するマシンでは、表面が汚れているためにオームプローブが故障した場合に、Z モーションを停止するバックアップ手段として、フローティングヘッドにスイッチを取り付けることを強くお勧めします。

ユーザーは、ベースマシンを手動で作成するか、既存の構成ヘルパーの 1 つを使用するかを選択できます。

Note

完全にテストおよび調整されるまで、ベースマシンの構成を単純にしておくことを強くお勧めします。

Stepconf または Pncconf を使用している場合は、VCP パネル、スピンドル、手動ツール変更、および従来のラダーオプションの選択を解除します。

前述のオプションは、必要に応じて後で手動で追加できます。

重要

プラズマ固有の I / O 要件を PNCCONF または STEPCONF ウィザードに追加しないでください。

Mesa Electronics ボードを使用している場合は、pncconf ウィザードを使用します（ターミナルウィンドウに次のコマンドを入力します）。

```
pncconf
```

パラレルポートを使用している場合は、stepconf ウィザードを使用します（ターミナルウィンドウに次のコマンドを入力します）。

```
stepconf
```

Pico Systems ボードを使用している場合：

この LinuxCNC フォーラムスレッドが役立つ場合があります。

ユーザーがすでにデュアルモーターガントリー構成を持っていて、構成を手動で編集する必要がある場合：

この LinuxCNC フォーラムスレッドが役立つ場合があります。

重要

続行する前に、ユーザーはマシンをホームに戻し、各軸をゼロにし、すべての軸をクラッシュすることなくソフトリミットにジョグし、エラーなしでテスト G コードプログラムを実行できる必要があります。

この基準が満たされた場合にのみ、ユーザーはコンフィギュレーターを実行して、動作中のマシンの「上」で PlasmaC の構成を続行する必要があります。

注意

ベースマシンが構成され、動作するまで続行しないでください

3.16.4 プラズマ固有の I / O に関する考慮事項

プラズマ構成を開始する前に、ユーザーが使用可能な動作モードと、プラズマ動作を成功させるために必要な I / O をしっかりと理解していることが重要です。

1.1.1.1 モード

PlasmaC では、次の 3 つの動作モードのいずれかを選択する必要があります。

モード	説明
0	外部アーク電圧入力を使用して、アーク電圧（トーチ高さ制御用）とアーク OK の両方を計算します。
1	外部アーク電圧入力を使用してアーク電圧を計算します（トーチ高さ制御用）。 ArcOK に外部 ArcOK 入力を使用します。
2	ArcOK に外部 ArcOK 入力を使用します。 トーチの高さ制御には外部の上下信号を使用します。

重要

プラズマ電源にアーク OK（転送）出力がある場合は、モード 0 で提供されるソフト（計算）アーク OK ではなく、アーク OK に使用することをお勧めします。

3.16.4.1 利用可能な I / O

Note

このセクションでは、PlasmaC コンポーネントに必要なハードウェア I / O についてのみ触れます。 リミットスイッチ、ホームスイッチなどの基本マシン要件はこれらに追加されており、ユーザーがコンフィギュレーターを実行する前に、すでに構成され、機能している必要があります。

名前	モデル	説明
アーク電圧	0,1	アナログ入力; オプション。 エンコーダ装備のブレークアウトボードのベロシティ出力に接続されています。 この信号は、アーク電圧を読み取り、切断中にワークピースからのトーチ距離を維持するために必要な補正を決定するために使用されます。
アーク OK	1,2	デジタル入力; オプション。 プラズマ電源の ArcOK 出力からブレークアウトボードの入力に接続されています。 この信号は、カッティングアークが確立され、マシンが移動しても問題がないかどうかを判断するために使用されます（アーク転送と呼ばれることもあります）。
フロートスイッチ	0,1,2	デジタル入力; オプション。以下の表の情報を参照してください。 ブレイクアウトボード入力からフローティングヘッドのスイッチに接続されています。 この信号は、トーチでワークピースを機械的にプローブ

		し、ワークピースの上部にZゼロを設定するために使用されます。 使用され、オームプローブが構成されていない場合、これが主要なプロービング方法です。 使用され、オームプローブが構成されている場合、これはフォールバックプロービング方法です。
オーミックプローブ	0,1,2	デジタル入力; オプション。以下の表の情報を参照してください。 オーミックプローブの出力からブレークアウトボードの入力に接続されています。 この信号は、ワークピースとトーチ消耗品を使用して回路を完成させ、ワークピースの上部にZゼロを設定することにより、電子的にプローブするために使用されます。 使用する場合、これが主要なプロービング方法です。 オーミックプローブがワークピースの位置を特定できず、フロートスイッチが存在しない場合、トーチが破損するか、最小Z制限に達するまで、プローブが続行されます。
オーミックプローブの有効化	0,1,2	デジタル出力; オプション。以下の表の情報を参照してください。 ブレークアウトボードの出力から入力に接続して、オーミックプローブの電力を制御します。
ブレイクアウェイスイッチ	0,1,2	デジタル入力; オプション。以下の表の情報を参照してください。 ブレイクアウトボード入力からトーチブレイクアウェイ検出スイッチに接続されています。 この信号は、トーチがクレードルから離れたかどうかを検知します。
トーチオン	0,1,2	デジタル出力; 必要。 ブレイクアウトボード出力からプラズマ電源のトーチオン入力に接続されています。 この信号は、プラズマ電源を制御し、アークを開始するために使用されます。
上に移動	2	デジタル入力; オプション。 外部THCコントロールのアップ出力からブレークアウトボード入力に接続されています。 この信号は、Z軸を上向きに制御し、切断中にワークピースからのトーチ距離を維持するために必要な修正を行うために使用されます。
下に移動	2	デジタル入力; オプション。 外部THCコントロールのダウン出力からブレークアウトボード入力に接続されています。 この信号は、Z軸を下向きに制御し、切断中にワークピースからのトーチ距離を維持するために必要な修正を行うために使用されます。
スクライブアーミング	0,1,2	デジタル出力; オプション。 ブレイクアウトボード出力からスクライブアーミング回路に接続されています。 この信号は、スクライブをワークピースの所定の位置に配置するために使用されます。
スクライブオン	0,1,2	デジタル出力; オプション。

		ブレイクアウトボード出力からスクライブオン回路に接続されています。 この信号は、スクライビングデバイスをオンにするために使用されます。

フロートスイッチまたはオーミックプローブのいずれか 1 つだけが必要です。 両方が使用されている場合、オーミックプローブが検出されない場合、フロートスイッチはフォールバックになります。

Ohmic Probe を使用する場合は、PlasmaCGUI で OhmicProbeEnable をチェックする必要があります。

フロートスイッチはプロービングしていないときはブレイクアウェイと同じように扱われるため、ブレイクアウェイスイッチは必須ではありません。 それらが 2 つの別個のスイッチであり、ブレークアウトボードに十分な入力がない場合は、それらを組み合わせてフロートスイッチとして接続することができます。

Note

PlasmaC 構成が機能するための最小 I / O 要件は、アーク電圧入力またはアーク OK 入力、フロートスイッチ入力、およびトーチオン出力です。 繰り返しになりますが、この場合、PlasmaC は、プロービングしていないときはフロートスイッチをブレークアウェイスイッチとして扱います。

重要

上記のピンは、後で PLASMACCONFIGURATOR プロセス中に入力されます。 配線中のブレイクアウトボード上の対応する入力と出力に注意してください。 これらのピンは、ベースマシンの.HAL ファイルを作成するために使用しないでください。

3.16.5 Configurator を介したベースマシンへの PlasmaC のインストール

この時点で、ユーザーは、先に進む前に、プラズマ固有の I / O に接続せずに、完全にテストされ、動作するベースマシン構成を用意する必要があります。 PlasmaC を追加した後、pncconf および stepconf を使用して構成を編集することはできなくなったため、基本マシン構成を編集することははるかに困難です。

警告

ベースマシンが完成するまで続行しないでください。

Note

ユーザーがアーク電圧測定に MesaElectronics THCAD カードを使用している場合は、先に進む前に MesaTHCAD を参照してください。

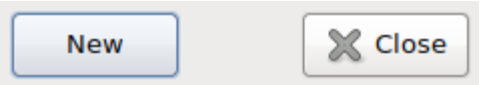
PlasmaC 構成のベースマシンへのインストールは、Configurator から実行されます。

ターミナルウィンドウに次のコマンドを入力します。

```
python /usr/share/doc/linuxcnc/examples/sample-configs/by_machine/plasmac/
configurator.py
```

1.1.1.1 構成、設定

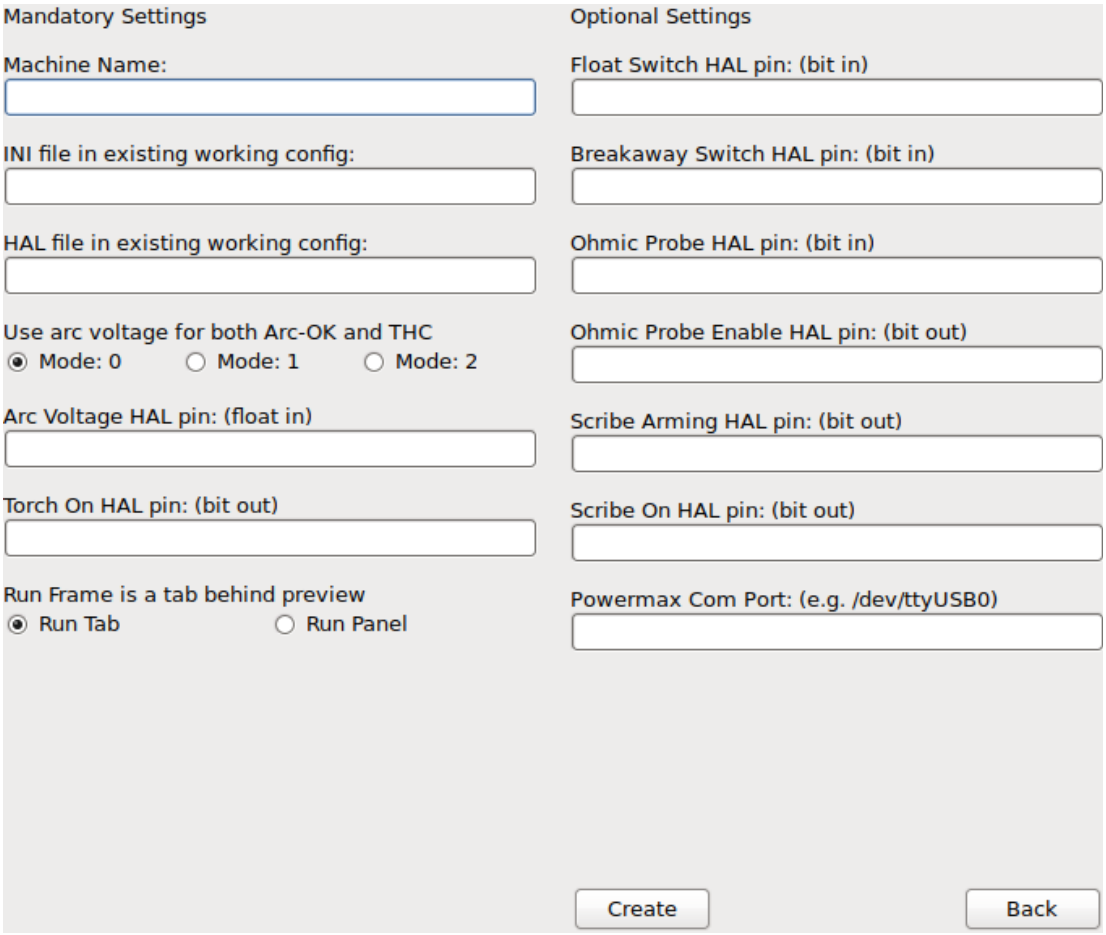
選択ウィンドウが表示されます。



選択ウィンドウから[新規]を選択すると、情報ダイアログが表示され、[続行]を選択すると、[新しい構成]ウィンドウが表示されます。

Note

選択したモードに応じて、異なるフィールドが表示されます。



次の表の HAL 接続の例は、パラレルポート構成と Mesa7i96 構成の両方の例を示しています。 ブレークアウトボードの構成に合わせて、必要に応じてピン名を変更します。

フィールド	説明	例
マシン名	マシンの新しい名前。	Plasma_table.ini。

	これにより、〜/ linuxcnc / <machine_name>ディレクトリと<machine_name> .ini ファイルが作成されます。「<machine_name>」は、このフィールドに入力された名前を示します	
INI ファイル	これは、動作し、テストされた基本マシン構成を作成した結果として作成された.ini ファイルです。	base.ini
HAL ファイル	これは、動作してテストされた基本マシン構成を作成した結果として作成された.hal ファイルです。	base.hal
モード	次の基準に基づいて、必要なモードを選択します。 1- 外部アーク電圧入力を使用してアーク電圧（トーチ高さ制御用）とアーク OK の両方を計算する場合。 2- アーク電圧を計算するために外部アーク電圧入力（トーチ高さ制御用）とアーク OK 用の外部アーク OK 入力の両方を使用する場合。 3- ArcOK の外部 ArcOK 入力と、トーチ高さ制御の外部上下信号の両方を使用する場合。	1
アーク電圧	モード 0 および 1 にのみ必要です。 アーク電圧信号が接続されている入力 HAL ピンを指します。	パラレルポートの例： エンコーダー.0。速度 Mesa 7i96 の例： hm2_7i96.0.encoder.00.velocity
トーチオン	すべてのモードに必要です。 トーチオン信号が接続されている出力 HAL ピンを指します。	パラレルポートの例： parport.0.pin-16-out Mesa 7i96 の例： hm2_7i96.0.ssr.00.out-00
アーク OK	モード 1 および 2 にのみ必要です。 ArcOK 信号が接続されている入力 HAL ピンを参照します。	パラレルポートの例： parport.0.pin-10-in-not Mesa 7i96 の例： hm2_7i96.0.gpio.008.in
オーミックプローブ	オーミックプローブを使用する場合に必要です。 オーミックプローブ信号が接続されている入力 HAL ピンを指します。	パラレルポートの例： parport.0.pin-11-in Mesa 7i96 の例： hm2_7i96.0.gpio.007.in
オーミックプローブイネーブル	オーミックプローブを使用する場合に必要です。 オーミックプローブイネーブル信号が接続されている出力 HAL ピンを指します。	パラレルポートの例： parport.1.pin-01-out Mesa 7i96 の例： hm2_7i96.0.ssr.00.out-01
フロートスイッチ	フロートスイッチを使用する場合に必要です。 フロートスイッチ信号が接続されている入力 HAL ピンを参照します。	パラレルポートの例： parport.0.pin-12-in Mesa 7i96 の例：

		hm2_7i96.0.gpio.006.in
ブレイクアウェイスイッチ	ブレークアウェイスイッチを使用する場合に必要です。 ブレークアウェイスイッチ信号が接続されている入力 HAL ピンを指します。	パラレルポートの例： parport.0.pin-13-in Mesa 7i96 の例： hm2_7i96.0.gpio.005.in
上に移動	モード 2 にのみ必要です。 ムーブアップ信号が接続されている入力 HAL ピンを指します。	パラレルポートの例： parport.1.pin-10-in Mesa 7i96 の例： hm2_7i96.0.gpio.004.in
下に移動	モード 2 にのみ必要です。 下移動信号が接続されている入力 HAL ピンを参照します。	パラレルポートの例： parport.1.pin-11-in Mesa 7i96 の例： hm2_7i96.0.gpio.003.in
パネルを実行	[実行]タブ-このオプションを選択すると、PlasmaC 実行フレームがプレビュータブの後ろのタブに配置されます。 パネルの実行-このオプションを選択すると、PlasmaC 実行フレームが GUI の横にあるパネルに配置されます。 それぞれの例については、実行パネルを参照してください。	タブを実行
スクライブアーミング	スクライブを使用する場合は必須です。 スクライブアーミングメカニズムが接続されている出力 HAL ピンを参照します。	パラレルポートの例： parport.1.pin-16-out Mesa 7i96 の例： hm2_7i96.0.ssr.00.out-02
スクライブオン	スクライブを使用する場合は必須です。 スクライブ電源が接続されている出力 HAL ピンを指します。	パラレルポートの例： parport.1.pin-16-out Mesa 7i96 の例： hm2_7i96.0.ssr.00.out-03
PowerMax 通信	PowerMax シリアル通信を使用する場合に必要です。 通信に使用するシリアルポートを指します。	/ dev / ttyUSB0

Note

HAL ピンのフルネームがわからない場合、ユーザーはベースマシン用に LinuxCNC を起動し、HalShow を実行してすべての HAL ピンの完全なリストを表示できます。

マシンの配線/ブレークアウトボードの構成に合わせて必要なエントリを入力し、[作成]をクリックすると、次のディレクトリに動作する PlasmaC 構成が作成されます：〜/ linuxcnc / configs / <machine_name>

新しく作成された PlasmaC 構成は、ターミナルウィンドウに次のコマンドを入力することで実行できます（「<machine_name>」を PlasmaC configurator に入力されたマシン名に変更します）。

```
linuxcnc ~/linuxcnc/configs/<machine_name>/<machine_name>.ini
```

新しい構成を作成した後、PlasmaC コンポーネントを使用する前にいくつかの初期設定が必要です。

3.16.5.1 初期設定

セクション 5.1 の最後でコマンドを実行した後、LinuxCNC は PlasmaC パネルが表示された状態で実行されているはずです。 [構成]タブをクリックして[構成]パネルを開き、これらの設定がすべてマシンに合わせて調整されていることを確認します。

構成パネルを参照してください

1. ユーザーは推奨される Z 軸設定に精通している必要があります
2. Z 軸をホームにします。
3. トーチの下に何もないことを確認してから、Z 軸 MINIMUM_LIMIT で停止するまで Z 軸をジョグし、Z 軸を選択して[タッチオフ]をクリックして、Z 軸をゼロオフセットに設定します。
4. Z 軸を再びホームにします。

マシンにフロートスイッチが装備されている場合、ユーザーは構成パネルでオフセットを設定する必要があります。 これは、「プローブテスト」サイクルを実行することによって実行されます。

1. 構成パネルのプローブ速度とプローブ高さが正しいことを確認します。 PlasmaC は、機械がフロートスイッチ内でオーバーランを吸収するのに十分な動きをしている限り、Z 軸の全速度でプローブできます。 マシンが適切な場合、ユーザーはプローブの高さを Z 軸の最小値に近い値に設定し、すべてのプローブを全速力で行うことができます。
2. マシンがまだホームになっておらず、ホームポジションにある場合は、マシンをホームに戻します。
3. トーチの下のスラットにいくつかの材料を置きます。
4. プローブテストボタンを押します。
5. Z 軸は下をプローブし、材料を見つけてから、現在選択されている材料で設定されている指定されたピアスの高さまで移動します。 トーチは、<machine_name> .ini ファイルで設定された時間この位置で待機します。 デフォルトのプローブテストの保持時間は 30 秒です。この値は、<machine_name> .ini ファイルで編集できます。 この後、トーチは開始高さに戻ります。
6. トーチがピアスハイトで待機している間に、材料とトーチの先端との間の距離を測定します。
7. 測定値が現在選択されている材料のピアス高さよりも大きい場合は、測定値と指定された値の差だけ、構成パネルの「フロートトラベル」を減らします。 測定値が現在選択されている材料のピアス高さよりも小さい場合は、指定された値と測定値の差だけ、構成パネルの「フロートトラベル」を増やします。
8. 「フロートトラベル」の調整が完了したら、材料とトーチの先端の間の測定距離が現在選択されている材料のピアスの高さと一致するまで、上記の＃4 からプロセスを繰り返します。

Note

トーチが材料に接触してからトーチが上昇してピアスの高さで静止するまでの時間が長すぎると思われる場合は、可能な解決策についてプロービングのセクションを参照してください。

重要

Mesa Electronics THCAD を使用している場合は、電圧スケールの値が数学的に取得されるまでです。ユーザーが製造元のカットチャートからのカット電圧を使用する場合は、実際の電圧を測定し、電圧スケールと電圧オフセットを微調整することをお勧めします。

警告

ユーザーがこれらの測定を行った経験がない場合、プラズマ切断電圧は致命的である可能性があります。

3.16.5.2 既存の PlasmaC 構成を再構成する

Configurator を使用して、ファイルを手動で変更する代わりに、既存の PlasmaC 構成を再構成して設定を変更することもできます。

Configurator は以下のみを変更できます。

- PlasmaC をマシンに接続する HAL ピン。
- PlasmaC で使用されるモード。
- GUI での実行パネルの位置。

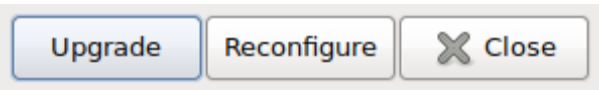
PlasmaC 構成の再構成は、構成ディレクトリにある Configurator から実行されます。

続行する前に、既存の PlasmaC 構成のバックアップコピーを作成することをお勧めします。

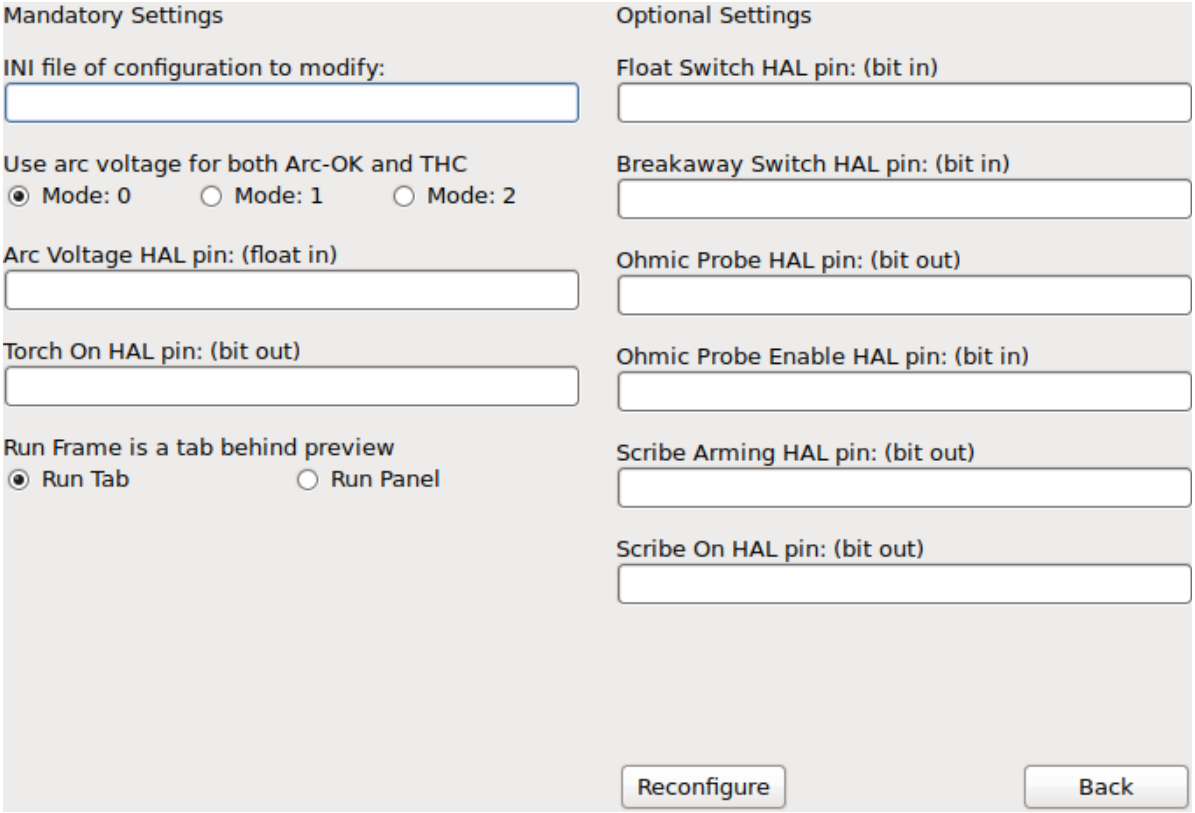
構成ツールを起動するには、ターミナルウィンドウに次のコマンドを入力します。

```
python linuxcnc/<the_users_configuration_directory>/configurator.py
```

選択ウィンドウが表示されます。



選択ウィンドウから[再構成]を選択します。これにより情報ダイアログが表示され、[続行]を選択すると、[再構成]ウィンドウに次のように表示されます。



PlasmaC 構成の<machine_name> .ini ファイルを選択して再構成します。

ユーザーがモードを変更すると、現在選択されているモードに応じて入力ボックスが変更されます。

すべてのエントリが正しい場合は、[再構成]をクリックすると、ユーザーの PlasmaC 構成が再構成されます。

エントリーの説明はここにあります

Note

マシン名と<machine_name> .hal ファイルは変更できません。

3.16.6 その他の PlasmaC セットアップに関する考慮事項

1.1.1.1 ローパスフィルタ

PlasmaC コンポーネントにはローパスフィルターが組み込まれており、使用すると、plasmac.arc-電圧入力ピンに適用されて、誤った電圧測定値を引き起こす可能性のあるノイズをフィルター処理します。 ローパスフィルターは、Halscope を使用して必要な周波数を決定し、ノイズの振幅が問題を引き起こすのに十分な大きさであるかどうかを判断した後にのみ使用する必要があります。 ほとんどのプラズママシンでは、ローパスは不要であり、必要な場合を除いて使用しないでください。

このフィルターに割り当てられた HAL ピンは plasmac.lowpass-frequency であり、デフォルトで 0（無効）に設定されています。 アーク電圧にローパスフィルターを適用するには、ユーザーは、マシンの構成ディレクトリにある<machine_name> _connections.hal ファイルの次のエントリを編集して、ヘルツ（Hz）で測定される適切なカットオフ周波数を追加します。

例えば：

```
setp plasmac.lowpass-frequency 100
```

上記の例では、100Hz のカットオフ周波数が得られます。

3.16.6.1 デバウンスに連絡する

機械的なリレー、スイッチ、または外部干渉からの接点の跳ね返りにより、次のスイッチの動作に一貫性がなくなる可能性があります。

- フロートスイッチ
- オーミックプローブ
- ブレイクアウェイスイッチ
- アーク OK（モード 1 および 2 の場合）

ソフトウェアは接触バウンス期間よりも速いレートをサンプリングできるため、ソフトウェアは、非常に短い期間に発生する入力状態のいくつかの変化として接触バウンスを認識し、非常に速い時間を誤って解釈する可能性があります。 -入力のオフ。 接触バウンスを軽減する 1 つの方法は、入力を「デバウンス」することです。 デバウンスを要約すると、ソフトウェアはデバウンス遅延の変化状態を持つ入力を認識するとすぐに、入力の状態を再度チェックする前に、指定された遅延期間待機します。 デバウンス期間の後、ソフトウェアは状態の変化を入力の通常の動作と見なし、それに応じて反応します。

デバウンス期間は、<machine_name>構成フォルダーの<machine_name> _connections.hal ファイルにある適切なデバウンス値を編集することで変更できます。

遅延が増加するたびに、デバウンス時間に 1 つのサーボスレッドサイクルが追加されます。 例：サーボスレッドの周期が 1000000（ナノ秒で測定）の場合、5 のデバウンス遅延は 5000000ns、つまり 5ms に相当します。

フロートスイッチとオーミックスイッチの場合、これはプローブされた高さの結果が 0.001mm（0.00004 "）増加することに相当します。

一貫した結果を達成しながら、デバウンス値をできるだけ低く保つことをお勧めします。 Halscope を使用して入力をプロットすることは、正しい値を確立するための良い方法です。

PlasmaC で利用可能なコンタクトデバウンスには 2 つのバージョンがあります。 使用される方法は、構成が最初に作成された日時によって決まります。 元のデバウンス設定から代替デバウンス設定に変換するために使用できる自動更新方法はありません。マシンの構成ファイルに行われた可能性のある手動編集を理解せずに必要な変更を適用することは困難であり、動作を「中断」する可能性があるためです。 構成。 ユーザーが元の設定から新しい設定に変更したい場合は、手動で行う必要があります

v0.173（2020 年 9 月 11 日リリース）以降を使用する PlasmaC インストールの場合、デバウンスは、元のデバウンスコンポーネントの後で代替となる HALdbounce コンポーネントを使用して実現されます。 この新しいバージョンでは、個々のデバウンスインスタンスのロードと命名が可能であり、TwopassHAL ファイル処理と互換性があります。

上記の 4 つの信号すべてに個別のデバウンスコンポーネントがあるため、デバウンス期間を各入力に個別に対応させることができます。 <machine_name> _connections.hal ファイルでこれらの値に加えられた変更は、PlasmaC のその後の更新によって上書きされません。

4 つの入力すべてのデフォルトの遅延は、5 つのサーボスレッド周期です。 ほとんどの場合、この値は非常にうまく機能します。 いずれかの入力が機械式スイッチを使用していない場合、それらの入力の遅延を減らすか取り除くことができる場合があります。

ホームスイッチやリミットスイッチなどの他の機器でデバウンスが必要な場合は、ここにリストされている信号に関係なく、HAL ファイルのいずれかにさらにドバウンスコンポーネントを追加できます。

v0.172（2020 年 9 月 10 日リリース）以前を使用した PlasmaC インストールの場合、デバウンスは HAL デバウンスコンポーネントを使用して実現されます。 ユーザーが v0.173 より前のバージョン（2020 年 9 月 11 日リリース）を使用して PlasmaC をインストールし、新しいデバウンス方法を使用したい場合は、「デバウンスタイプの変更」セクションの指示に従う必要があります。

この日付より前に作成された標準構成では、フロートスイッチ、オーミックプローブ、およびブレイクアウェイスイッチの入力のみがデバウンスされ、単一の遅延値が 3 つのスイッチすべてに同時に適用されます。 これらの入力は、信号ごとに専用のデバウンスグループを使用することにより、個別に分離および制御できます。

ホームスイッチやリミットスイッチなどの他の機器にデバウンスが必要な場合は、既存のグループにフィルターを追加するか、<machine_name> _connections.hal ファイルの既存の loadrt デバウンス行を編集して別のグループを追加できます。

既存のグループにフィルターを追加するには：以下のデフォルト行を変更します。

```
loadrt debounce cfg=3
```

に：

```
loadrt debounce cfg=4
```

これにより、別のスイッチ入力信号をデバウンスするために使用できる debounce.0.3 という名前の新しいフィルターが追加されます。

1 つのフィルターでグループを追加するには：以下のデフォルト行を変更します（さらに 2 行追加します）。

```
loadrt debounce cfg=3
```

に：

```
loadrt debounce cfg=3,1
setp debounce.1.delay n
addf debounce.1 servo-thread
```

これにより、debounce.1 という名前の新しいグループが追加され、別のスイッチ入力信号をデバウンスするために使用できる n サーボサイクルの遅延を持つ debounce.1.0 という名前の新しいフィルターが追加されます。 ユーザーは、「n」を新しい入力の適切な遅延量に変更する必要があります。

<machine_name> _connections.hal ファイルを編集することにより、古いデバウンス設定から後のデバウンス設定に手動で変更することができます。 これを行う場合は、この例の connections.hal ファイルで使用されている命名規則に従う必要があります。

<machine_name> .ini ファイルを編集して、次の変更を追加する必要もあります。

```
# required for upgrades (DO NOT CHANGE)
LAST_MAJOR_UPGRADE = 0.144
```

に：

```
# required for upgrades (DO NOT CHANGE)
LAST_MAJOR_UPGRADE = 0.144
DBOUNCE = 1
```

この手順に従わないと、LinuxCNC のロード時にエラーが発生します。

3.16.6.2 デスクトップランチャー

構成の作成時に構成の起動へのリンクが作成されなかった場合、ユーザーはデスクトップを右クリックして[ランチャーの作成]などを選択することにより、構成へのデスクトップランチャーを作成できます。 これにより、ランチャーを作成するためのダイアログが表示されます。 アイコンにわかりやすい短い名前を付け、コマンドに何かを入力して[OK]をクリックします。

ランチャーがデスクトップに表示されたら、ランチャーを右クリックして、選択したユーザーのエディターで編集します。 次のようにファイルを編集します。

```
[Desktop Entry]
Comment=
Terminal=false
Name=LinuxCNC
Exec=sh -c "linuxcnc $HOME/linuxcnc/configs/<machine_name>/<machine_name>.ini"
Type=Application
Icon=/usr/share/pixmaps/linuxcncicon.png
```

ユーザーが GUI ウィンドウの背後でターミナルウィンドウを開きたい場合は、ターミナルラインを次のように変更します。

```
Terminal=true
```

端末の表示は、エラーメッセージや情報メッセージに便利です。

3.16.6.3 PlasmaC ファイル

PlasmaC のインストールが成功すると、次のファイルが構成ディレクトリに作成されます。

ファイル名	機能

<machine_name> .ini	基本システムパラメータ、および PlasmaC に必要なさまざまな設定を含む構成ファイル。
<machine_name> .hal	LinuxCNC へのベースシステム I / OHAL ピン接続を含む HAL 接続ファイル。
<machine_name> _connections.hal	PlasmaC に固有の I / OHAL ピン接続を含む HAL 接続。
<machine_name> _material.cfg	このファイルは、実行パネルからの材料設定を保存するために使用されます
postgui.tcl	ユーザーがカスタマイズするために GUI がロードされた後に実行される HAL ファイル。
tool.tbl	PlasmaC 構成で使用される追加のツール（スクライブなど）のオフセット情報を格納するために使用されるツールテーブル。

Note

<machine_name>は、ユーザーがコンフィギュレーターの「Machine Name」フィールドに入力した名前です。注：カスタムコマンドは、更新中に上書きされないため、<machine_name> _connections.hal ファイルと postgui.hal ファイルで使用できます。

上記のファイルに加えて、ソースディレクトリ内のファイルまたはディレクトリへの次のリンクが作成されます。上記のファイルに加えて、ソースディレクトリ内のファイルまたはディレクトリへの次のリンクが作成されます。

ファイル名	機能
テスト	シミュレーション構成用のテストパネルを含むディレクトリ。
ウィザード	PlasmaC 内の会話機能をサポートするためのシェイプライブラリに関連するファイルを含むディレクトリ。
configurator.py	新しい PlasmaC 構成を構成するために使用される Python スクリプト。 また、既存の PlasmaC 構成を更新/再構成するためにも使用されます。
materialverter.py	さまざまな CAM ソフトウェアからツールライブラリを変換し、PlasmaC マテリアルファイルにデータを入力するために使用される Python スクリプト。
pmx_test.py	正しいシリアルポートを見つけ、PlasmaC と PowerMax 電源間の通信をテストするためのパネルを提供するために使用される Python スクリプト。
version.html	PlasmaC の完全なバージョンと更新履歴をリストする HTML ファイル。

最後に、次のディレクトリが作成されます。

フォルダー	機能
backup	既存のインストールへの更新中に変更されたファイルと、元の.hal および.ini ファイルのバックアップを含むディレクトリ。
plasmac	PlasmaC ソースファイルへのリンクを含むディレクトリ。

新しい構成を初めて実行した後、次のファイルが構成ディレクトリに作成されます。

ファイル名	機能
<machine_name> _config.cfg	このファイルは、構成パネルからの構成設定を保存するために使用されます
<machine_name> _run.cfg	このファイルは、実行パネルからの構成設定を保存するために使用されます
<machine_name> _wizards.cfg	このファイルは、会話型シェイプライブラリの構成設定を保存するために使用されます
Plasmac_stats.var	このファイルは、保存された切削統計を保存するために使用されます

Note

PlasmaC によって作成された構成ファイル（<machine_name> .ini および<machine_name> .hal）は、これらの構成の手動操作を支援するための要件を説明するために表記されています。 これらは、任意のテキストエディタで編集できます。

Note

.cfg ファイルはプレーンテキストであり、任意のテキストエディタで編集できます。

3.16.6.4 INI ファイル

PlasmaC には、次のような特定の<machine_name> .ini ファイル変数が必要です。

【PLASMAC】セクション

```
MODE        = 0 (use external arc voltage in for Arc Voltage)
                  (use external arc voltage in for Arc OK)
              = 1 (use external arc voltage in for Arc Voltage)
                  (use external Arc OK in for Arc OK)
              = 2 (Use external Arc OK in for Arc OK)
                  (use external up/down for THC)
CONFIG_DISABLE = 0 (0=enable or 1=disable the PlasmaC Config Panel)
PAUSED-MOTION-SPEED = n (multiply cut-feed-rate by this value for paused motion speed)
TORCH-PULSE-TIME = n (torch on time when manual pulse requested)
BUTTON_n_NAME = <NAME> (the name of a custom user buttons)
```

```
BUTTON_n_CODE = <CODE> (the code run by a custom user button)
BUTTON_n_IMAGE = <IMAGE> (the image displayed by buttons 10~19)
```

【フィルター】セクション

```
PROGRAM_EXTENSION = .ngc (filter gcode files)
ngc = ./plasmac/plasmac_gcode.py
nc = ./plasmac/plasmac_gcode.py
tap = ./plasmac/plasmac_gcode.py
```

[RS274NGC]セクション

```
RS274NGC_STARTUP_CODE = o<metric_startup> call (machine startup G-Code)
SUBROUTINE_PATH = ./:./plasmac:../../nc_files/subroutines (./ must be in this path)
FEATURES = 12 (for reading .ini and HAL variables)
USER_M_PATH = ./:./plasmac (for M190 material change)
```

重要

G64 に関連する RS274NGC_STARTUP_CODE 情報のパス許容値を参照してください。

[HAL]セクション

```
TWOPASS = on (needed for multiple .hal files)
HALFILE = <machine_name>.hal (your base machine .hal file)
HALFILE = plasmac.tcl (the standard PlasmaC .hal file )
HALFILE = <machine_name>_connections.hal (PlasmaC connections to the machine)
HALFILE = HALUI = halui (required)
```

<machine_name> .hal ファイルでは、必要に応じてスクライブを追加できるように、loadrtmotmod 行の最後に num_spindles = [TRAJ] SPINDLES が追加されています。

Note

このファイルは PlasmaC の更新によって上書きされないため、ユーザーはカスタム HAL コマンドを <machine_name> _connections.hal ファイルに配置できます。

[TRAJ]セクション

```
SPINDLES = 3
```

[AXIS_X]セクション

```
MAX_VELOCITY = double the value in the corresponding joint
MAX_ACCELERATION = double the value in the corresponding joint
OFFSET_AV_RATIO = 0.5
```

[AXIS_Y]セクション

```
MAX_VELOCITY = double the value in the corresponding joint
MAX_ACCELERATION = double the value in the corresponding joint
OFFSET_AV_RATIO = 0.5
```

[AXIS_Z]セクション

```
MIN_LIMIT = the top of your slats or just below
MAX_VELOCITY = double the value in the corresponding joint
MAX_ACCELERATION = double the value in the corresponding joint
OFFSET_AV_RATIO = 0.5
```

Note

PlasmaC は、すべての Z 軸の動き、および一時停止中の消耗品の変更のために X 軸や Y 軸を移動するために、LinuxCNC 外部オフセット機能を使用します。 この機能の詳細については、LinuxCNC ドキュメントの外部軸オフセットをお読みください。

【PLASMAC】セクション

```
FONT = sans 10 (valid font sizes are from 9 to 15 inclusive)
THEME = Clearlooks (any installed theme, only for the plasmaC tabs)
WINDOW_SIZE = 0 (0 = minimum size to suit font, 1 = maximised, width x height = custom
size  -
)
AXIS_ORIENT = portrait
```

上記のパラメーターが指定されていない場合のデフォルトは次のとおりです。

```
FONT = sans 10
THEME = current system theme
WINDOW_SIZE = minimum size to suit font
AXIS_ORIENT = landscape
```

FONT サイズを変更すると、最小ウィンドウサイズが変更されます。 PlasmaC には、標準の Axis ウィンドウよりも大きなウィンドウサイズが必要です。 ウィンドウサイズは、ユーザーが実行パネルをタブとして表示するか、GUI の右側のパネルとして表示するかによって異なります。 ポートレートモードを使用する場合、ウィンドウの高さは、ユーザーが回転軸を持っているかどうかによって異なります。 ウィンドウが画面に完全に収まらない場合、ユーザーはフォントサイズを小さくする必要があります。

カスタムウィンドウサイズは、幅 x 高さとして指定されます。例： 1600 x900。スペースは無視されますが、読みやすくするために使用される場合があり、x は小文字または大文字の場合があります。

Note

WINDOW_SIZE が MAXIMISED に置き換わりました。 下位互換性のために、これらの名前は交換可能です。

【表示】セクション

```
TOOL_EDITOR = tooledit x y
USER_COMMAND_FILE = plasmac_axis.py.py
EMBED_TAB_NAME = Statistics
EMBED_TAB_COMMAND = gladevcp -c plasmac_stats -x <XID> -u
./plasmac/plasmac_stats.py -H ./  -
plasmac/plasmac_stats.hal ./plasmac/plasmac_stats.glade
#use one of the next two
#run frame in tab behind preview
EMBED_TAB_NAME = Plasma Run
EMBED_TAB_COMMAND = gladevcp -c plasmac_run -x <XID> -u ./plasmac/plasmac_run.py -
H ./  -
plasmac/plasmac_run.hal ./plasmac/plasmac_run_tab.glade
#run frame in panel on right side
#GLADEVCP = -c plasmac_run -u ./plasmac/plasmac_run.py -H
./plasmac/plasmac_run.hal ./  -
plasmac/plasmac_run_panel.glade
EMBED_TAB_NAME = Plasma Config
```

```
EMBED_TAB_COMMAND = gladevcp -c plasmac_config -x <XID> -u
./plasmac/plasmac_config.py -H  -
./plasmac/plasmac_config.hal ./plasmac/plasmac_config.glade
EMBED_TAB_NAME = Extras
EMBED_TAB_COMMAND = gladevcp -c plasmac_wizards -x {XID} -u
./plasmac/plasmac_wizards.py ./  -
plasmac/plasmac_wizards.glade
```

実行ウィンドウは、次のいずれかとして表示できます。

1.ワイドスクリーンディスプレイに適した AxisGUI の右側のパネル。

2. 4：3 の比率の表示に適した[プレビュー]タブの後ろのタブ。

[表示]セクション

```
EMBED_TAB_NAME = plasmac_buttons
EMBED_TAB_LOCATION = box_cooling
EMBED_TAB_COMMAND = gladevcp -c plasmac_buttons -x <XID> -u
./plasmac/plasmac_buttons.py -  -
H ./plasmac/plasmac_buttons.hal ./plasmac/plasmac_buttons.glade
EMBED_TAB_NAME = plasmac_control
EMBED_TAB_LOCATION = box_spindle
EMBED_TAB_COMMAND = gladevcp -c plasmac_control -x <XID> -u
./plasmac/plasmac_control.py -  -
H ./plasmac/plasmac_control.hal ./plasmac/plasmac_control.glade
EMBED_TAB_NAME = Statistics
EMBED_TAB_LOCATION = ntb_preview
EMBED_TAB_COMMAND = gladevcp -c plasmac_stats -x <XID> -u
./plasmac/plasmac_stats.py -H ./  -
plasmac/plasmac_stats.hal ./plasmac/plasmac_stats.glade
EMBED_TAB_NAME = Plasma Run
#use one of the next two
#run panel in tab behind preview
EMBED_TAB_LOCATION = ntb_preview
EMBED_TAB_COMMAND = gladevcp -c plasmac_run -x <XID> -u ./plasmac/plasmac_run.py -
H ./  -
plasmac/plasmac_run.hal ./plasmac/plasmac_run_tab.glade
#run panel in panel on left side
#EMBED_TAB_LOCATION = box_left
```

```
#EMBED_TAB_COMMAND = gladevcp -c plasmac_run -x <XID> -u ./plasmac/plasmac_run.py
-H ./  -
plasmac/plasmac_run.hal ./plasmac/plasmac_run_panel.glade
EMBED_TAB_NAME = Plasma Config
EMBED_TAB_LOCATION = ntb_preview
EMBED_TAB_COMMAND = gladevcp -c plasmac_config -x <XID> -u
./plasmac/plasmac_config.py -H  -
./plasmac/plasmac_config.hal ./plasmac/plasmac_config.glade
EMBED_TAB_NAME = plasmac_monitor
EMBED_TAB_LOCATION = box_tool_and_code_info
EMBED_TAB_COMMAND = gladevcp -c plasmac_monitor -x <XID> -u
./plasmac/plasmac_monitor.py -  -
H ./plasmac/plasmac_monitor.hal ./plasmac/plasmac_monitor.glade
EMBED_TAB_NAME = Extras
EMBED_TAB_LOCATION = ntb_preview
EMBED_TAB_COMMAND = gladevcp -c plasmac_wizards -x {XID} -u
./plasmac/plasmac_wizards.py  -
./plasmac/plasmac_wizards.glade
```

実行ウィンドウは、次のいずれかとして表示できます。

1.ワイドスクリーンディスプレイに適した GmoccapyGUI の左側にあるパネル。

2. 4：3 の比率の表示に適した[プレビュー]タブの後ろのタブ。

3.16.7 PlasmaC の使用

PlasmaC が正常にインストールされると、G コードカットプログラムの一部として Z 軸の動きは必要ありません。 実際、カットプログラムに Z 軸参照が存在する場合、標準の PlasmaC 構成では、プログラムのロードプロセス中にそれらが削除されます。

PlasmaC を確実に使用するために、ユーザーは座標系オフセット（G54-G59.3）以外の Z 軸オフセットを使用しないでください。

PlasmaC が Z の動きを自動的に処理する方法により、手動カットを実行したり G-を実行したりする前に、Z 軸を Z 軸の最大制限（ワークピースから最も遠いトーチ）より約 5mm（0.2 "）下に駐車することをお勧めします。コードプログラム。

1.1.1.1 PlasmaCGUI パネル

PlasmaC は GUI にいくつかの変更を加えます。 一部の追加は永続的に表示されるパネルであり、その他はプレビュータブの後ろのタブです。

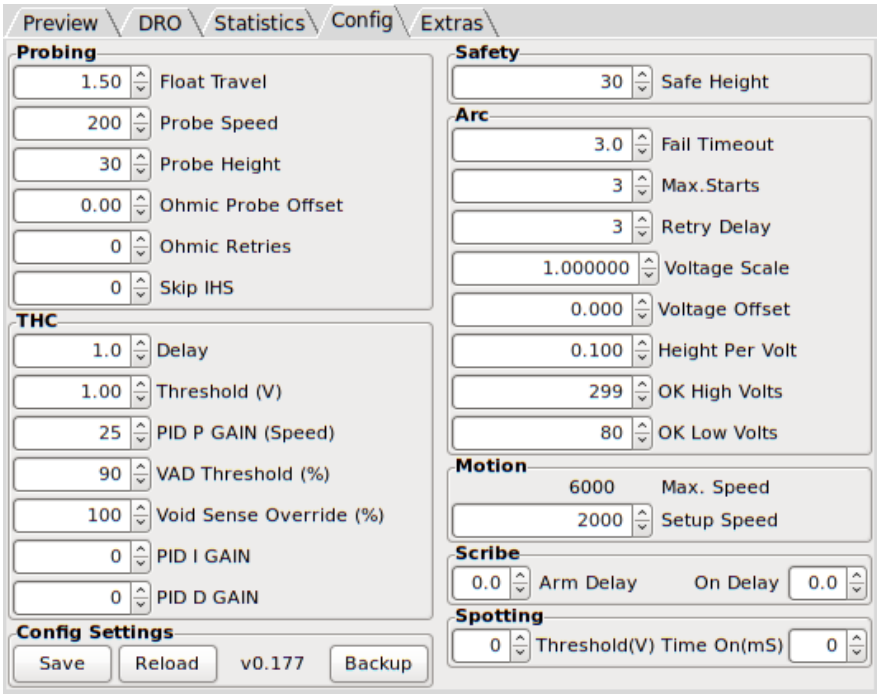
一部の機能/機能は特定のモードでのみ使用され、選択した PlasmaC モードで必要とされない場合は表示されません。

このパネルは、頻繁に変更されない構成パラメーターを表示するために使用されます。

このパネルを無効にして、許可されていない人がマシンの設定を変更できないようにすることができます。 これは、<machine_name> .ini ファイルに次の変数を設定することで実現されます。

```
[PLASMAC] CONFIG_DISABLE = 1
```

許可された担当者が構成パネルを無効にした後に構成を変更する必要がある場合は、plasmac_config.config-disable という名前の HAL ピンを 0（ゼロ）に設定して、パネルを有効にすることができます。 このピンは、キャビネットの物理的なキースイッチなどに接続して、許可された担当者のみが構成パネルで設定を変更できるようにすることができます。



プロービング

名前	説明
フロートトラベル	これにより、フロートスイッチ回路が完成する前にフロートスイッチが移動する移動量が設定されます。 この距離は、[プローブテスト]ボタンと、初期設定で説明されている方法を使用して測定できます。
プローブ速度	これにより、トーチがプローブの高さに移動した後、トーチが材料を見つけるためにプローブする速度が設定されます。
プローブの高さ	これにより、プローブ速度が開始する Z 軸の最小制限を超える高さが設定されます。
オーミックプローブオフセット	これにより、オーミックプローブが成功した後にトーチが移動する材料の上の距離が設定されます。 これは主に、高いプロービング速度を補正するために使用され

	ます。
オーミックリトライ	これにより、PlasmaC が材料検出のためにフロートスイッチにフォールバックする前に、失敗したオーミックプローブを再試行する回数が設定されます。
IHS をスキップ	これにより、現在のカットで初期高さ検知（プローブ）をスキップできるかどうかを判断するために使用される距離しきい値が設定されます。IHS スキップを参照してください。

Note

トーチが材料に接触してからトーチが上昇してピアスの高さで静止するまでの時間が長すぎると思われる場合は、可能な解決策についてプロービングのセクションを参照してください。

THC

名前	モード	内容
遅れ	0、1、2	これは、アーク OK 信号が受信されてからトーチ高さコントローラー（THC）がアクティブになるまでに測定される遅延（秒単位）を設定します。
しきい値（V）	0、1、2	これにより、THC がトーチの高さを修正するための動作を行う前に、ターゲット電圧から許容される電圧変動が設定されます。
PID P ゲイン（速度）	0、1	これにより、THCPID ループの比例ゲインが設定されます。 これは、THC が高さの変化を修正しようとする速度とほぼ同じです。
VAD しきい値（%）	0、1	（Velocity Anti Dive）これは、トーチダイブを防ぐために THC をロックする前に、マシンが減速できる現在のカット送り速度のパーセンテージを設定します。
ボイドセンスオーバーライド（%）	0、1	これにより、トーチダイブを防ぐために THC をロックするために必要なカット電圧の変化のサイズが設定されます（値が大きいほど、THC をロックするためにより大きな電圧変化が必要になります）。
PIDI ゲイン	0、1	これにより、THCPID ループの積分ゲインが設定されます。 積分ゲインは、時間の経過に伴うシステムのエラーの合計に関連付けられており、常に必要なわけではありません。
PIDD ゲイン	0、1	これにより、THCPID ループの微分ゲインが設定されます。 微分ゲインは、システムを減衰させ、過補正振動を低減するように機能し、常に必要なわけではありません。

Note

PID ループの調整は複雑なプロセスであり、このユーザーガイドの範囲外です。 PID ループの理解と調整に役立つ情報源はたくさんあります。 THC が十分な速度で補正を行っていない場合は、システムが正常

に動作するまで、P ゲインを少しずつ増やすことをお勧めします。 P ゲインを大きく調整すると、過補正や発振が発生する可能性があります。

安全性

名前	内容
安全な高さ	これにより、高速移動を実行する前にトーチが収縮する材料の上の高さが設定されます。 ゼロに設定すると、Z 軸の最大高さが安全な高さに使用されます。

アーク(Arc)

名前	オード	内容
失敗タイムアウト	0、1、2	これにより、PlasmaC が「トーチオン」のコマンドを実行してから Arc OK 信号を受信してからタイムアウトになり、エラーメッセージが表示されるまでの時間（秒単位）が設定されます。
最大。 開始	0、1、2	これにより、PlasmaC がアークの開始を試行する回数が設定されます。
再試行遅延	0、1、2	これにより、アーク障害が発生してから別のアーク開始が試行されるまでの時間（秒単位）が設定されます。
電圧スケール	0、1	これはアーク電圧入力スケールを設定し、正しいアーク電圧を表示するために使用されます。 初期設定については、キャリブレーション値を参照してください。
電圧オフセット	0、1	これはアーク電圧オフセットを設定し、アーク電圧入力がゼロのときにゼロボルトを表示するために使用されます。 初期設定については、キャリブレーション値を参照してください。
ボルトあたりの高さ	0、1、2	これにより、アーク電圧を 1 ボルト変更するためにトーチが移動する必要がある距離が設定されます。
OK ハイボルト	0	これにより、ArcOK 信号が有効になる電圧しきい値が設定されます。
OK 低ボルト	0	これにより、ArcOK 信号が有効になる電圧しきい値が設定されます。

Note

モード 0 で OKLowVolts と OKHigh Volts を設定する場合、有効な Arc OK 信号を受信するには、安定したアークのカット電圧が OK Low Volts 値より大きく、PlasmaC の OK HighVolts 値より低くなければなりません。 さらに明確にするために、有効なアーク OK を得るには、アーク電圧が 2 つの制限の間にある必要があります。

モーション

名前	内容
最大。 スピード	Z 軸が可能な最大速度を表示します（これは<machine_name> .ini ファイルによっ

	て制御されます）。
セットアップ速度	セットアップの Z 軸速度が移動します（プローブの高さ、ピアスの高さ、カットの高さなどへの移動）。

Note

セットアップ速度は、最大で表示される速度が可能な THC 速度には影響しません。 スピードフィールド。

スクライブ

名前	内容
アームディレイ	これにより、スクライブコマンドが受信されてからスクライブがアクティブになるまでの遅延（秒単位）が設定されます。 これにより、スクライブをアクティブにする前に、スクライブが材料の表面に到達できるようになります。
遅延時	これにより、モーションを開始する前にスクライブメカニズムを開始できるように遅延（秒単位）が設定されます。

スポッティング

名前	内容
しきい値（V）	これにより、遅延タイマーが開始するアーク電圧が設定されます。 0V は、トーチオン信号がアクティブになると遅延を開始します。
タイムオン（mS）	これは、しきい値電圧に達した後にトーチがオンになる時間の長さ（ミリ秒単位）を設定します。

保存および再読み込みボタン

[保存]ボタンは、現在表示されているパラメータを<machine_name> _config.cfg ファイルに保存します。

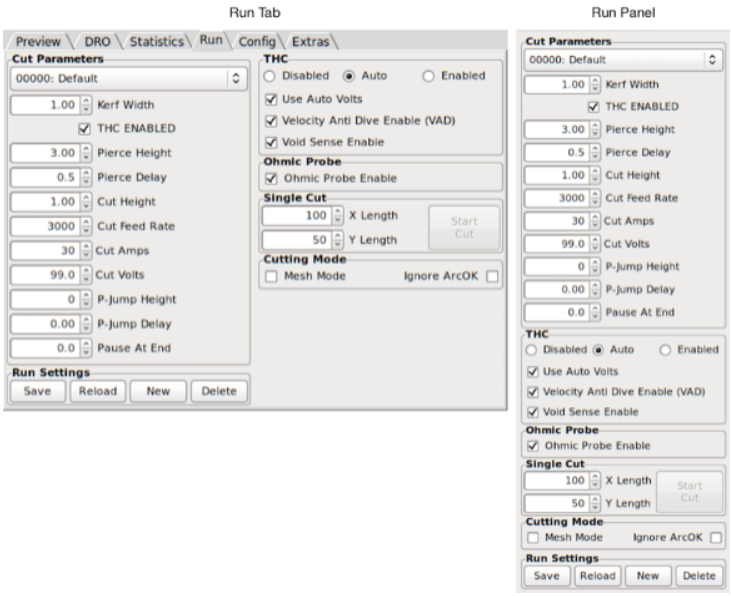
[再読み込み]ボタンは、<machine_name> _config.cfg ファイルからすべてのパラメーターを再読み込みします。

バージョンラベルには、現在の PlasmaC バージョンが表示されます。

[バックアップ]ボタンは、アーカイブまたは障害診断を支援するための完全なマシン構成バックアップを作成します。 マシン構成の圧縮バックアップは、ユーザーの Linux ホームディレクトリに保存されます。 ファイル名は <machine_name> _ <version_info> .tar.gz になります。ここで、<machine_name>はコンフィギュレーターに入力されたマシン名であり、<version_info>はユーザーが使用している現在の PlasmaC バージョンです。

このパネルには、現在のカットでアクティブなパラメータが表示されます。

このパネルには、プレビュータブの後ろにあるタブと、GUI の右側にあるパネルの 2 つの形式があります。フォーマットは異なりますが、どちらもカットパラメータを制御するための同じ機能を提供します。



<machine_name> .ini ファイルの 2 つの形式を切り替える方法については、Axis [DISPLAY]セクションまたは Gmoccapy [DISPLAY]セクションを参照してください。 または、ユーザーはコンフィギュレーターを再度実行して、別のオプションを選択することもできます。

カットパラメータ

名前	内容
素材	上部のドロップダウンメニューは、現在のマテリアルカットパラメータを手動で選択するために使用されます。 マテリアルファイルにマテリアルがない場合は、デフォルトのマテリアルのみが表示されます。
カーフ幅	これにより、現在選択されているマテリアルのカーフ幅が設定されます。
THC 対応	このチェックボックスは、現在選択されている材料の THC を有効または無効にします。
ピアスハイト	これにより、現在選択されているマテリアルのピアスの高さが設定されます。
ピアス遅延	これにより、現在選択されているマテリアルのピアス遅延（秒単位）が設定されます。
カット高さ	これにより、現在選択されているマテリアルのカット高さが設定されます。
送り速度を下げる	これにより、現在選択されている材料のカット送り速度が設定されます。
カットアンペア	これにより、現在選択されている材料のカットアンペア数が設定されます。 PowerMax 通信が使用されていない限り、これはオペレーターのみの視覚的なインジケーターです。
カットボルト	これにより、現在選択されている材料のカット電圧が設定されます。
P ジャンプの高さ	これにより、現在選択されているマテリアルのパドルジャンプの高さが設定されます。 通常、より厚い材料に使用されるパドルジャンプにより、トーチはピアスの高さとカットの高さの中間のステップを持つことができます。 設定されている

	場合、トーチは一定期間ピアス高さからPジャンプ高さ（Pジャンプ遅延）に進み、その後カット高さに進んで溶融水たまりを効果的に「ジャンプ」します。
Pジャンプ遅延	これにより、現在選択されているマテリアルのパドルジャンプ遅延（秒単位）が設定されます。 この値は、トーチがカット高さに進む前にPジャンプ高さに留まる時間の長さを設定するため、Pジャンプ高さが設定されている場合は必須です。
最後に一時停止	これにより、M5コマンドを続行してトーチをオフにして上げる前に、トーチがカットの終了時にオンのままになる時間（秒単位）が設定されます。 詳細については、カットの終了時に一時停止を参照してください。
ガス圧	これにより、現在選択されている材料のガス圧が設定されます。 この設定は、PowerMax通信が使用されている場合にのみ有効です。 0 = PowerMaxの自動圧力モードを使用します。
カットモード	これにより、現在選択されているマテリアルのカットモードが設定されます。 この設定は、PowerMax通信が使用されている場合にのみ有効です。 1 =通常 2 = CPA（コンスタントパイロットアーク） 3 =ガウジ/マーク

THC

名前	モード	内容
State	0、1、2	Auto = THCステータスは、[THC ENABLED]チェックボックスの状態によって判断したり、実行中のプログラムの適切なMコードによって制御したりできます。 有効=実行中のプログラムの適切なモードによって無効にされない限り、THCはオンになります。
オートボルトを使用する	0、1	チェック済み= THCターゲット電圧は、THC遅延の期限が切れた後にアーク電圧をサンプリングすることによって確認されます。 チェックなし= THCターゲット電圧は、「カットボルト」材料パラメータから取得されます。
Velocity Anti Dive Enable（VAD）	0、1、2	チェック済み= VelocityAnti-Diveが有効になっています。 オフ= VelocityAnti-Diveが無効になっています。
Void Sense Enable	0、1	チェック済み=ボイド検知が有効になっています。 オフ=ボイド検知が無効になっています。

オーミックプローブ

名前	内容
オーミックプローブイネーブル	このチェックボックスは、オーミックプローブ入力を有効または無効にします。

Note

オーミックプローブ入力が無効になっている場合でも、オーミックプローブ LED はプローブ入力のステータスを示しますが、オーミックプローブの結果は無視されます。

シングルカット

名前	内容
X 軸の長さ	これにより、1 回のカットで移動する X 軸の距離が設定されます。
Y 軸の長さ	これにより、1 回のカットで移動する Y 軸の距離が設定されます。
カットボタンを開始	このボタンを押すと、シングルカットが開始されます。

カッティングモード

名前	内容
メッシュモード	このチェックボックスは、エキスパンドメタルの切断のメッシュモードを有効または無効にします。 このチェックボックスは、通常の切断中はいつでも有効または無効にできます。 さらに、このモードは、実行中のプログラムの適切な M コードを介して有効または無効にできます。 メッシュモード： -機械の動作を開始するには、ArcOK 信号が必要です。 -THC を無効にします。 -アーク OK 信号が失われた場合、機械の動きを停止しません。 -PowerMax 通信が使用されている場合、CPA モードが自動的に選択されます。 詳細については、メッシュモード（エキスパンドメタル）を参照してください。
ArcOK を無視する	このチェックボックスは、PlasmaC が ArcOK 信号を無視するかどうかを決定します。 このチェックボックスは、通常の切断中はいつでも有効または無効にできます。 さらに、このモードは、実行中のプログラムの適切な M コードを介して有効または無効にできます。 Arc OK モードを無視します： -マシンオプションを開始する前に ArcOK 信号を受信する必要はありません。 「トーチオン」信号が出されます。 -THC を無効にします。 -アーク OK 信号が失われた場合、機械の動きを停止しません。 詳細については、「アーク OK を無視する」を参照してください。

PowerMax 通信

Note

このフレームは、PM_PORT が<machine_name> .ini ファイルの[PLASMAC]セクションで構成されている場合にのみ表示されます。

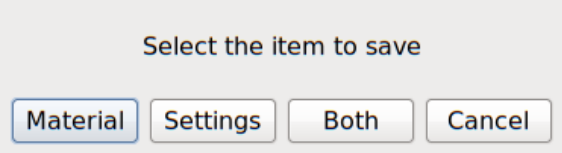
名前	内容
有効	このチェックボックスは、PowerMax への通信を有効または無効にします。
スターテス	PowerMax 通信が有効になっている場合、これは次のいずれかを表示します：接続中、接続済み、通信エラー、または障害コード。

詳細については、「PowerMax 通信」セクションを参照してください。

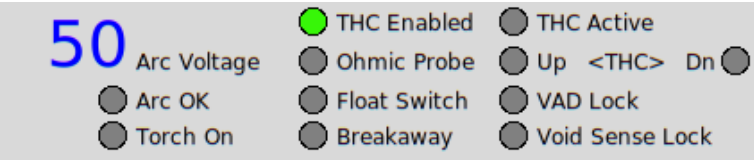
実行設定

名前	内容
保存	このボタンをクリックすると、次のオプションを含むダイアログボックスが表示されます。 マテリアル-このボタンは、現在表示されているマテリアルのカットパラメータを保存します。 設定-このボタンは、THC、オーミックプローブ、シングルカット、カッティングモード、および PowerMax 通信設定。 [カットパラメータ]セクションの設定は保存されません。 両方-このボタンは、実行パネルに表示されるすべての設定を保存します。 キャンセル-このボタンは、保存せずにダイアログボックスを閉じます。
リロード	このボタンは、実行パネルに表示されているすべての設定を再ロードし、現在選択されているマテリアルを再表示します。 以前に保存した設定に関係なく、PowerMax の場合再ロードを押す前に通信が有効になり、その後も有効のままになります。
新しい	このボタンを使用すると、新しいマテリアルをマテリアルファイルに追加できます。 ユーザーは材料番号と材料名の入力を求められ、他のすべてのパラメータは現在選択されている材料から読み取られます。 入力すると、PlasmaC はマテリアルファイルをリロードし、新しいマテリアルを表示します。 次に、新しいマテリアルのカットパラメータを調整して保存する必要があります。
消去	このボタンは、マテリアルを削除するために使用されます。 それを押すと、ユーザーは削除する材料番号の入力を求められ、ユーザーが確実であることを確認するために再度プロンプトが表示されます。 削除後、マテリアルファイルが再ロードされ、ドロップダウンリストにデフォルトのマテリアルが表示されます。

[保存]ダイアログボックスの例を以下に示します。



モニターパネルは、関連するさまざまなI / Oのステータスを表示するために使用されます。 Axis と Gmoccapy の両方に同様のモニターパネルがあります。



名前	モード	内容
アーク電圧	0、1	実際のアーク電圧を表示します。
アーク OK	1、2	ArcOK 信号のステータスを示します。
トーチオン	0、1、2	トーチオン出力信号のステータスを示します。
THC 対応	0、1、2	カット中に THC を有効にするか無効にするかを示します。
オーミックプローブ	0、1、2	プローブが材料を検知したことを示します。
フロートスイッチ	0、1、2	フロートスイッチがアクティブになっていることを示します。
離脱する	0、1、2	トーチ離脱センサーが作動していることを示します。
THC アクティブ	0、1、2	THC が Z 軸をアクティブに制御していることを示します。
THC アップ	0、1、2	THC が Z 軸に上昇を命じていることを示します。
THC ダウン	0、1、2	THC が Z 軸を下げるように命令していることを示します。
VAD ロック	0、1、2	カット速度が構成パネルで設定された VAD しきい値のパーセンテージを下回ったため、THC が現在の高さでロックされていることを示します。
ボイドセンスロック	0、1	ボイドが検知されたために THC がロックされていることを示します。

Note

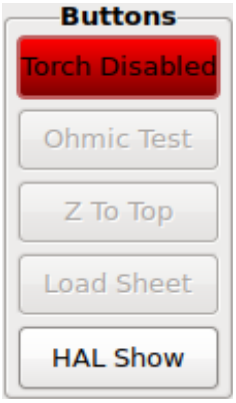
オーミックプローブ LED は、オーミックプローブが有効か無効かに関係なく、プローブ入力のステータスを示します。

ボタンパネルには、機械の操作に役立つボタンが含まれています。

両方の GUI には永続的なトーチ有効化ボタンがあり、他のすべてのボタンは<machine_name> .ini ファイルでユーザーがプログラムできます。 Axis には 5 つのユーザーボタンがあり、Gmoccapy には 4 つのユーザーボタンがあります。

軸ボタンのレイアウト：

Gmoccapy ボタンのレイアウト：



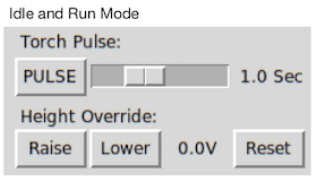
トーチの無効化/有効化ボタンは、トーチの有効化と無効化を切り替えます。 このボタンは、PlasmaC を最初に実行したときにデフォルトで「トーチ無効」に設定されており、材料の切断を開始する前にクリックして「トーチ有効」に変更する必要があります。

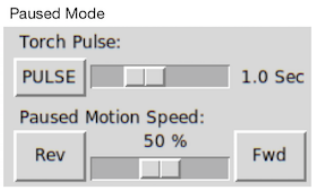
このボタンで「トーチ無効」が表示されている場合、ロードされたプログラムを実行すると、マシンはトーチを発射せずにサイクルを実行します。 これは「ドライラン」と呼ばれることもあります。

コントロールパネルを使用すると、ユーザーはトーチを希望の時間パルスし、高さ制御を手動で無効にし、一時停止中の動作中に希望の速度でマシンを操作できます。 Axis と Gmoccapy はどちらも似ていますが、Gmoccapy コントロールパネルがフィードと高速オーバーライドを含む既存のフレームに統合されている点が異なります。

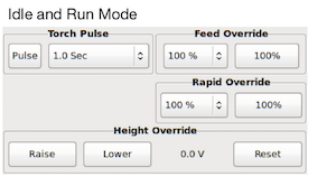
マシンがホームに戻ると、トーチパルスと高さオーバーライドがアイドルモードと実行モードで永続的に表示されます。 マシンが一時停止モードに入ると、一時停止モーション速度が有効になります。 さまざまな状態の例を以下に示します。

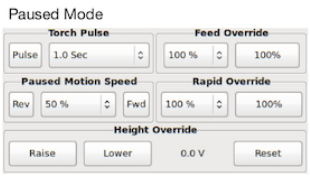
軸コントロールパネル：





Gmoccapy コントロールパネル：





トーチパルス

名前	内容
パルス	このボタンはトーチをパルスします。
パルスタイムスライダー	このスライダーは、パルス時にトーチがオンのままになる時間（秒単位）を設定します。

高さのオーバーライド

名前	内容
高める	このボタンを押すたびに、構成パネルのアークフレームの[ボルトあたりの高さ]フィールドで設定された値だけトーチの高さが上がります。
低い	このボタンを押すたびに、構成パネルのアークフレームの[ボルトあたりの高さ]フィールドで設定された値だけトーチの高さが低くなります。
リセット	このボタンは、手動による高さの上書きをキャンセルします。

一時停止したモーション速度

プログラムが一時停止した場合、このインターフェースにより、X / Y モーションがプログラムされたパスを逆方向または順方向にたどることができます。

名前	内容
Rev	プログラムが一時停止した場合、このボタンは、プログラムが一時停止する前に実行された最後の M3 コマンド、または PlasmaC が実行を試みたコマンドに到達するまで、プログラムされたパスに沿ってマシンを逆方向に移動します。
モーションスピードスライダー	このスライダーは、現在選択されている材料の実行パネルのカットパラメータセクションに表示される現在のカット送り速度のパーセンテージを設定します。
Fwd	プログラムが一時停止した場合、このボタンは、プログラムが終了するまで、プログラムされたパスに沿ってマシンを無期限に前進させ、M3 コマンドをスキップします。

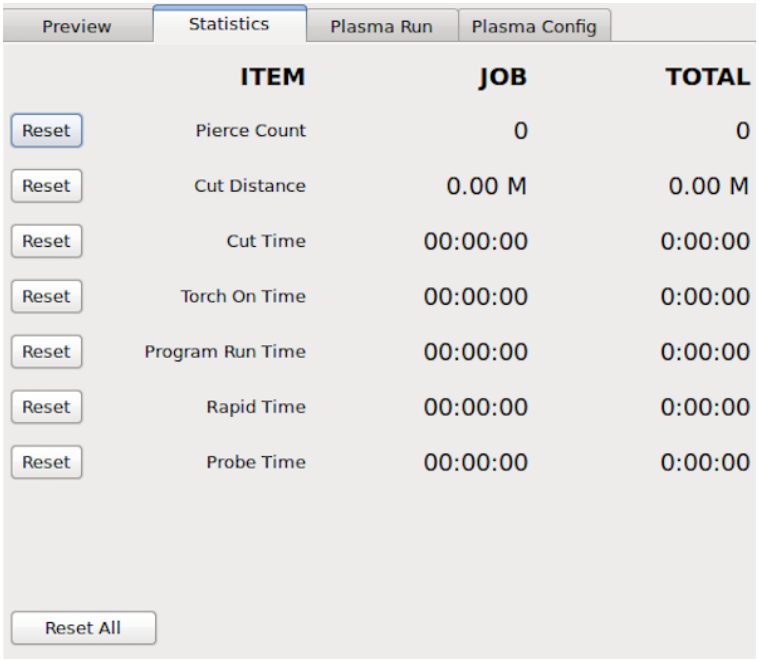
失敗したカットからの高度なリカバリについては、カットリカバリを参照してください。

統計パネルは、消耗品の摩耗とジョブの実行時間を追跡できるようにする統計を提供します。

これらの統計は、現在のジョブと現在の合計について表示されます。

次のプログラムが実行されると、前のジョブ統計がリセットされます。

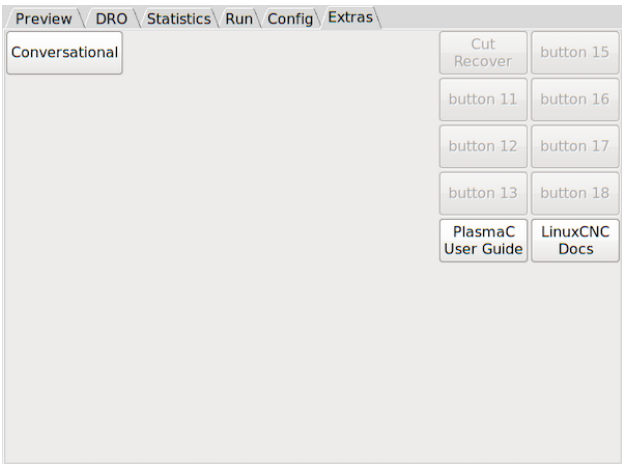
合計値は、対応する[リセット]ボタンをクリックして個別にリセットすることも、[すべてリセット]をクリックしてすべて一緒にリセットすることもできます。



Extras Panel には、ユーザーがカスタマイズできる 10 個の追加ボタンと、[会話]ダイアログボックスを表示するためのボタンがあります。 会話型形状ライブラリを使用すると、CAM ソフトウェアを使用せずに、さまざまな単純な形状をすばやくプログラムしてすばやく切断できます。

会話機能の詳細については、会話形状ライブラリを参照してください。

カスタムユーザーボタンの詳細については、カスタムユーザーボタンを参照してください。



<machine_name> .ini ファイルの[DISPLAY]セクションで適切な編集を行うことにより、Extras パネルを無効にすることができます。 これは、PlasmaC ソフトウェアの動作には影響しません。

エクストラパネルを無効にするには、次のように変更します。

```
EMBED_TAB_NAME = Extras
```

```
EMBED_TAB_COMMAND = gladevcp -c plasmac_wizards -x {XID} -u
./plasmac/plasmac_wizards  -
.py ./plasmac/plasmac_wizards.glade
```

に：

```
#EMBED_TAB_NAME = Extras
#EMBED_TAB_COMMAND = gladevcp -c plasmac_wizards -x {XID} -u ./plasmac/  -
plasmac_wizards.py ./plasmac/plasmac_wizards.glade
```

3.16.7.1 必須コード

プリアンブルコード、ポストアンブルコード、および X / Y モーションコードを除いて、PlasmaC が G コードプログラムを実行するために必須の G コード構文は、カットを開始するための M3 $ 0 S1 と、カットを終了するための M5 $ 0 のみです。

ユーザーが複数のツールを有効にせずに PlasmaC を使用している場合は、M3 $ 0S1 の代わりに M3S1 を使用して切断ジョブを開始し、M5 $ 0 の代わりに M5 を使用できます。

3.16.7.2 座標

推奨される Z 軸設定を参照してください。

LinuxCNC（PlasmaC）が開始されるたびに、ジョイントホーミングが必要です。 これにより、LinuxCNC（PlasmaC）は各軸の既知の座標を確立し、ソフト制限を<machine_name> .ini ファイルで指定された値に設定して、通常の使用中にマシンがハードストップにクラッシュするのを防ぎます。

マシンにホームスイッチがない場合、ユーザーは、ホーミングする前に、すべての軸が<machine_name> .ini ファイルで指定されたホーム座標にあることを確認する必要があります。

マシンにホームスイッチがある場合、ジョイントがホームされると、指定されたホーム座標に移動します。

マシンの構成に応じて、[すべてホーム]ボタンがあるか、各軸を個別にホームにする必要があります。 適切なボタンを使用して、マシンをホームに戻します。

初期設定のセクションで説明したように、PlasmaC を初めて使用するときは、トーチの下に何もないことを確認してから、Z 軸 MINIMUM_LIMIT で停止するまで Z 軸をジョギングしてから、Z 軸で[タッチオフ]をクリックすることをお勧めします。 Z 軸をゼロオフセットに設定するために選択しました。 これを再度行う必要はありません。

プログラムを開始する前に、Z 軸を移動の最上部から約 5mm（0.196 "）までジョグします（Z 軸 MAXIMUM_LIMIT）。PlasmaC はプログラムの実行中にすべての Z 軸の動きを制御するため、Z 軸をこの位置のままにして駐車します。 プログラムが完了したときのこの位置の Z 軸。

ユーザーが毎回テーブルのまったく同じ場所に材料を配置する場合、ユーザーはCAMソフトウェアによって確立された対応するX0 Y0位置にマシンのX軸とY軸をジョグし、次に両方の軸をタッチオフすることができます。ゼロオフセット。

ユーザーが材料をテーブルにランダムに配置する場合は、プログラムを開始する前に、適切な位置でX軸とY軸をタッチオフする必要があります。

3.16.7.3 パストレランス

提供されているRS274NGC_STARTUP_CODEファイル：metric_startup.ngcおよびimperial_startup.ngcは、G64コマンドのモーションブレンディングパスの許容値をそれぞれ0.1mmおよび0.004 "に設定します。P値は、マシンがたどる実際のカットパスが逸脱する可能性のある量に対応します。 LinuxCNC（PlasmaC）がいずれかの段階でE-stop信号を受信した場合、パス許容値はデフォルト（P値なし）に設定され、可能な限り最高の速度を維持し、コーナーを丸めます。 これを防ぐには、各Gコードファイルのヘッダーに適切なG64コマンドとP値を配置してパス許容値を設定することをお勧めします。

メトリックの場合：

```
G64 P0.1
```

インペリアルの場合：

```
G64 P0.004
```

3.16.7.4 一時停止したモーション

PlasmaCには、Gコードプログラムが一時停止している間、現在のカットパスに沿ってX軸とY軸を再配置できるようにする機能があります。

この機能を使用するには、LinuxCNCのAdaptive Feed Control（M52）をオンにする必要があります（P1）。

一時停止モーションを有効にするにはGコードのプリアンブルに次の行が含まれている必要があります。

```
M52 P1
```

一時停止モーションをいつでもオフにするには、次のコマンドを使用します。

```
M52 P0
```

3.16.7.5 カット終了時に一時停止

この機能を使用して、アークをトーチ位置に「追いつき」、カットを完全に終了させることができます。 これは通常、より厚い材料に必要であり、ステンレス鋼を切断するときに特に役立ちます。

この機能を使用すると、トーチがまだオンになっている間、カットの最後ですべてのモーションが一時停止します。 実行パネルの[カットパラメータ]セクションの[終了時に一時停止]パラメータで設定された滞留時間（秒単位）が経過すると、PlasmaC は M5 コマンドを続行してオフにし、トーチを上げます。

3.16.7.6 複数のツール

PlasmaC には、カットプログラムの実行時に複数のツールを使用できるようにする機能があります。 使用する前に、PlasmaC で複数のツールを有効にする必要があります。 有効なツールは次のとおりです。

- プラズマトーチ-通常のプラズマ切断に使用されます。
- けがき針-材料の彫刻に使用されます。
- プラズマトーチ-スポッティングに使用されます（穴あけを支援するためのディンプルを作成します）。

複数のツールが有効になっている場合、使用する正しいツールを選択するには、M3 コマンドに LinuxCNC ツール番号（$ n で指定）を含める必要があります。 例：

- M3 $ 0 S1 は、プラズマ切断ツールを選択して開始します。
- M3 $ 1S1 がスクライブを選択して開始します。
- M3 $ 2S1 がプラズマスポッティングツールを選択します。

複数のツール機能を有効にするには、ユーザーは<machine_name> _connections.hal ファイルの次の行を編集する必要があります。

から：

```
setp plasmac.multi-tool 0
```

に：

```
setp plasmac.multi-tool 1
```

スクライブで複数のツールを使用するには、ユーザーが X 軸と Y 軸のオフセットを LinuxCNC ツールテーブルに追加する必要があります。 tool.tbl ファイルは、<machine_name>構成フォルダーにあります。 ツール 0 はプラズマトーチに割り当てられ、ツール 1 はスクライブに割り当てられます。 ツールは TnM6 コマンドで選択され、選択されたツールにオフセットを適用するには G43H0 コマンドが必要です。 LinuxCNC ツールテーブルとツールコマンドは、ユーザーがプラズマトーチに加えてスクライブを使用している場合にのみ機能することに注意することが重要です。 詳細については、スクライブを参照してください。

3.16.7.7 速度低下

Motion.analog-out-03 という名前の HAL ピンがあり、M67（モーションと同期）/ M68（即時）コマンドを使用して G コードで変更できます。 このピンは、コマンドで指定されたパーセンテージまで速度を下げます。

Synchronized withMotion と Immediate の違いを完全に理解することが重要です。

- M67（モーションと同期）-指定された出力（たとえば、P2（THC））の実際の変更は、次のモーションコマンドの開始時に発生します。 後続のモーションコマンドがない場合、出力の変更は発生しません。 M67 の直後にモーションコード（たとえば、G0 または G1）をプログラムすることをお勧めします。
- M68（即時）-これらのコマンドは、モーションコントローラによって受信されるとすぐに実行されます。 これらはモーションと同期していないため、ブレンドが壊れます。 つまり、これらのコードがアクティブなモーションコードの途中で使用されている場合、モーションは一時停止してこれらのコマンドをアクティブにします。

例：

- M67 E3 Q0 は、速度を CutFeedRate の 100%に設定します。
- M67 E3 Q40 は、速度を CutFeedRate の 40%に設定します。
- M67 E3 Q60 は、速度を CutFeedRate の 60%に設定します。
- M67 E3 Q100 は、速度を CutFeedRate の 100%に設定します。

許可される最小パーセンテージは 10%で、これより低い値は 10%に設定されます。

許可される最大パーセンテージは 100%で、これを超える値は 100%に設定されます。

ユーザーがこの機能を使用する場合は、G コードプログラムのプリアンブルとポストアンブルの両方に M68 E3 Q0 を追加して、マシンが既知の状態で開始および終了するようにするのが賢明です。

ヒント

同じ結果を達成する別の方法は、乗数とともに F＃<_ hal [plasmac.cut-feed-rate]を使用することです。

例えば：

```
F[#<_hal[plasmac.cut-feed-rate] * 0.6] 重要
```

重要

G コード THC および速度ベースの THC は、カッター補正が有効な場合は使用できません。 エラーメッセージが表示されます。

警告

実行パネルのカットフィードレートがゼロに設定されている場合、PlasmaC は THC 計算に motion.requested-velocity（G コードの標準フィードレート呼び出しで設定）を使用します。 これは、速度ベースの THC を実装する信頼できる方法ではないため、お勧めしません。

Note

CutFeedRate へのすべての参照は、実行パネルに表示されるカットフィードレート値を参照します。

3.16.7.8 マテリアルハンドリング

マテリアルハンドリングは、コンフィギュレーターの実行時にマシン構成用に作成されたマテリアルファイルを使用し、ユーザーが既知のマテリアル設定を便利に保存して、手動または G コードを介して自動的に呼び出すことができるようにします。 結果のマテリアルファイルの名前は<machine_name> _material.cfg です。

PlasmaC では、マテリアルファイルを使用する必要はありません。 代わりに、ユーザーは実行パネルから手動でカットパラメータを変更できます。 また、自動マテリアル変更を使用する必要はありません。 ユーザーがこの機能を使用したくない場合は、G コードファイルから材料変更コードを省略できます。

材料ファイル内の材料番号は、連続している必要はなく、番号順になっている必要もありません。

次の変数は必須であり、マテリアルファイルのロード時にエラーメッセージが見つからない場合はエラーメッセージが表示されます。

- PIERCE_HEIGHT
- PIERCE_DELAY
- CUT_HEIGHT
- CUT_SPEED

次の変数はオプションです。 それらが検出されないか、値が割り当てられていない場合、値 0 が割り当てられ、エラーメッセージは表示されません。

- NAME
- KERF_WIDTH
- THC
- PUDDLE_JUMP_HEIGHT
- PUDDLE_JUMP_DELAY
- CUT_AMPS
- CUT_VOLTS
- PAUSE_AT_END

- GAS_PRESSURE
- CUT_MODE

> 警告
>
> 変数が G コードを実行するための要件である場合、変数が含まれていることを確認するのはオペレーターの責任です。

マテリアルファイルは次の形式を使用します。

```
[MATERIAL_NUMBER_1]
NAME = name
KERF_WIDTH = value
THC = value (0 = off, 1 = on)
PIERCE_HEIGHT = value
PIERCE_DELAY = value
PUDDLE_JUMP_HEIGHT = value
PUDDLE_JUMP_DELAY = value
CUT_HEIGHT = value
CUT_SPEED = value
CUT_AMPS = value (for info only unless PowerMax communications is enabled)
CUT_VOLTS = value (modes 0 & 1 only, if not using auto voltage sampling)
PAUSE_AT_END = value
GAS_PRESSURE = value (only used for PowerMax communications)
CUT_MODE = value (only used for PowerMax communications)
```

実行パネルから、新しいマテリアルを追加したり、マテリアルを削除したり、既存のマテリアルを編集したりすることができます。 g コードファイルで魔法のコメントを使用してこれを実現することも可能です。

LinuxCNC の実行中に、テキストエディタを使用してマテリアルファイルを編集できます。 変更を保存したら、実行パネルで[再読み込み]を押して、材料ファイルを再読み込みします。

手動のマテリアルハンドリングの場合、ユーザーは G コードプログラムを開始する前に、実行パネルのマテリアルリストからマテリアルを手動で選択します。 実行パネルで材料リストを使用して材料を選択することに加えて、ユーザーは次のコマンドで MDI を使用して材料を変更できます。

```
M190 Pn
```

次のコードは、手動の材料選択方法を使用してカットを成功させるために必要な最小限のコードです。

```
F#<_hal[plasmac.cut-feed-rate]>
M3 $0 S1
.
```

```
.
M5 $0
```

Note

手動のマテリアルハンドリングでは、ユーザーはジョブ全体で 1 つのマテリアルのみに制限されます。

自動マテリアルハンドリングの場合、ユーザーは G コードファイルにコマンドを追加して、PlasmaC がマテリアルを自動的に変更できるようにします。

次のコードを使用して、PlasmaC が材料を自動的に変更できるようにすることができます。

- M190Pn-現在表示されている品目を品目番号 n に変更します。
- M66 P3 L3 Q1-PlasmaC が材料を正常に変更したことを確認するのを待つために、わずかな遅延（この例では 1 秒）を追加します。
- F#<_ hal [plasmac.cut-feed-rate]>-カット送り速度を、実行パネルの[カットパラメータ]セクションに表示されている送り速度に設定します。

自動マテリアルハンドリングの場合、コードは示されている順序で適用する必要があります。 1 つまたは複数の材料変更コマンドを含む G コードプログラムがロードされている場合、プログラムのロード中に最初の材料が実行パネルに表示されます。 次のコードは、自動材料選択方法を使用してカットを成功させるために必要な最小限のコードです。

```
M190 Pn
M66 P3 L3 Q1
F#<_hal[plasmac.cut-feed-rate]>
M3 $0 S1
.
.
M5 $0
```

このアプリケーションは、既存のツールテーブルを PlasmaC マテリアルファイルに変換するために使用されます。 また、手動のユーザー入力から入力フィールドへの材料ファイルを作成することもできます。

この段階で利用できる変換は、SheetCam と Fusion360 のツールテーブルのみです。

SheetCam ツールテーブルは完全であり、変換は完全に自動化されています。 SheetCam ツールファイルは、SheetCam.tools 形式である必要があります。

Fusion360 ツールテーブルには必須フィールドのすべてが含まれていないため、ユーザーは不足しているパラメーターの入力を求められます。 Fusion360 ツールファイルは Fusion360.json 形式である必要があります。

ユーザーが変換したい別の CAM ソフトウェアのフォーマットを使用している場合は、LinuxCNC フォーラムの PlasmaC フォーラムセクションで新しいトピックを作成して、この追加をリクエストしてください。

Materialverter は、ユーザーの構成ディレクトリで materialverter.py をダブルクリックして GUI ファイルマネージャから実行するか、ターミナルウィンドウに次のコマンドを入力して実行できます。

```
python～ / linuxcnc / configs / <machine_name> /materialverter.py
```

これにより、Materialverter ダイアログが表示されます。

次のいずれかを選択します。

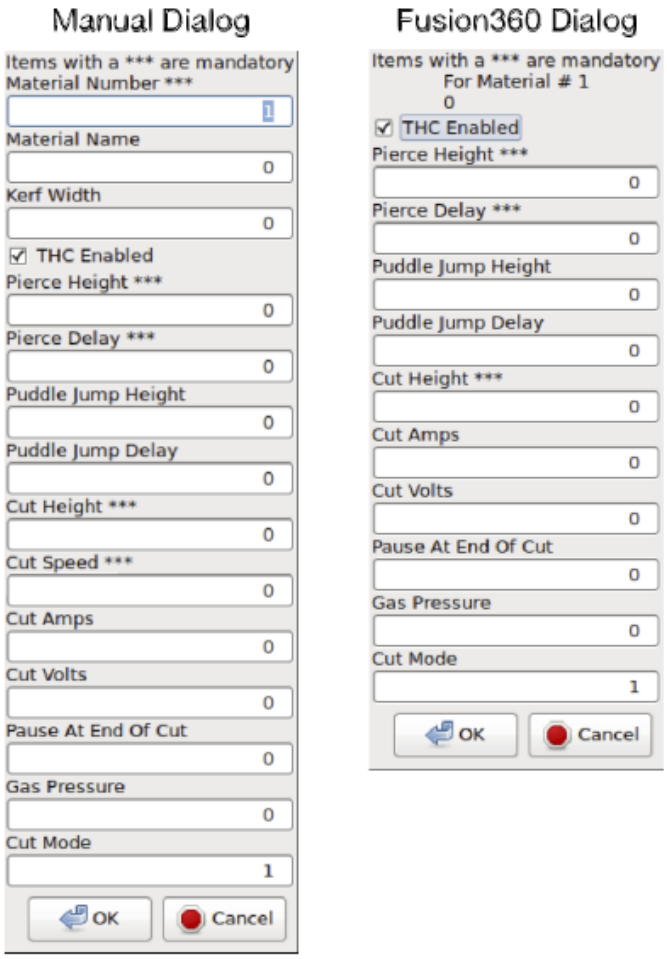
- 手動-新しいマテリアルファイルを手動で作成します。
- SheetCam-SheetCam ツールファイルを変換します。
- Fusion360-Fusion360 ツールファイルを変換します。

SheetCam の場合のみ、ユーザーがメトリックまたはインペリアル出力ファイルを必要とするかどうかを選択します。

変換する：

1. 変換する入力ファイルを選択します。
2. 書き込む出力ファイルを選択します。 これは通常、～/ linuxcnc / configs / <machine_name> _material.cfg になります。 必要に応じて、ユーザーは別のファイルを選択し、<machine_name> _material.cfg ファイルを手動で編集できます。
3. [作成/変換]をクリックすると、新しいマテリアルファイルが作成されます。

手動作成または Fusion360 変換の両方で、ダイアログが表示され、入力用に使用可能なすべてのパラメーターが表示されます。 ***でマークされたエントリは必須であり、他のすべてのエントリはユーザーの構成ニーズに応じてオプションです。



Note

ユーザーが〜/ linuxcnc / configs / <machine_name> _material.cfg を選択し、ファイルがすでに存在する場合、ファイルは上書きされます。

Gコードファイルの「マジックコメント」を使用して、新しい素材を追加したり、既存の素材を編集したりすることができます。 フォーマットルールは次のとおりです。

- コメント全体を括弧で囲む必要があります。
- マジックコメントの先頭は次のようにする必要があります：（o =
- 等号は、スペースを入れずに各パラメーターの直後に続ける必要があります。
- 必須パラメーターはマジックコメントに含まれている必要があります（オプション0にはnuとnaは必要ありません）。
- オプション0を除いて、Gコードファイルには任意の数とタイプのマジックコメントを含めることができます。Gコードファイルに複数のオプション0コメントが見つかった場合は、最後に見つかったコメントのみが使用されます。
- オプション1および/またはオプション2に加えてオプション0を使用する場合は、オプション0のコメントをGコードファイルの最後のコメントにする必要があります。

オプションは次のとおりです。

オプション	内容
0	一時的なデフォルトのマテリアルを作成します。 このオプションで追加された材料情報は、手動のリロードまたは LinuxCNC の再起動によって破棄されます。
1	指定された数が存在しない場合は、新しい材料を追加します。
2	指定された数が存在する場合、既存のマテリアルを上書きします。指定された数が存在しない場合は、新しい材料を追加します。

必須パラメーターは次のとおりです。

名前	内容
o	使用するオプションを選択します。
nu	材料番号を設定します（オプション 0 には必要ありません）。
na	材料名を設定します（オプション 0 には必要ありません）。
ph	ピアスの高さを設定します。
pd	ピアス遅延を設定します。
ch	カットの高さを設定します。
fr	送り速度を設定します。

オプションのパラメータは次のとおりです。

名前	内容
kw	切り口の幅を設定します。
th	THC ステータスを設定します（0 =無効、1 =有効）。
ca	カットアンペアを設定します。
cv	カット電圧を設定します。
pe	終了遅延での一時停止を設定します。
gp	ガス圧（PowerMax）を設定します。
cm	カットモード（PowerMax）を設定します。
jh	水たまりのジャンプの高さを設定します。

jd	水たまりのジャンプ遅延を設定します。

完全な例：

```
(o=1, nu=2, na=5mm Mild Steel 40A, ph=3.1, pd=0.1, ch=0.75, fr=3000, kw=0.5, th=1, ca=45, -
cv=110, pe=0.1, gp=5, cm=1, jh=0, jd=0)
```

3.16.7.9 THC（トーチ高さコントローラー）

THC は、実行パネルの THC フレームから制御できます。

実行パネルの THC セクションで THC が無効になっていない場合は、G コードプログラムから直接 THC を有効または無効にすることもできます。

PlasmaC は、実行パネルの[自動ボルトを使用]チェックボックスの状態に依存する制御電圧を使用します。

1. Use Auto Volts がチェックされている場合、実際のカット電圧はカット開始後にサンプリングされます。アーク電圧を安定させるために、PlasmaC は、電圧をサンプリングする前に、構成パネルの THC セクションの[遅延]フィールドに表示される時間待機します。 サンプリングされた電圧は、トーチの高さを調整するためのターゲット電圧として使用されます。
2. [自動ボルトを使用]がチェックされていない場合、実行パネルの[カットパラメータ]セクションに[カットボルト]として表示される電圧が、トーチの高さを調整するためのターゲット電圧として使用されます。

THC は、速度が CutFeedRate の 99.9%に達するまでアクティブになりません。

G コード THC

THC は、実行パネルで THC が無効になっていない場合、M コード M62-M65 で motion.digital-out-02 ピンを設定またはリセットすることにより、G コードから直接無効または有効にすることができます。

- M62 P2 は THC（モーションと同期）を無効にします
- M63 P2 は THC（モーションと同期）を有効にします
- M64 P2 は THC を無効にします（すぐに）
- M65 P2 は THC を有効にします（すぐに）

Synchronized withMotion と Immediate の違いを完全に理解することが重要です。

- M62 および M63（モーションと同期）-指定された出力（たとえば、P2（THC））の実際の変更は、次のモーションコマンドの開始時に発生します。 後続のモーションコマンドがない場合、出力の変更は発生しません。 M62 または M63 の直後にモーションコード（たとえば、G0 または G1）をプログラムすることをお勧めします。
- M64 および M65（即時）-これらのコマンドは、モーションコントローラによって受信されるとすぐに実行されます。 これらはモーションと同期していないため、ブレンドが壊れます。 つまり、これらのコードがアク

ティブなモーションコードの途中で使用されている場合、モーションは一時停止してこれらのコマンドをアクティブにします。

速度ベースの THC

カット速度が CutFeedRate のパーセンテージ（構成パネルの THC フレームの VAD しきい値%値で定義）を下回ると、カット速度が CutFeedRate の少なくとも 99.9%に戻るまで THC がロックされます。 これは、モニターパネルの THC ベロシティロックインジケーターが点灯することで明らかになります。

速度ベースの THC は、鋭い角や小さな穴で速度が低下したときにトーチの高さが変化するのを防ぎます。

ベロシティリダクションは、ベロシティベースの THC に次のように影響することに注意することが重要です。

1. カットの途中でベロシティリダクションが呼び出されると、THC がロックされます。
2. THC は、VAD しきい値を超える値に戻すことによって速度低下がキャンセルされるまでロックされたままになり、トーチは実際に CutFeedRate の 99.9%に達します。

3.16.7.10 カッター補正

LinuxCNC（PlasmaC）には、選択したマテリアルのカットパラメータのカーフ幅で指定された量だけ、現在のプログラムのカットパスを自動的に調整する機能があります。 これは、G コードが公称カットパスにプログラムされており、ユーザーがさまざまな厚さの材料でプログラムを実行して、一貫したサイズの部品を確保するのに役立つ場合に役立ちます。

カッター補正を使用するには、ユーザーは切り口幅の HAL ピンで G41.1、G42.1、および G40 を使用する必要があります。

- G41.1 D＃<_ hal [plasmac_run.kerf-width-f]>：プログラムされたパスの左側にトーチをオフセットします
- G42.1 D＃<_ hal [plasmac_run.kerf-width-f]>：プログラムされたパスの右側にトーチをオフセットします
- G40 はカッター補正をオフにします

> **重要**
>
> カッター補正が有効な G コード THC である場合、速度ベースの THC およびオーバーカットは使用できません。 エラーメッセージが表示されます。

3.16.7.11 初期高さ検知（IHS）スキップ

IHS は、次の 2 つの方法のいずれかでスキップできます。

1. THC が無効になっている場合、カットの開始が最後に成功したプローブからのスキップ IHS 距離よりも小さい場合、IHS スキップが発生します。
2. THC が有効になっている場合、カットの開始が最後のカットの終了から IHS をスキップする距離よりも小さいと、IHS スキップが発生します。

Skip IHS の値がゼロの場合、IHS スキップは無効になります。

カット中にエラーが発生すると、スキップ IHS が有効になっている場合、次のカットの IHS スキップが無効になります。

3.16.7.12 プロービング

プロービングは、オームセンシングまたはフロートスイッチのいずれかを使用して実行できます。 2つの方法を組み合わせることも可能です。その場合、フロートスイッチはオーミックプロービングへのフォールバックを提供します。

マシンのトーチがオーミックプロービングをサポートしていない場合、ユーザーはトーチの横に別のプローブを置くことができます。 この場合、ユーザーはトーチの下にプローブを伸ばします。 プローブはトーチの下の最小カット高さを超えて伸びてはならず、Z 軸オフセット距離は構成パネルにオームプローブオフセットとして入力する必要があります。

プロービングのセットアップは、構成パネルのプロービングフレームで行われます。 プロービング速度は、構成パネルのモーションフレームで制御されます。

PlasmaC は、機械がフロートスイッチ内でオーバーランを吸収するのに十分な動きをしている限り、Z 軸の全速度でプローブできます。 マシンのフロートスイッチの移動が適切な場合、ユーザーはプローブの高さを Z 軸の MINIMUM_LIMIT の近くに設定し、すべてのプローブを全速力で行うことができます。

一部のフロートスイッチは、最終的なプローブアップを完了するための過剰な時間としてプローブシーケンスに現れる大きなスイッチングヒステリシスを示す可能性があります。

- この時間は、ファイナルプローブの速度を上げることで短縮できます。
- この速度のデフォルトは、サーボサイクルあたり 0.001mm（0.000039 "）です。
- <machine_name> _connections.hal ファイルに次の行を追加することで、この速度を最大 10 倍に上げることができます。

```
setp plasmac.probe-final-speed n
```

ここで、n は 1〜10 の値です。 この値をできるだけ低く保つことをお勧めします。

この機能を使用すると、最終的な高さがわずかに変化し、最終的な高さを確認するために徹底的なプローブテストが必要になります。

この速度値はすべてのプロービングに影響するため、ユーザーがオーミックプロービングを使用し、ユーザーがこの速度値を変更した場合、ユーザーはこの速度変更とフロート移動を補正するために必要なオフセットを設定するためにプローブテストを行う必要があります。

この機能の信頼性は、フロートスイッチの再現性と同程度になります。

Note

プローブの高さは、Z 軸の MINIMUM_LIMIT より上の高さを指します。

3.16.7.13 カットタイプ

PlasmaC では、2 つの異なるカットモードが可能です。

1. ピアスとカット-ロードされた G コードプログラムを実行して、ピアスしてからカットします。
2. ピアスのみ-ロードされた G コードプログラムを変換してから、プログラムを実行して、M3S1 または M3 $ 0S1 コマンド位置でのみ材料をピアスします。 Scribe（M3 $ 1 S1）および Spotting（M3 $ 2 S1）コマンドは無視され、これらの場所で貫通は行われません。

ピアスオンリーモードは、切断中にトーチと干渉するためにピアスから材料表面に十分なドロスを生成する可能性がある厚い材料に役立ちます。 これにより、カットする前にシート全体に穴を開けてきれいにすることができます。

これにより、寿命に近い消耗品をピアシングに使用し、切断中に使用できる優れた消耗品と交換することもできます。

この機能を有効にする方法は 2 つあります。

1. デフォルトのカスタムユーザーボタンを使用して、カットタイプを切り替えます。
2. 最初のカットの前に G コードプログラムに次の行を追加して、現在のファイルのピアスのみモードを有効にします。

```
#<pierce-only> = 1
```

カスタムユーザーボタンを使用している場合、カットタイプが切り替えられると、PlasmaC はファイルを自動的にリロードします。

3.16.7.14 穴あけ-イントロ

カットする穴の直径は、カットする材料の厚さの 1.5 倍以上にすることをお勧めします。

また、直径 32mm（1.26 "）未満の穴は、プロファイルカットに使用される送り速度の 60％でカットすることをお勧めします。これにより、速度の制約により THC もロックアウトされます。

PlasmaC は、通常 CAM ポストプロセッサ（PP）によって設定される G コードコマンドを利用して穴あけを支援できます。ユーザーが PP を持っていない場合、またはユーザーの PP がこれらの方法をサポートしていない場合、PlasmaC は G コードを自動的に適応させることができます。 スーツ。 この自動モードはデフォルトで無効になっています。

小さな穴の品質を向上させるために利用できる 3 つの方法があります。

1. 速度の低下-速度を CutFeedRate の約 60％に低下させます。

2. アークドウェル（終了時に一時停止）-動きが停止している間、穴の端でトーチを短時間オンにして、アークが追いつくようにします。
3. オーバーカット-穴の端でトーチをオフにしてから、パスに沿って続行します。

Note

Arc Dwell と Overcut の両方が同時にアクティブな場合、Overcut が優先されます。

重要

カッター補正が有効な場合、オーバーカットを使用することはできません。 エラーメッセージが表示されます。

3.16.7.15 穴あけ

G コードコマンドは、CAM ポストプロセッサ（PP）または手動コーディングのいずれかによって設定できます。

穴あけ速度の低減

穴を開けるのに速度を落とす必要がある場合、ユーザーは次のコマンドを使用して速度を設定します。M67E3 Qnn ここで、nn は必要な速度のパーセンテージです。 たとえば、M67 E3 Q60 は、速度を現在のマテリアルの CutFeedRate の 60%に設定します。

速度ベースの THC セクションを参照してください。

サンプルコード：

```
G21 (metric)
G64 P0.005
M52 P1 (allow paused motion)
F#<_hal[plasmac.cut-feed-rate]> (feed rate from cut parameters)
G0 X10 Y10
M3 $0 S1 (start cut)
G1 X0
M67 E3 Q60 (reduce feed rate to 60%)
G3 I10 (the hole)
M67 E3 Q0 (restore feed rate to 100%)
M5 $0 (end cut)
G0 X0 Y0
M2 (end job)
```

アークドウェル（終了時に一時停止）

このメソッドは、実行パネルの[カットパラメーター]フレームで[終了時に一時停止]パラメーターを設定することで呼び出すことができます。

オーバーカット

トーチは、M コード M62（モーションと同期）または M64（即時）で motion.digital-out-03 ピンを設定することにより、穴の端でオフにすることができます。 トーチをオフにした後、motion.digital-out-03 ピンを M コード M63 または M65 でリセットして次のカットを開始する前に、トーチを再びオンにする必要があります。これは、PlasmaC によって自動的に行われます。 M63P3 または M65P3 が表示されずに M5 コマンドに到達した場合の G コードパーサー。

トーチをオフにすると、デフォルトの長さ 4mm（0.157 "）で穴のパスがたどられます。この距離は、G コードファイルに #<oclength> = n を追加することで指定できます。

- M62 P3 はトーチをオフにします（モーションと同期）
- M63 P3 を使用すると、トーチをオンにすることができます（モーションと同期）
- M64 P3 はトーチをオフにします（すぐに）
- M65 P3 を使用すると、トーチをオンにすることができます（すぐに）

Synchronized withmotion と Immediate の違いを完全に理解することが重要です。

- M62 および M63（モーションと同期）-指定された出力（たとえば、P2（THC））の実際の変更は、次のモーションコマンドの開始時に発生します。 後続のモーションコマンドがない場合、出力の変更は発生しません。 M62 または M63 の直後にモーションコード（たとえば、G0 または G1）をプログラムすることをお勧めします。
- M64 および M65（即時）-これらのコマンドは、モーションコントローラによって受信されるとすぐに実行されます。 これらはモーションと同期していないため、ブレンドが壊れます。 つまり、これらのコードがアクティブなモーションコードの途中で使用されている場合、モーションは一時停止してこれらのコマンドをアクティブにします。

サンプルコード：

```
G21 (metric)
G64 P0.005
M52 P1 (allow paused motion)
F#<_hal[plasmac.cut-feed-rate]> (feed rate from cut parameters)
G0 X10 Y10
M3 $0 S1 (start cut)
G1 X0
M67 E3 Q60 (reduce feed rate to 60%)
G3 I10 (the hole)
```

```
M62 P3 (turn torch off)
G3 X0.8 Y6.081 I10 (continue motion for 4mm)
M63 P3 (allow torch to be turned on)
M67 E3 Q0 (restore feed rate to 100%)
M5 $0 (end cut)
G0 X0 Y0
M2 (end job)
```

3.16.7.16 穴あけ-自動

PlasmaC には、G コードを自動的に変更して速度を下げたり、オーバーカットを適用したりする機能があります。これは、穴を開けるときに役立ちます。

PlasmaC ホールセンシングのデフォルトのホールサイズは 32mm（1.26 "）です。この値は、G コードファイルで次のコマンドを使用して変更できます。

- ＃<h_diameter> = nn-G コードファイルの残りの部分と同じ単位系で直径（nn）を設定します。

PlasmaC の小さな穴のデフォルトの速度は、現在の送り速度の 60%です。 この値は、G コードファイルで次のコマンドを使用して変更できます。

- ＃<h_velocity> = nn-必要な現在の送り速度のパーセンテージ（nn）を設定します。

PlasmaC ホールセンシングはデフォルトで無効になっています。 次の G コードパラメータを使用して目的の穴検知モードを選択することにより、有効/無効にできます。

- ＃<holes> = 0-以前に有効にされていた場合、PlasmaC はホール検知を無効にします。
- ＃<holes> = 1-PlasmaC が 32mm（1.26 "）未満の穴の速度を CutFeedRate の 60%に低下させます。
- ＃<holes> = 2-設定 1 での速度の変化に加えて、PlasmaC が穴をオーバーカットします。
- ＃<holes> = 3-PlasmaC が 32mm（1.26 "）未満の穴と 16mm（0.63"）未満の円弧の速度を CutFeedRate の 60%に低下させます。
- ＃<holes> = 4-設定 3 での速度変更に加えて、PlasmaC が穴をオーバーカットします。

アークドウェル（終了時に一時停止）

この機能は、実行パネルの[カットパラメータ]フレームで[終了時に一時停止]パラメータを設定することにより、適切な G コードパラメータを介して目的の穴検知モードを設定することに加えて使用できます。

オーバーカット

穴検知モード 2 または 4 がアクティブな場合、PlasmaC は、モード 1 および 3 に関連する速度の変化に加えて、穴をオーバーカットします。

PlasmaC ホールセンシングのデフォルトのオーバーカット長は 4mm（0.157 "）です。この値は、G コードファイルで次のコマンドを使用して変更できます。

- ＃<oclength> = nn は、G コードファイルの残りの部分と同じ単位系でオーバーカット長（nn）を指定します。

サンプルコード：

```
G21 (metric)
G64 P0.005
M52 P1 (allow paused motion)
F#<_hal[plasmac.cut-feed-rate]> (feed rate from cut parameters)
#<holes> = 2 (overcut for holes)
#<oclength> = 6.5 (optional, 6.5mm overcut length)
G0 X10 Y10
M3 $0 S1 (start cut)
G1 X0
G3 I10 (the hole)
M5 $0 (end cut)
G0 X0 Y0
M2 (end job)
```

Note

1 つの G コードファイルに複数の混合穴コマンドを含めることは問題ありません。

3.16.7.17 シングルカット

シングルカットは、G コードプログラムを実行する前にシートを小さな断片にカットするためによく使用される単一の一方向のカット移動です。

シングルカットを開始する前に、マシンをホームに戻す必要があります。

シングルカットは、マシンの現在の X / Y 位置から開始されます。

自動シングルカット

これが推奨される方法です。 このメソッドのパラメータは、実行パネルに入力されます。

1. 必要な X / Y 開始位置にジョグします。
2. 実行パネルのカットパラメータで必要なカット送り速度を設定します。
3. X 軸および/または Y 軸に沿ったカットの長さを入力します。
4. カット開始ボタンを押すとカットが始まります。

ペンダントシングルカット

スピンドルを始動および停止し、X 軸と Y 軸をジョグできるペンダントが機械に装備されている場合、ユーザーは手動でシングルカットを実行できます。

1. 必要な X / Y 開始位置にジョグします。
2. ジョグ速度スライダーで必要な送り速度を設定します。
3. スピンドルを始動して、カットプロセスを開始します。
4. トーチをプローブした後、発火します。
5. Arc OK を受信すると、ジョグボタンを使用してカットラインに沿ってマシンをジョギングできます。
6. カットが完了したら、スピンドルを停止します。
7. トーチがオフになり、Z 軸が開始位置に戻ります。

軸の手動シングルカット

1. 必要な X / Y 開始位置にジョグします。
2. ジョグ速度スライダーで必要な送り速度を設定します。
3. F9 を押して手順を開始します。
4. トーチをプローブした後、発火します。
5. Arc OK を受信すると、ジョグキーを使用してカットラインに沿ってマシンをジョギングできます。
6. カットが完了したら、F9 または Esc を押します。
7. トーチがオフになり、Z 軸が開始位置に戻ります。

Note

切断中にトーチが炎上した場合でも、ユーザーは F9 または Esc を押して切断を終了する必要があります。これにより、Z オフセットがクリアされ、トーチが開始位置に戻ります。

Gmoccapy の手動シングルカット

ユーザーがキーボードジョギングキーを使用する場合は、[設定]ページの[ハードウェア]タブでキーボードショートカットを有効にします。それ以外の場合は、GUI ジョグボタンを使用する必要があります。

1. 必要な X / Y 開始位置にジョグします。
2. ジョグ速度スライダーで必要な送り速度を設定します。
3. MDI モードに入ります。
4. M3S1 と入力して手順を開始します。
5. 手動モードに入ります。
6. トーチをプローブした後、発火します。
7. Arc OK を受信すると、GUI ジョグボタン（またはキーボードショートカットが有効になっている場合はジョグキー）を使用して、カットラインに沿ってマシンをジョグできます。
8. カットが完了したら、Esc キーを押します。

9. トーチがオフになり、Z 軸が開始位置に戻ります。

Note

切断中にトーチが炎上した場合でも、ユーザーは F9 または Esc を押して切断を終了する必要があります。これにより、Z オフセットがクリアされ、トーチが開始位置に戻ります。

3.16.7.18 メッシュモード（エキスパンドメタルカッティング）

PlasmaC は、マシンにパイロットアークトーチがあり、コンスタントパイロットアーク（CPA）モードが可能な場合、エキスパンド（メッシュ）金属を切断できます。

メッシュモードは THC を無効にし、カット中に失われた ArcOK 信号も無視します。 実行パネルの[切断モード]セクションにある[メッシュモード]チェックボタンをオンにすることで選択できます。

マシンで HyperthermPowerMax プラズマカッターで RS485 通信が有効になっている場合、メッシュモードを選択すると、実行パネルの[カットパラメーター]セクションで[カットモード]スピンボタンが無効になり、カットモード 2（CPA）が選択されます。 メッシュモードを無効にすると、カットモードスピンボタンが再度有効になり、元のカットモードが復元されます。

実行パネルの[切断モード]セクションにある[アーク OK を無視]チェックボタンをオンにすることで、アーク OK 信号を受信せずにメッシュモードカットを開始することもできます。

メッシュモードと IgnoreArc OK はどちらも、ジョブ中いつでも有効/無効にできます。

3.16.7.19 ArcOK を無視する

Ignore Arc OK モードは、THC を無効にし、Arc OK 信号を必要とせずにカットを開始し、カット中に失われた ArcOK 信号を無視します。

このモードは、次の方法で選択できます。

1. 実行パネルの[切断モード]セクションにある[アーク OK を無視]チェックボタンをオンにします。
2. G コードを介して HAL ピン motion.digital-out-01 を 1 に設定します。

- M62 P1 は、Ignore Arc OK（Synchronized with Motion）を有効にします
- M63 P1 は、Ignore Arc OK（Synchronized with Motion）を無効にします
- M64 P1 は、Ignore Arc OK を有効にします（すぐに）
- M65 P1 は Ignore Arc OK を無効にします（すぐに）

Synchronized withmotion と Immediate の違いを完全に理解することが重要です。

- M62 および M63（モーションと同期）-指定された出力（たとえば、P2（THC））の実際の変更は、次のモーションコマンドの開始時に発生します。 後続のモーションコマンドがない場合、出力の変更は発生しま

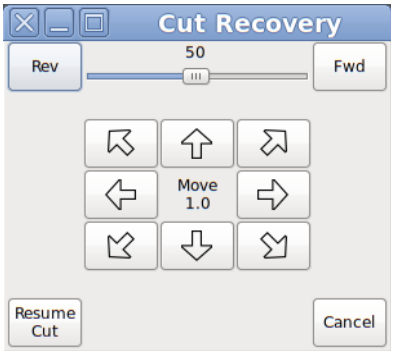
せん。 M62 または M63 の直後にモーションコード（たとえば、G0 または G1）をプログラムすることをお勧めします。

- M64 および M65（即時）-これらのコマンドは、モーションコントローラによって受信されるとすぐに実行されます。 これらはモーションと同期していないため、ブレンドが壊れます。 つまり、これらのコードがアクティブなモーションコードの途中で使用されている場合、モーションは一時停止してこれらのコマンドをアクティブにします。

このモードは、ユーザーがカットを開始するために Arc OK 信号を必要としない場合は、メッシュモードと組み合わせて使用することもできます。

メッシュモードと IgnoreArc OK はどちらも、ジョブ中いつでも有効/無効にできます。

3.16.7.20 カットリカバリー



この機能は、一時停止されたモーションイベント中にトーチをカットパスから離して、カットされる材料のスクラップ部分の上にトーチを配置して、カットが最小化されたアークで再開できるようにするダイアログボックスを生成します。 ディボット。 この機能は、Extras パネルのユーザープログラムボタン（デフォルトでは Cut-Recovery という名前）からのみ使用でき、モーションが一時停止しているときにのみクリックできます。

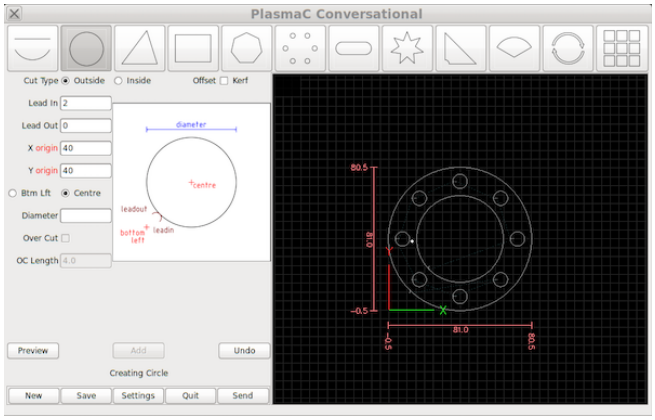
一時停止モーションが発生したポイントからトーチ位置を調整することが望ましいですが、新しい開始ポイントを設定する前にカットパスに沿って移動する必要がある場合、ユーザーは一時停止モーションコントロール（Rev、Fwd、およびジョグ）を使用できます。 -スピードスライダー）ダイアログボックスの上部にあります。 ユーザーがカットパスに沿ったトーチの配置に満足したら、方向ボタンを押すことでカットパスから離れることができます。 方向ボタンを押すたびに、現在選択されているマテリアルの KerfWidth パラメータに相当する距離だけトーチが移動します。

トーチがカットパスから外れると、ダイアログボックスの上部にある一時停止したモーションコントロール（Rev、Fwd、およびジョグスピードスライダー）が無効になります。

トーチの位置が適切になったら、[カットの再開]または GUI の[再開]ボタン（一時停止記号）を押すと、カットが新しい位置から再開され、元の一時停止したモーション位置まで最短距離で移動します。 トーチが元の一時停止したモーション位置に戻ると、ダイアログボックスが閉じます。

[キャンセル]（またはタイトルバーの X）を押すと、トーチがモーションが一時停止したときの位置に戻り、ダイアログボックスが閉じます。

3.16.7.21 会話型ライブラリ



Conversational Shape Library は、いくつかの基本的な形状と機能で構成されており、ユーザーがマシンですばやく GCode を生成して、単純な形状をすばやくカットできるようにします。 この機能は、[エクストラ]タブにある[会話]ボタンをクリックすると開始されます。

> **Note**
>
> 会話型ライブラリは、達成できるものに制限があるため、CAD / CAM の代替となることを意図したものではありません。

形状入力ボックスの空白のエントリは、G コードが生成されたときの現在の設定を使用します。 たとえば、X の開始が空白のままの場合、現在の X 軸の位置が使用されます。

以下を除いて、すべてのリードインとリードアウトはアークです。

- 小さな穴
- 出演者

カット順序は、シェイプが作成されたのと同じ順序で発生します。

パラメータの編集中にキーボードの Return キーを押すと、形状を作成するのに十分なパラメータが入力されている場合、形状のプレビューが自動的に表示されます。 使用可能なチェックボックスのいずれかをクリックすると、同じことが行われます。

ボタンの機能は次のとおりです。

名前	内容
新規	現在の G コードファイルを削除し、空の G コードファイルをロードします。
保存	ダイアログを開き、現在の形状を G コードファイルとして保存できるようにします。

設定	グローバル設定の変更を許可します。
やめる	会話パネルを閉じます。 すべての編集は破棄されます。
送信	会話パネルを閉じて、現在の図形を LinuxCNC（PlasmaC）にロードします。 最後の編集が追加されなかった場合、それは破棄されます。
プレビュー	必要な情報が存在する場合、現在の形状のプレビューを表示します。
継続する	このボタンは、線と円弧にのみ使用されます。 別のセグメントを現在のセグメントに追加できるようにします。
追加	現在の図形を現在のジョブに保存します。
元に戻す	以前に保存された状態に戻ります。 最後のストア以降に編集がない場合は、現在の状態のままになります。

会話型パネルを開いたときに LinuxCNC（PlasmaC）にロードされた G コードファイルがある場合、そのコードはジョブの最初の形状として会話型にインポートされます。 このコードが不要な場合は、[新規]ボタンを押して削除できます。

Note

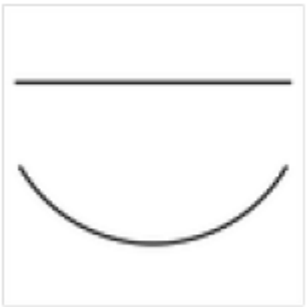
すべての距離は現在のユーザー座標系を基準にしたマシン単位であり、すべての角度は度です。

シェイプライブラリのグローバル設定は、会話型パネルの[設定]ボタンを押すことで設定できます。 これにより、G コードプログラムの作成に使用されるすべての使用可能な設定パラメーターが表示されます。 これらには以下が含まれます：

- Preamble
- Postamble
- Origin (Center or Bottom Left)
- Lead-in length
- Lead-out length
- Small hole diameter
- Small hole speed
- Window Size

小穴の直径よりも小さい直径の内部円は、小穴として分類され、穴の半径または指定された引き込み長さのいずれか短い方の長さの真っ直ぐな引き込みがあります。 また、送り速度は小穴速度に設定されます。

[保存]ボタンを押すと、すべての設定が表示どおりに保存され、ウィンドウサイズが表示されたサイズに変更されます。 保存後、ユーザーは[終了]を押すか、任意の図形をクリックして会話プロセスを開始できます。



線と円弧には、複雑な形状を作成するためにそれらをつなぎ合わせることができるという追加のオプションがあります。 2つの線種と3つの円弧タイプが利用可能です。

1. 始点と終点を指定した線。
2. 始点、長さ、角度を指定した線。
3. 始点、ウェイポイント、および終点が指定されたアーク。
4. 始点、終点、および半径が指定された円弧。
5. 始点、長さ、角度、および半径が指定された円弧。

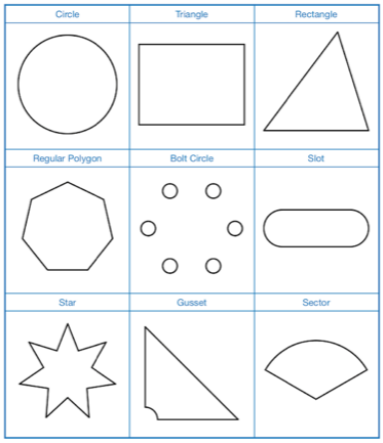
線と円弧を使用するには：

1. [線と円弧]を選択します。
2. 作成する線または円弧のタイプを選択します。
3. 必要なパラメータを入力します。
4. プレビューを押して形状を確認します。
5. 形状に満足したら、続行を押します。
6. 必要に応じて線または円弧のタイプを変更し、形状が完成するまでこの手順を続けます。
7. [送信]を押して、Gコードファイルを LinuxCNC（PlasmaC）に送信してカットします。

ユーザーが閉じた図形を作成する場合は、図形の最初のセグメントとして必要なリードインを作成する必要があります。 リードアウトが必要な場合は、シェイプの最後のセグメントである必要があります。

> Note
>
> この段階では、形状が閉じている場合、リードイン/リードアウトを自動的に作成するオプションはありません。

次の形状を作成できます。



形状を作成するには：

1. 作成する形状を選択します。 使用可能なパラメータが表示されます。
2. 適切な値を入力し、プレビューを押して形状を表示します。
3. 形状が正しくない場合は、値を編集して[プレビュー]を押すと、新しい形状が表示されます。 形に満足するまで繰り返します。
4. [追加]を押して、形状を G コードファイルに追加します。
5. [送信]を押して、G コードファイルを LinuxCNC（PlasmaC）に送信してカットします。

複数の形状を一緒に追加して、複雑なグループを作成できます。

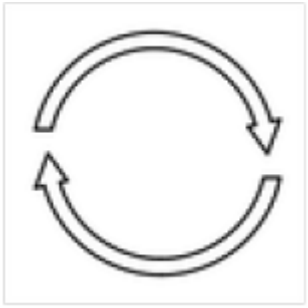
グループのカット順序は、個々の形状がグループに追加される順序によって決まります。

図形がグループに追加されると、編集または削除することはできません。

グループに図形を削除することはできず、追加するだけです。

図形のグループを作成するには：

1. シングルシェイプのように最初のシェイプを作成します。
2. [追加]を押すと、図形がグループに追加されます。
3. ユーザーが同じ形状の別のバージョンを追加したい場合は、必要なパラメーターを編集し、形状に満足したら追加を押します。
4. ユーザーが別の形状を追加したい場合は、その形状を選択して、単一形状のように作成します。
5. グループを完成させるために必要なすべての形状が追加されるまで繰り返します。
6. [送信]を押して、G コードファイルを LinuxCNC（PlasmaC）に送信してカットします。

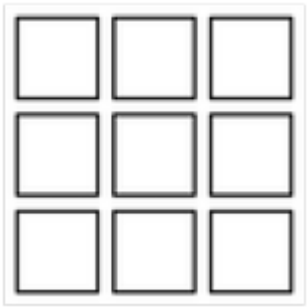


回転機能は、[プレビュー]タブに表示されている現在のコードを回転します。これは、単一の図形、グループ、またはインポートされたGコードファイルにすることができます。

Conversationalを介して作成されたアレイはローテーションできますが、インポートされたアレイはローテーションできません。

数学的に計算されたGコードでは回転は機能しません。値を指定する必要があります。

1. 図形やグループを作成するか、インポートしたGコードファイルを使用します。
2. [回転]を押して、回転ダイアログを開きます。
3. 回転ダイアログに適切な値を入力し、プレビューを押して回転したコードを表示します。
4. ローテーションが正しくない場合は、値を編集して[プレビュー]を押すと、新しいローテーションされたコードが表示されます。 回転したコードに満足するまで繰り返します。
5. [追加]を押して手順を完了します。
6. [送信]を押して、Gコードファイルを LinuxCNC（PlasmaC）に送信してカットします。



配列は、図形、グループ、またはインポートされたGコードファイルから作成できます。

Conversationalで作成された配列は配列できますが、インポートされた配列は配列できません。

配列のカット順序は、左の列から右の列で、一番下の行から始まり、一番上の行で終わります。

1. 図形やグループを作成するか、インポートしたGコードファイルを使用します。
2. アレイを押して、アレイダイアログを開きます。
3. アレイダイアログに適切な値を入力し、プレビューを押してアレイを表示します。

4. 配列が正しくない場合は、値を編集して[プレビュー]を押すと、新しい配列が表示されます。 配列に満足するまで繰り返します。

5. [追加]を押して手順を完了します。

6. [送信]を押して、G コードファイルを LinuxCNC（PlasmaC）に送信してカットします。

プレビューパネルに表示されている現在のジョブは、下部の[保存]ボタンを使用していつでも保存できます。 G コードが LinuxCNC（PlasmaC）に送信され、[会話]ダイアログボックスが閉じている場合でも、ユーザーは GUI から G コードファイルを保存できます。 または、ユーザーは[エクストラ]タブの[会話]ボタンをクリックしてジョブを再読み込みし、そのときに[保存]ボタンを押すこともできます。

3.16.7.22 スクライブ

スクライブは、プラズマトーチに加えて PlasmaC によって操作される場合があります。 この機能は、既存の構成に悪影響を与えないように、デフォルトで無効になっています。 スクライビングを有効にするには、<machine_name> _connections.hal ファイルを編集する際のガイダンスについて、「複数のツール」セクションを参照してください。

スクライブを使用するには、LinuxCNC ツールテーブルを使用する必要があります。 ツール 0 はプラズマトーチに割り当てられ、ツール 1 はスクライブに割り当てられます。 プラズマトーチからのスクライブ X 軸と Y 軸のオフセットは、LinuxCNC ツールテーブルに入力する必要があります。 これは、メイン GUI を介してツールテーブルを編集するか、<machine_name>構成ディレクトリの tool.tbl ファイルを編集することによって行われます。 これは、スクライブがワークピースに移動して適切なオフセットを決定できるようになった後に実行されます。

X と Y のプラズマトーチオフセットは常にゼロになります。 ツールは、オフセットを適用するために必要な TnM6 コマンドとそれに続く G43H0 コマンドによって選択されます。 次に、ツールは M3 $ nS1 コマンドで起動されます。 n の場合、トーチ切断には 0 を使用し、スクライビングには 1 を使用します。

ユーザーが Configurator のスクライブに HAL ピンをまだ割り当てていない場合は、Configurator の再構成モードを呼び出して割り当てる必要があります。

スクライブを操作するために使用される 2 つの HAL 出力ピンがあります。最初のピンは、スクライブを材料の表面に移動させるスクライブを準備するために使用されます。 アーム遅延が経過した後、2 番目のピンを使用してスクライブを開始します。 オンディレイが経過すると、モーションが始まります。

スクライブを有効にした後で PlasmaC を使用するには、LinuxCNC ツールとして各 G コードファイルでトーチまたはスクライブのいずれかを選択する必要があります。

次のステップは、GUI でツールテーブルエディタを開くか、<machine_name>構成ディレクトリで tool.tbl を編集して、X 軸と Y 軸のオフセットを入力することです。 これらは、トーチノズルの中心から測定した、スクライブがトーチから離れている X と Y の距離です。 ほとんどの場合、ユーザーがマシンの正面に立っていて、スクライブがトーチの右側にある場合は、正の X オフセットが必要になります。 同様に、スクライブがトーチの後方にある場合は、正の Y オフセットが必要になります。 トーチのオフセットは常にゼロである必要があります。

最後のステップは、必要なスクライブ遅延を設定することです。

1. アーム遅延-スクライブがマテリアルの表面に下降する時間を許可します。

2. 遅延時-モーションが開始する前にスクライブが開始する時間を許可します。

[構成]タブにパラメーターを保存します。

上記の指示が完了した後、MDI 入力で M3 $ 1 S1 コマンドを発行することにより、スクライブを手動でテストできます。 ユーザーは、この方法を使用して小さなディボットをスクライブしてから、同じ場所でトーチをパルスして、スクライブとトーチの間のオフセットを揃えると便利な場合があります。

G コードのスクライブを使用するには：

```
...
M52 P1 (paused motion on)
F#<_hal[plasmac.cut-feed-rate]>
T1 M6 (select scribe)
G43 H0 (apply offsets for current tool)
M3 $1 S1 (start the scribe)
.
M5 $1 (stop the scribe)
.
T0 M6 (select torch)
G43 H0 (apply offsets for current tool)
G0 X0 Y0 (parking position)
M5 $-1 (end all)
```

プログラムの最後に、最後の高速駐車移動の前にトーチに切り替えて、マシンがアイドル状態で常に同じ状態になるようにすることをお勧めします。

ユーザーは、適切な G コードを使用して、プログラム中にトーチとスクライブを何度でも切り替えることができます。

M3 S1（$ n なし）を発行すると、マシンは M3 $ 0 S1 が発行されたかのように動作し、M5（$ n なし）を発行すると、マシンは M5 $ 0 が発行されたかのように動作します。 これにより、以前の G コードファイルに下位互換性を提供するために、デフォルトでトーチの発射が制御されます。

警告

<machine_name> .hal ファイルに既存の手動工具交換パラメータが設定されている場合、PlasmaC はそれを自動工具交換に変換します。

3.16.7.23 スポッティング

穴あけなどの前に材料に印を付けるためのスポッティングを実現するために、PlasmaC はトーチを短時間パルスして、穴を開ける場所に印を付けることができます。

スポッティングは、次の手順で構成できます。

1. 構成パネルの[スポッティング]セクションでアーク電圧のしきい値を設定します。 電圧しきい値をゼロに設定すると、トーチを開始するとすぐに遅延タイマーが開始されます。 電圧しきい値をゼロより上に設定すると、アーク電圧がしきい値電圧に達したときに遅延タイマーが開始されます。
2. 設定パネルのスポッティングセクションでタイムオンを設定します。 タイムオンタイマーが経過すると、トーチがオフになります。 時間は 0 から 9999 ミリ秒まで調整可能です。

次に、プラズマトーチをスポッティングツールとして選択する M3 $ 2 S1 コマンドを使用して、G コードでトーチをオンにします。

スポッティングを有効にするには、複数のツールを参照してください。

LinuxCNC（PlasmaC）では、M3 コマンドと M5 コマンドの間に何らかの動きが必要です。 このため、高速での最小限の動きをプログラムする必要があります。

G コードの例は次のとおりです。

```
G21 (metric)
F99999 (high feed rate)
.
.
G0 X10 Y10
M3 $2 S1 (spotting on)
G91 (relative distance mode)
G1 X0.000001
G90 (absolute distance mode)
M5 $2 (spotting off)
.
.
G0 X0 Y0
G90
M2
```

Note

99999 の高い送り速度は、モーションがマシンの最高送り速度になるようにするためのものです。

重要

一部のプラズマ切断機は、この機能には適していません。

プラズマ切断機がこの機能を利用できることを確認するために、ユーザーがいくつかのテストスポッティングを実行することをお勧めします。

3.16.7.24 Hypertherm PowerMax Communications

通信は、RS485 ポートを備えた HyperthermPowerMax プラズマカッターで確立できます。 この機能により、材料ファイルのカットパラメータからカットモード、カットアンペア数、ガス圧を自動的に設定できます。

ガス圧がゼロに設定されている場合、PowerMax は、カットモード、カット電流、トーチタイプ、およびトーチ長から必要な圧力を自動的に計算します。

切断モードを変更すると、ガス圧がゼロに設定され、マシンは自動ガス圧モードを使用します。

これらのパラメータの最大値と最小値はプラズマカッターから読み取られ、カットパラメータの関連するスピンボタンはこれらの値によって制限されます。 通信が確立されるまで、ガス圧をゼロから変更することはできません。

この機能は、<machine_ name> .ini ファイルの[PLASMAC]セクションで PM_PORT 変数に正しいポート名を設定することで有効になります。 PM_PORT 変数が<machine_name> .ini ファイルに設定されていない場合、この機能に関連付けられているウィジェットは表示されません。

USB0 で HyperthermPowerMax 通信を有効にする例：

```
[PLASMAC]
PM_PORT = /dev/ttyusb0
```

ユーザーがポートの名前がわからない場合は、構成ディレクトリに Python スクリプトがあり、使用可能なすべてのポートが表示されます。また、PlasmaC GUI でこの機能を有効にする前に、Plasma ユニットとの通信をテストするために使用できます。

```
python ~/<user_name>/linuxcnc/configs/<the_user_config>/pmx_test.py
```

ガス圧単位の表示（psi または bar）は、通信リンクの初期設定中に受信したデータによって決定され、実行パネルのガス圧スピンボタンの横に表示されます。

PowerMax マシンは、通信が確立された後にリモートモードになり、この時点では（PlasmaC GUI を介して）リモートでのみ制御できます。 PowerMax ディスプレイを観察することで、接続を検証できます。

PowerMax をローカルモードに戻すには、次のいずれかを実行できます。

1. [実行]パネルから PowerMax 通信を無効にする
2. LinuxCNC を閉じます。これにより、シャットダウン中に PowerMax がローカルモードになります。
3. PowerMax を 30 秒間オフにしてから、電源をオンに戻します。

ヒント

PowerMax 通信がアクティブな場合、メッシュモードを選択すると、PowerMax ユニットで CPA モードが自動的に選択されます。

Note

PowerMax 通信機能を使用するには、Pythonpyserial モジュールがインストールされている必要があります。 pyserial がインストールされていない場合、エラーメッセージが表示されます。

pyserial をインストールするには、ターミナルウィンドウに次のコマンドを入力します。

```
sudo apt install python-serial
```

一般的な接続図は、このドキュメントの付録と確認済みの動作インターフェイスに示されています。

3.16.7.25 エラーメッセージ

障害が発生したときにユーザーに通知するために、PlasmaC によって出力されるエラーメッセージがいくつかあります。 メッセージは、クリティカルと警告の 2 つのグループに分けることができます。

重大なエラーが発生すると、実行中のプログラムが一時停止します。オペレーターは、続行する前にエラーの原因をクリアする必要があります。

切断中にエラーが発生した場合は、機械が一時停止している間、前進または後進が許可され、ユーザーは切断を再開する前に機械の位置を変更できます。

エラーがクリアされると、プログラムが再開される場合があります。

これらのエラーは、対応するセンサーが切断中にアクティブ化されたことを示します。

- breakaway switch activated program is paused

ブレイクアウェイスイッチ起動プログラムが一時停止

- float switch activated program is paused

フロートスイッチ起動プログラムが一時停止している

- ohmic probe activated program is paused

オーミックプローブ起動プログラムが一時停止している

これらのエラーは、プロービングが開始される前に対応するセンサーがアクティブ化されたことを示します。

- ohmic probe detected before probing program is paused

プローブプログラムが一時停止する前にオーミックプローブが検出されました

- float switch detected before probing program is paused

プローブプログラムが一時停止する前にフロートスイッチが検出されました

- breakaway switch detected before probing program is paused

プロービングプログラムが一時停止する前にブレークアウェイスイッチが検出されました

M5 コマンドに到達する前に、切断動作中に ArcOK 信号が失われました。

- valid arc lost program is paused

有効なアークロストプログラムが一時停止されています

ワークピースが検出される前に、Z 軸が下限に達しました。

- bottom limit reached while probing down program is paused

プローブダウンプログラムが一時停止しているときに下限に達しました

ワークピースが高すぎるため、安全にすばやく削除できません。

- material too high for safe traverse program is paused

安全なトラバースプログラムには高すぎる材料が一時停止されています

実行パネルのカットパラメータのこれらの値の 1 つが無効です（例：ゼロに設定されている場合）：

- invalid pierce height or invalid cut height or invalid cut volts program is paused

無効なピアス高さまたは無効なカット高さまたは無効なカットボルトプログラムが一時停止されています

構成パネルのアーク開始試行で示された回数の開始を試みた後、アークが検出されませんでした。

- no arc detected after <n>d start attempts program is paused

<n> d の開始試行プログラムが一時停止した後、アークが検出されませんでした

- no arc detected after <n>d start attempts manual cut is stopped

<n> d の開始試行後に手動カットが停止した後、アークが検出されない

THC により、切断中に下限に達しました。

- bottom limit reached while THC moving down program is paused

THC の下降プログラムが一時停止しているときに下限に達しました

THC により、切断中に上限に達しました。

- top limit reached while THC moving up program is paused

THC 上昇プログラムが一時停止している間に上限に達しました

警告エラーは実行中のプログラムを一時停止することはなく、情報提供のみを目的としています。

これらのエラーは、プローブテストが開始される前に対応するセンサーがアクティブ化されたことを示します。

- プローブテストが中止される前にオーミックプローブが検出されました

プローブテストが中止される前にオーミックプローブが検出されました

- float switch detected before probing probe test aborted

プロービングプローブテストが中止される前にフロートスイッチが検出されました

- breakaway switch detected before probing probe test aborted

プロービングプローブテストが中止される前にブレークアウェイスイッチが検出されました

これは、ゼロ点を見つけるためにプローブする前にプローブの接触が失われたことを示しています。

- probe trip error while probing

プローブ中のプローブトリップエラー

これは、プローブテスト中に下限に達したことを示します。

- bottom limit reached while probe testing

プローブテスト中に下限に達しました

これは、THC が切断中に Z 軸を上げるため、安全な高さが低下したことを示しています。

- safe traverse height has been reduced

安全なトラバースの高さが減少しました

これは、PlasmaC の起動時にアーク電圧の値が無効（NAN または INF）であったことを示しています。

- invalid arc-voltage-in

無効なアーク電圧入力

3.16.7.26 PlasmaC の更新

PlasmaC の更新通知は、https：//forum.linuxcnc.org/plasmac/37233-plasmac-updates に掲載されています。

ユーザー名を作成し、上記のスレッドにサブスクライブして更新通知を受け取ることを強くお勧めします。

標準の ISO インストールの場合、LinuxCNC は、新しいマイナーリリースがリリースされたときにのみ更新されます。 その後、PlasmaC は、LinuxCNC の更新後に初めて実行されたときに、その構成を自動的に更新します。

LinuxCNC は通常、ターミナルウィンドウに次のコマンドを（一度に 1 つずつ）入力することで更新されます。

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

LinuxCNC の新しいマイナーリリースがリリースされるまで、拡張機能とバグ修正は標準インストールでは利用できません。 ユーザーが新しい PlasmaC バージョンがプッシュされるたびに更新する場合は、
http：//buildbot.linuxcnc.org/の手順に従って、標準の LinuxCNC リポジトリではなく LinuxCNCBuildbot リポジトリを使用できます。

3.16.8 PlasmaC の高度なトピック

1.1.1.1 カスタムユーザーボタン

PlasmaC GUI には、<machine_name> .ini ファイルにコマンドを追加することでプログラム可能なボタンがあります。 ボタンの数は、ユーザーが Axis または Gmoccapy で PlasmaC を使用することを選択したかどうかによって異なります。

Axis GUI のメインウィンドウには、左から右に 1～5 の番号が付いた 5 つのボタンがあります。

Gmoccapy GUI のメインウィンドウには、上から下に 1～4 の番号が付いた 4 つのボタンがあります。

Extras パネルには、左上から右下に 10～19 の番号が付いた追加の 10 個のボタンがあります。

ボタンのすべての<machine_name> .ini ファイル設定は、[PLASMAC]セクションにあります。

ボタンに表示されるテキストは、次のように設定されます。

```
BUTTON_n_NAME = HAL Show
```

ここで、n はボタン番号、HALShow はテキストです。

Axis GUI の複数行のテキストの場合、テキストを\で分割します

```
BUTTON_n_NAME = HAL\Show
```

ボタンは、外部コマンド、G コード、または 6 つの特別な機能のいずれかを実行できます。HAL ピンの切り替え、プローブテスト、オーミックテスト、カットタイプ、消耗品の変更、または X と Y のタッチオフです。

これに加えて、Extras パネルのボタン 10～19 は、特別な機能である Cut Recovery、Load G-Code プログラムを実行したり、ボタン名の代わりにカスタム画像を表示したりできます。

外部コマンド

外部コマンドを実行するには、コマンドの前に%文字を付けます。

```
BUTTON_n_CODE = %halshow
```

G コード

G コードを実行するには、実行するコードを入力するだけです。

```
BUTTON_n_CODE = G0 X100
```

既存のサブルーチンを実行します。

```
BUTTON_n_CODE = o<my_subroutine> call
```

<machine_name> .ini ファイル変数は、{}を使用して入力できます（スペースは}の後に配置する必要があります）

```
BUTTON_n_CODE = G0 X{JOINT_0 HOME} Y1
BUTTON_n_CODE = G53 G0 Z[{AXIS_Z MAX_LIMIT} - 1.001]
```

コードを\記号で区切ることにより、複数のコードを実行できます。 例外は、ボタンごとに 1 つのコマンドである必要がある特別なコマンドです。

```
BUTTON_n_CODE = G0 X0 Y0 \ G1 X5 \ G1 Y5
```

外部コマンドと G コードを同じボタンに混在させることができます。

```
BUTTON_n_CODE = %halshow \ g0x.5y.5 \ %halmeter
```

HAL ピンを切り替えます

次のコードを使用すると、ユーザーはボタンを使用して HAL ビットピンの現在の状態を反転できます（このコードは単一のコマンドとして使用する必要があり、ボタンごとに 1 つの HAL ビットピンのみを制御できます）。

```
BUTTON_n_CODE = toggle-halpin my-hal-pin-name
```

コードを設定した後、クリックするとボタンが緑色に変わり、ピンの状態が反転し、もう一度クリックするまで「ラッチ」されたままになります。 もう一度クリックすると、ボタンが元の色に戻り、ピンの状態が反転します。

プローブテスト

PlasmaC はプローブを開始し、材料が検出されると、Z 軸は実行パネルの[カットパラメータ]セクションに現在表示されているピアスの高さまで上昇します。 その後、PlasmaC は、この状態で指定された時間（秒単位）待機してから、Z 軸を開始位置に戻します。 30 秒の遅延の例を以下に示します。

```
BUTTON_n_CODE = probe-test 30
```

オーミックテスト

PlasmaC はオーミックプローブイネーブル出力信号を有効にし、オーミックプローブ入力が検知されると、モニターパネルの LED インジケーターが点灯します。 これの主な目的は、ショートしたトーチチップのクイックテストを可能にすることです。

```
BUTTON_n_CODE = ohmic-test
```

カットタイプ

このボタンを選択すると、2 つのカットタイプ、ピアスとカット（デフォルトのカットモード）またはピアスのみが切り替わります。

```
BUTTON_n_CODE = cut-type
```

消耗品の変更

このボタンを押すと、マシンが一時停止しているときにトーチが指定された座標に移動し、ユーザーがトーチの消耗品を簡単に変更できるようになります。

有効なエントリは XnnnYnnnFnnn です。 送り速度（F）は必須であり、X 座標または Y 座標の少なくとも 1 つが必要です。 X 座標と Y 座標は、絶対的なマシン座標です。 X または Y が欠落している場合は、その軸の現在の座標が使用されます。

以前の座標に戻すには、次の 3 つの方法があります。

1. 消耗品の変更ボタンをもう一度押します。トーチは元の座標に戻り、マシンはこの位置でユーザーがプログラムを再開するのを待ちます。
2. プログラムを再開します-トーチを元の座標に戻すと、プログラムが再開されます。
3. プログラムを停止します-トーチを元の座標に戻すと、プログラムは中止されます。

```
BUTTON_n_CODE = change-consumables X10 Y10 F1000
```

X と Y をタッチオフ

ボタン 1～19 は、X 軸と Y 軸をタッチオフ（ゼロ調整）し、G コードプログラムの実行を開始するために適切な高さに Z を移動することができます。

ユーザーが AxisGUI を使用している場合は、プログラムをリロードし、ライブプロットをクリアし、プレビューウィンドウを再ズームするルーチンを実行するコマンドを追加することもできます。 これにより、ユーザーは、現在ロードされている G コードプログラムに対して新しいゼロ座標がどこにあるかを簡単に確認できます。

X 軸と Y 軸をゼロに設定するには：

```
G10 L20 P0 X0 Y0
```

Z 軸を MAXIMUM_LIMIT から少し離して、切断の準備をします（次の例では 5.0mm）。

```
G53 G0 Z[{AXIS_Z MAX_LIMIT} - 5.0]
```

プレビューウィンドウを更新するには（Axis GUI のみ）：

```
%halcmd setp axisui.refresh 1
```

3 つのタスクすべてを実行するために組み合わせる：

```
G10 L20 P0 X0 Y0 \ G53 G0 Z[{AXIS_Z MAX_LIMIT} - 5.0] \ %halcmd setp axisui.refresh 1
```

カットリカバリー

このボタンを押すと、カットリカバリダイアログが表示され、ユーザーはトーチをカットパスから外して、カットを再開するのに適した材料を見つけることができます。 この機能は Extras パネルからのみ使用でき、ボタンは一時停止中のみ有効です。

```
BUTTON_n_CODE = cut-recovery
```

ロード

エクストラパネルのボタン 10〜19 は、次の形式を使用して G コードプログラムをロードできます。

```
BUTTON_n_CODE = load path/to/G-Code.ngc
```

パスは、<machine_name> .ini ファイルの PROGRAM_PREFIX で指定されたディレクトリからの相対パスです。これは通常〜/ linuxcnc / nc_files です。 例：ユーザーの G コードファイルが plasma という名前の PROGRAM_PREFIX ディレクトリのサブディレクトリにある場合、エントリは plasma /G-Code.ngc になります。

エクストラパネルのボタン 10〜19 は、テキストの代わりに画像を表示できます。 画像が指定されている場合、これが優先され、ボタン名の代わりに画像が表示されます。

```
BUTTON_n_IMAGE = path/to/image.png
```

パスは構成ディレクトリからの相対パスであるため、イメージが images という名前の構成ディレクトリのサブディレクトリにある場合、エントリは images /image.png になります。

すべての画像は 60x60 形式で表示されます。

3.16.8.1 テストパネル

マシンの構成フォルダーに test という名前のディレクトリがあります。このディレクトリには、<machine_name> .ini ファイルで参照されている構成例をテストするために使用できる、単純なテストパネルと関連する Python ファイルがあります。

ユーザーがこれを sim 構成で使用する場合は、このディレクトリへのリンクを作成し、それに合わせて<machine_name> .ini ファイルを編集する必要があります。

3.16.8.2 ボイドセンシングの調整

ボイドロックを THC に適用する前に、しきい値を超えた必要な連続回数を調整することにより、ボイド検知をさらに調整することができます。

これを調整するためのパラメーターの名前は plasmac.kerf-errors-max で、デフォルト値は 2 です。この値を変更するには、パラメーターと必要な値を<machine_name> _connections.hal ファイルに追加します。 次の例では、しきい値を超えるために必要な連続回数を 3 に設定します。

```
setp plasmac.kerf-error-max 3
```

3.16.8.3 ロストアークディレイ

一部のプラズマ電源/マシン構成では、カット中に一時的に Arc OK 信号が失われるか、カットの終わり近くで永久に失われ、PlasmaC がプログラムを一時停止して「有効なアーク損失」エラーを報告する場合があります。失われた ArcOK 信号が回復した場合、または M5 コマンドに到達する前にプログラム/エラーの一時停止を防ぐ遅延（秒単位）を設定するために使用できる plasmac.arc-lost-delay という名前の HAL ピンがあります。 設定された遅延期間が終了します。

次のコードは、0.1 秒の遅延を設定します。

```
setp plasmac.arc-lost-delay 0.1
```

<machine_name> _connections.hal ファイルでこのパラメーターを設定することをお勧めします。

この設定は、ユーザーが上記の症状を経験した場合にのみ使用する必要があります。 また、ユーザーは適切な Ignore Arc OK G-Code コマンドを使用して、同様の結果を得ることができることにも注意してください。

3.16.8.4 PlasmaC 状態出力

PlasmaC には plasmac.state-out という名前の HAL ピンがあり、これを使用してユーザーがコーディングしたコンポーネントとインターフェイスし、PlasmaC の現在の状態を提供できます。

PlasmaC が遭遇する可能性のあるさまざまな状態は次のとおりです。

状態	内容
0	IDLE

1	PROBE_HEIGHT
2	PROBE_DOWN
3	PROBE_UP
4	ZERO_HEIGHT
5	PIERCE_HEIGHT
6	TORCH_ON
7	ARC_OK
8	PIERCE_DELAY
9	PUDDLE_JUMP
10	CUT_HEIGHT
11	CUT
12	PAUSE_AT_END
13	SAFE_HEIGHT
14	MAX_HEIGHT
15	FINISH
16	TORCHPULSE
17	PAUSED_MOTION
18	OHIMC_TEST
19	PROBE_TEST
20	SCRIBING
21	CONSUMABLE_CHANGE_ON
22	CONSUMABLE_CHANGE_OFF
23	CUT_RECOVERY_ON
24	CUT_RECOVERY_OFF
25	DEBUG

DEBUG 状態はテストのみを目的としており、通常は発生しません。

3.16.9 付録

1.1.1.1 構成例

PlasmaCHAL コンポーネントを使用してプラズマ切断機をシミュレートする AxisGUI と GmoccapyGUI の両方の構成ファイルの例があります。

- metric_plasmac.ini
- imperial_plasmac.ini

ユーザーが設定例を～/ linuxcnc / configs ディレクトリにコピーした場合は、Configurator を更新モードで実行することで更新できます。 これにより、コピーされた PlasmaC システムファイルが、ユーザーが更新した最新バージョンへのリンクに置き換えられます。

3.16.9.1 NGC サンプル

nc_files / plasmac にいくつかのサンプル G コードファイルがあります。

3.16.9.2 LinuxCNC インストールのタイプを変更する

インストールのタイプを変更する前に、既存の PlasmaC インストールを最新バージョンに更新する必要があります。 PlasmaC が最新バージョンでない場合、インストールタイプを変更すると、既存の PlasmaC 構成が破損する可能性があります。

続行する前に、既存の PlasmaC 構成のバックアップコピーを作成することをお勧めします。

ユーザーが LinuxCNC インストールのタイプをパッケージインストールからランインプレースに、またはその逆に変更した場合、ユーザーは〜/ linuxcnc / configs ディレクトリからではなく、新しい LinuxCNC インストールディレクトリから Configurator を実行する必要があります。 これは、正しいインストールタイプのファイルリンクをリセットするために必要です。

ターミナルウィンドウに次のコマンドを入力し、次のいずれかを実行します。

新しいパッケージのインストールにリセットします。

```
python /usr/share/doc/linuxcnc/examples/sample-configs/by_machine/plasmac/configurator.py
```

新しい RunInPlace インストールにリセットします。

```
python〜 / linuxcnc-dev / configs / by_machine / plasmac / configureator.py
```

3.16.9.3 PlasmaC 固有の G コード

内容	コード
Begin cut	M3 $0 S1
End cut	M5 $0
Begin scribe	Begin scribe
End scribe	M5 $1
Begin center spot	M3 $2 S1
End center spot	M5 $2
End all the above	M5 $-1
Select a material	M190 Pn n denotes the material number
Wait for material change confirmation	M66 P3 L3 Qn n is delay time (in seconds) This value may need to be increased for very large material files

Set feed rate from material	F#<_hal[plasmac.cut-feed-rate]>
Enable Ignore Arc OK	M62 P1 (synchronized with motion) M64 P1 (immediate)
Disable Ignore Arc OK	M63 P1 (synchronized with motion) M65 P1 (immediate)
Disable THC	M62 P2 (synchronized with motion) M64 P2 (immediate)
Enable THC	M63 P2 (synchronized with motion) M65 P2 (immediate)
Disable Torch	M62 P3 (synchronized with motion) M64 P3 (immediate)
Enable Torch	M63 P3 (synchronized with motion) M65 P3 (immediate)
Set velocity to a percentage of feed rate	M67 E3 Qn (synchronized with motion) M68 E3 Qn (immediate) n is the percentage to set 10 is the minimum, below this will be set to 100% 100 is the maximum, above this will be set to 100% It is recommended to have M68 E3 Q0 in preamble and postamble
Cutter compensation - left of path	G41.1 D#<_hal[plasmac_run.kerf-width-f]>
Cutter compensation - right of path	G42.1 D#<_hal[plasmac_run.kerf-width-f]>
Cutter compensation off	G40 Note that M62 through M68 are invalid while cutter compensation is on
Cut holes at 60% feed rate	#<holes> = 1 for holes less than 32mm (1.26") diameter
Cut holes at 60% feed rate, turn torch off at hole end, continue hole path for overcut	#<holes> = 2 for holes less than 32mm (1.26") diameter overcut length = 4mm (0.157")
Cut holes and arcs at 60% feed rate	#<holes> = 3 for holes less than 32mm (1.26") diameter for arcs less than 16mm (0.63") radius
Cut holes and arcs at 60% feed rate, turn torch off at hole end, continue hole path for overcut	#<holes> = 4 for holes less than 32mm (1.26") diameter for arcs less than 16mm (0.63") radius overcut length = 4mm (0.157")
Specify hole diameter for	#<h_diameter> = n (n is the diameter, use the same units

#<holes> = 1-4	system as the rest of the G-Code file)
Specify hole velocity for #<holes> = 1-4	#<h_velocity> = n (n is the percentage, set the percentage of the current feed rate)
Specify overcut length	#<oclength> = n (n is the length, use the same units system as the rest of the G-Code file)
Specify pierce-only mode	#<pierce-only> = n (n is the mode, 0=normal cut mode, 1=pierce only mode)
Create or edit materials. options: 1 - Create temporary default 2 - Add if not existing 3 - Overwrite if existing else add new	mandatory parameters: (o=<option>, nu=<nn>, na=<ll>, ph=<nn>, pd=<nn>, ch=<nn>, fr=<nn>) optional parameters: (kw=<nn>, th=<nn>, ca=<nn>, cv=<nn>, pe=<nn>, gp=<nn>, cm=<nn>, jh=<nn>, jd=<nn>)

3.16.9.4 PlasmaCGcode の例

説明	例
Select material and do a normal cut	M190 P3 M66 P3 L3 Q1 F#<_hal[plasmac.cut-feed-rate]> M3 $0 S1 .. M5 $0
Set velocity to 100% of CutFeedRate	M67 E3 Q0 or M67 E3 Q100
Set velocity to 60% of CutFeedRate	M67 E3 Q60
Set velocity to 40% of CutFeedRate	M67 E3 Q40
Cut a hole with 60% reduced speed using velocity setting	G21 (metric) G64 P0.05 M52 P1 (allow paused motion) F#<_hal[plasmac.cut-feed-rate]> G0 X10 Y10 M3 $0 S1 (start cut) G1 X0 M67 E3 Q60 (reduce feed rate to 60%) G3 I10 (the hole) M67 E3 Q100 (restore feed rate to 100%)

	M5 $0 (end cut) G0 X0 Y0 M2 (end job)
Cut a hole with 60% reduced speed using the #<holes> command	G21 (metric) G64 P0.05 M52 P1 (allow paused motion) #<holes> = 1 (velocity reduction for holes) F#<_hal[plasmac.cut-feed-rate]> G0 X10 Y10 M3 $0 S1 (start cut) G1 X0 G3 I10 (the hole) M5 $0 (end cut) G0 X0 Y0 M2 (end job)
Cut a hole with overcut using torch disable	G21 (metric) G64 P0.05 M52 P1 (allow paused motion) F#<_hal[plasmac.cut-feed-rate]> G0 X10 Y10 M3 $0 S1 (start cut) G1 X0 M67 E3 Q60 (reduce feed rate to 60%) G3 I10 (the hole) M62 P3 (turn torch off) G3 X0.8 Y6.081 I10 (continue motion for 4mm) M63 P3 (allow torch to be turned on) M67 E3 Q0 (restore feed rate to 100%) M5 $0 (end cut) G0 X0 Y0 M2 (end job)
Cut a hole with overcut using the #<holes> command	G21 (metric) G64 P0.05 M52 P1 (allow paused motion) #<holes> = 2 (overcut for holes) F#<_hal[plasmac.cut-feed-rate]> G0 X10 Y10 M3 $0 S1 (start cut) G1 X0 G3 I10 (the hole)

	M5 $0 (end cut) G0 X0 Y0 M2 (end job)
Cut a hole with 6.5mm overcut using the #<holes> command	G21 (metric) G64 P0.05 M52 P1 (allow paused motion) #<holes> = 2 (overcut for holes) <oclength> = 6.5 (6.5mm overcut length) F<_hal[plasmac.cut-feed-rate]> G0 X10 Y10 M3 $0 S1 (start cut) G1 X0 G3 I10 (the hole) M5 $0 (end cut) G0 X0 Y0 M2 (end job)
Select scribe and select torch at end of scribing	.. M52 P1 (paused motion on) F#<_hal[plasmac.cut-feed-rate]> T1 M6 (select scribe) G43 H0 (apply offsets) M3 $1 S1 (start plasmac with scribe) .. T0 M6 (select torch) G43 H0 (apply offsets) G0 X0 Y0 (parking position) M5 $1 (end)
Hole center spotting	(Requires a small motion command or nothing happens) G21 (metric) F99999 (high feed rate) G0 X10 Y10 M3 $2 S1 (spotting on) G91 (relative distance mode) G1 X0.000001 G90 (absolute distance mode) M5 $2 (spotting off) G0 X0 Y0 G90 M2
Create temporary default	(o=1, nu=2, na=5mm Mild Steel 40A, ph=3.1, pd=0.1, ch=0.75,

material	fr=3000)
Edit material, if not existing create a new one	(o=3, nu=2, na=5mm Mild Steel 40A, ph=3.1, pd=0.1, ch=0.75, fr=3000,kw=1.0)

3.16.9.5 Mesa THCAD

Mesa THCAD は、プラズマ切断機からアーク電圧を取得する最も一般的な方法です。 THCAD は、パラレルポート構成または MesaElectronics ハードウェアを使用した構成に使用できます。

推奨される方法は、THCAD に必要なパラメーターを<machine_name> _connections.hal ファイルに設定することです。

ソフトウェアエンコーダを使用したパラレルポート設定の例。

THCAD ジャンパー設定 UNIPOLAR および F / 64

Note

ユーザーのコンピューターのレイテンシーが良好な場合、ユーザーは F / 32 を使用できる可能性があります

```
loadrt encoder num_chan=1
setp encoder.0.counter-mode 1
setp encoder.0.position-scale -1
addf encoder.update-counters base-thread
addf encoder.capture-position servo-thread
net arc-voltage-raw parport.1.pin-04-in encoder.0.phase-A
```

次に、ユーザーは、PlasmaCConfigurator のアーク電圧 HAL ピンエントリが次のようになることに注意する必要があります。

```
encoder.0.velocity
```

オンボードハードウェアエンコーダーを使用した Mesa7i96 構成の例を以下に示します。 他の Mesa ハードウェアも同様ですが、ボード識別子が異なります。

THCAD ジャンパー設定 UNIPOLAR および F / 32

```
setp hm2_7i96.0.encoder.00.scale -1
setp hm2_7i96.0.encoder.00.filter 1
```

```
setp hm2_7i96.0.encoder.00.counter-mode 1
```

次に、ユーザーは、PlasmaCConfigurator のアーク電圧 HAL ピンエントリが次のようになることに注意する必要があります。

```
hm2_7i96.0.encoder.00.velocity
```

Note

THCAD を使用していて、返されるアーク電圧に多くのノイズがある場合に役立つ可能性のあるローパスフィルターが利用可能です。

THCAD-10

プラズマ CNC ポートに接続する場合、分周比はプラズママシンから選択されます。

プラズママシンにフルアーク電圧を接続する場合、一般的な設定は、アーク負から THCAD 負まで 1 メガオームの抵抗を使用し、アーク正から THCAD 正まで 1 メガオームの抵抗を使用することです。 これにより、210V のフルスケール読み取り値が得られます。 THCAD は、最大 500V の過電圧を無期限に処理できます。

分周比を取得するには、（R1 + R2 + 100000）/ 100000 を使用します。上記の抵抗値の例は、（1000000 + 1000000 + 100000）/ 100000 で、比率は 21 になります。

重要

ユーザーが HF 始動プラズマ電源を使用している場合、これらの抵抗のそれぞれは、いくつかの高電圧抵抗器で構成されている必要があります。

THCAD-300

これはプラズママシンのフルアーク電圧に接続されており、分圧比= 1

これらの値は、構成パネルで電圧オフセットと電圧スケールの開始値を取得するために使用されます。

これらは、THCAD の背面に、次のことを示すキャリブレーションステッカーを介して配置されています。

```
THCAD-nnn
0V 121.1 kHz
5V 925.3 kHz
```

または同様の値の場合、これらはそれぞれ 0 ボルトとフルスケールボルトに対して THCAD によって生成される周波数です。

電圧スケールと電圧オフセットは、以下に示すように手動で計算するか、このオンライン計算機を使用して計算できるようになりました。

電圧スケール

式を使用する

```
Divider_Ratio / ((THCAD_Full_Scale_Frequency - THCAD_0V_Frequency) /  -
THCAD_Frequency_Divider / THCAD_Full_Scale_Voltage)
```

この結果は、構成を設定するときに構成パネルで電圧スケールを設定するために使用されます。

F / 32 のジャンパー設定を使用した THCAD-10 の例は次のようになります。

```
21 / ((925300 - 121100) / 32 / 10) = 0.00835613
```

上記の計算では、21 は分周器の計算例から、32 はジャンパー設定 F / 32、10 は THCAD カード自体のフルスケール読み取り値です。

THCAD-300 の例は次のようになります。

```
/ ((925300 - 121100) / 32 / 300) = 0.01193732
```

上記の計算では、1 は分周比、32 はジャンパー設定 F / 32、300 は THCAD カード自体のフルスケール読み取り値です。

電圧オフセット

次の式を使用します。

```
THCAD_0V_Frequency / THCAD_Frequency_Divider
```

この結果は、構成を設定するときに構成パネルで電圧オフセットを設定するために使用されます。

上記の THCAD-10 および THCAD-300 の例は次のとおりです。

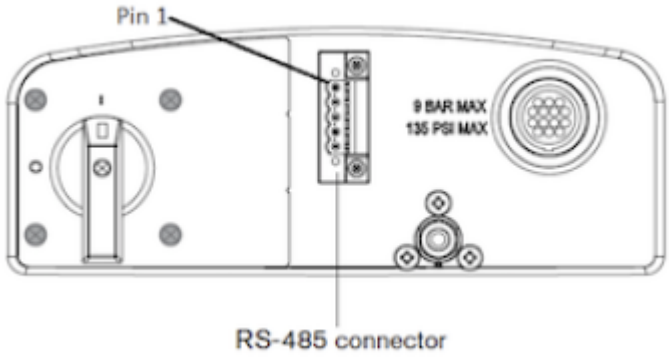
```
121100 / 32 = 3784.375
```

3.16.9.6 RS485 接続

Hypertherm RS485 配線図（括弧内の Hypertherm 内の配線色）：

マシンピン番号での接続	ブレイクアウトボードでの接続
1-Tx +（赤）	-> RXD +
2-Tx-（黒）	-> RXD-
3-Rx +（ブラウン）	-> T / R +
4-Rx-（白）	-> T / R-
5-GND（緑）	-> GND

マシンピン番号での接続	USB RS485 ピン番号での接続（左から右）
1:Tx +（赤）	1:T/R-(黒)
2:Tx-（黒）	2:T/R+(赤)
3:Rx +（ブラウン）	3:RXD-(白)
4:Rx-（白）	4:RXD+(茶)
5:GND（緑）	5:GND(緑)



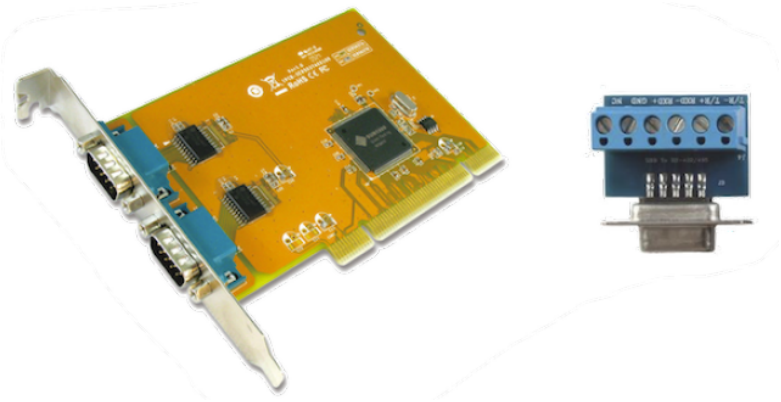
動作することがわかっている RS485 インターフェース：

DTECH DT-5019 USB から RS485 へのコンバーターアダプター：

マザーボードのシリアル接続またはシリアルカード（RS232）を RS485 に変換するには、次のものが必要です。

DTECH RS232 から RS485 へのコンバーター：

シリアルカードの例（ブレイクアウトボード付きの Sunnix SER5037A PCI カード）：

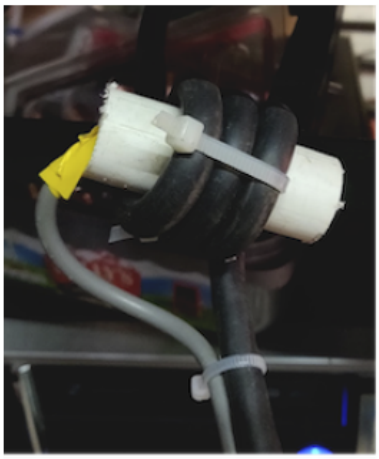


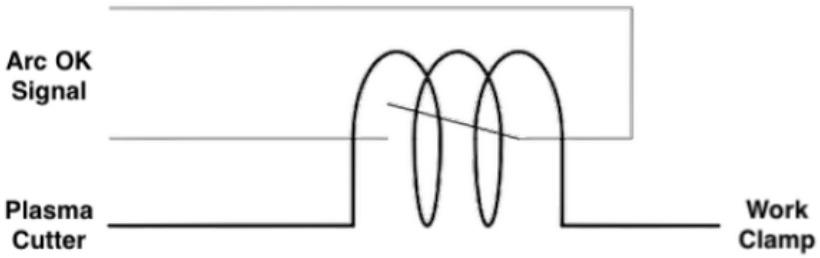
3.16.9.7 リードリレーでアーク OK

CNC ポートなしでプラズマ電源から ArcOK 信号を取得する効果的で非常に信頼性の高い方法は、非導電性チューブ内にリードリレーを取り付け、チューブの周りにワークリードを 3 ターン巻き付けて固定することです。

このアセンブリは、カッティングアークが確立されたときにのみ発生する電流がワークリードを流れるときにオンになるリレーとして機能します。

これには、PlasmaC をモード 0 ではなくモード 1 で操作する必要があります。詳細については、「INI ファイル」および「PlasmaC モード」のセクションを参照してください。





3.16.9.8

3.16.10 サポート

オンラインヘルプとサポートは、LinuxCNC フォーラムの PlasmaC セクションから入手できます。

ユーザーは、バックアップセクションの指示に従うことで、障害診断に役立つ完全なマシン構成を含む圧縮ファイルを作成できます。 結果のファイルは、コミュニティが特定の問題を診断するのに役立つ LinuxCNC フォーラムの投稿に添付するのに適しています。

4章 プログラミング

4.1 座標系

4.1.1 序章

この章では、LinuxCNC で使用されるオフセットについて説明します。 これらには以下が含まれます：

- 機械座標（G53）
- 9 つの座標系オフセット（G54-G59.3）
- グローバルオフセット（G92）およびローカルオフセット（G52）

4.1.2 機械座標系

LinuxCNC が開始されると、各軸の位置がマシンの原点になります。 軸が原点復帰すると、その軸の機械原点が原点復帰位置に設定されます。 機械の原点は、他のすべての座標系が基づいている機械座標系です。 G53 G コードは、機械座標系での移動に使用できます。

4.1.3 座標系

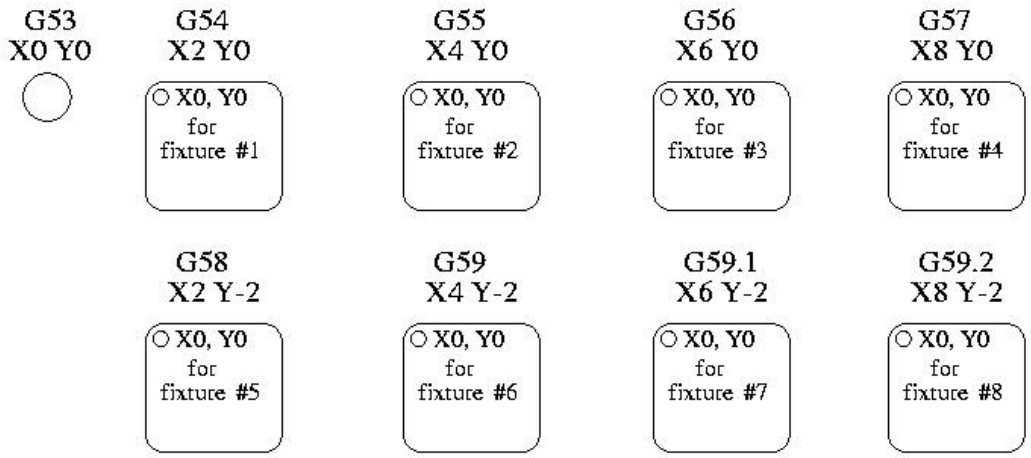


図 5-49

座標系のオフセット

- G54-座標系 1 を使用
- G55-座標系 2 を使用
- G56-座標系 3 を使用
- G57-座標系 4 を使用
- G58-座標系 5 を使用
- G59-座標系 6 を使用

- G59.1-座標系 7 を使用
- G59.2-座標系 8 を使用
- G59.3-座標系 9 を使用

座標系オフセットは、座標系を機械座標系からシフトするために使用されます。 これにより、マシン上の部品の位置に関係なく、部品の G コードをプログラムできます。 座標系オフセットを使用すると、同じ G コードを使用して複数の場所で部品を加工できます。

オフセットの値は、LinuxCNC の起動時に INI ファイルによって要求される VAR ファイルに保存されます。

VAR ファイルスキームでは、最初の変数番号は X オフセットを格納し、2 番目は Y オフセットを格納し、以下同様に 9 つの軸すべてについて格納します。 座標系のオフセットごとに、このような番号の付いたセットがあります。

各グラフィカルインターフェイスには、これらのオフセットの値を設定する方法があります。 VAR ファイル自体を編集してこれらの値を設定し、LinuxCNC を再起動して、LinuxCNC が新しい値を読み取るようにすることもできますが、これは推奨される方法ではありません。 G10、G52、G92、G28.1 などを使用することは、変数を設定するためのより良い方法です。

Table 5.1: Example of G55 parameters

Axis	Variable	Value
X	5241	2.000000
Y	5242	1.000000
Z	5243	-2.000000
A	5244	0.000000
B	5245	0.000000
C	5246	0.000000
U	5247	0.000000
V	5248	0.000000
W	5249	0.000000

これは、G55 のゼロ位置を絶対零度から X = 2 単位、Y = 1 単位、および Z = -2 単位移動するものとして読む必要があります。

値が割り当てられると、プログラムブロックで G55 を呼び出すと、格納されている値だけゼロ参照がシフトされます。 次の行は、各軸を新しいゼロ位置に移動します。 G53 とは異なり、G54 から G59.3 はモーダルコマンドです。 それらの 1 つが設定された後、それらはコードのすべてのブロックに作用します。 フィクスチャオフセットを使用して実行される可能性のあるプログラムでは、場所ごとに 1 つの座標参照のみが必要であり、すべての作業はそこで実行されます。 次のコードは、上記で設定した G55 オフセットを使用して正方形を作成する例として提供されています。

```
G55 ; use coordinate system 2
G0 X0 Y0 Z0
G1 F2 Z-0.2000
```

```
X1
Y1
X0
Y0
G0 Z0
G54 ; use coordinate system 1
G0 X0 Y0 Z0
M2
```

この例では、終わり近くの G54 は、すべてゼロのオフセットで G54 座標系を離れるので、絶対的なマシンベースの軸位置のモーダルコードがあります。 このプログラムは、それを実行したことを前提とし、ゼロをマシン化するコマンドとして終了コマンドを使用します。 G53 を使用して同じ場所に到着することは可能でしたが、そのコマンドはモーダルではなく、G55 オフセットの使用に戻った後に発行されたコマンドは、その座標系が引き続き有効であるためです。

1.1.1.1 デフォルトの座標系

オフセットシステムについて考えるとき、VAR ファイルのもう 1 つの変数が重要になります。 この変数の名前は 5220 です。デフォルトのファイルでは、その値は 1.00000 に設定されています。 これは、LinuxCNC の起動時に、最初の座標系をデフォルトとして使用する必要があることを意味します。 これを 9.00000 に設定すると、起動とリセットのデフォルトとして 9 番目のオフセットシステムが使用されます。 1 から 9 までの整数（実際には 10 進数）以外の値、または 5220 変数がない場合、LinuxCNC は起動時にデフォルト値の 1.00000 に戻ります。

4.1.3.1 座標系オフセットの設定

G10 L2x コマンドを使用して、座標系のオフセットを設定できます。

- G10 L2 P（1-9）-オフセットを値に設定します。 現在の位置は関係ありません。（詳細については、G10 L2 を参照してください）
- G10 L20 P（1-9）-現在の位置が値になるようにオフセットを設定します。（詳細は G10 L20 を参照）

4.1.4 ローカルおよびグローバルオフセット

1.1.1.1 G52 コマンド

G52 は、ワーク座標系内の一時的な「ローカル座標系オフセット」としてパートプログラムで使用されます。 ユースケースの例は、パーツの異なる場所で複数の同一のフィーチャーを加工する場合です。 フィーチャーごとに、G52 はワーク座標内のローカル基準点をプログラムし、サブプログラムが呼び出されて、そのポイントを基準にしてフィーチャーを加工します。

G52 軸オフセットは、ワークピース座標オフセット G54 から G59.3 を基準にしてプログラムされます。 ローカルオフセットとして、G52 は回転を含むワークピースオフセットの後に適用されます。 したがって、パーツフィーチャーは、パレット上のパーツの方向に関係なく、各パーツで同じように加工されます。

Caution

一時的なオフセットとして、パートプログラムのローカライズされたスコープ内で設定および設定解除します。他の g コードインタープリターでは、G52 は、マシンのリセット後、M02 または M30 では持続しません。 LinuxCNC では、G52 は G92 とパラメーターを共有します。これは、歴史的な理由から、これらのパラメーターを保持します。 以下の G92 永続性に関する注意を参照してください。

Caution

G52 と G92 は同じオフセットレジスタを共有します。 したがって、G52 の設定は、以前の G92 設定を上書きし、G92 の永続性が有効になっている場合、G52 はマシンのリセット後も永続化します。 これらの相互作用により、予期しないオフセットが発生する可能性があります。 以下の G92 および G52 の相互作用に関する注意を参照してください。

プログラミング G52X1 Y2 は、現在のワーク座標系の X 軸を 1 だけ、Y 軸を 2 だけオフセットします。したがって、DRO では、現在の工具位置の X 座標と Y 座標がそれぞれ 1 と 2 減少します。 前の例の Z など、コマンドで設定されていない軸は影響を受けません。以前の G52 Z オフセットは引き続き有効です。それ以外の場合、Z オフセットはゼロになります。

一時的なローカルオフセットは、G52 X0Y0 でキャンセルできます。 明示的にゼロ調整されていない軸は、前のオフセットを保持します。

G52 は G92 と同じオフセットレジスタを共有するため、G52 は DRO に表示され、G92 のラベルが付いたプレビューが表示されます。

4.1.4.1 G92 コマンド

G92 は通常、「グローバル座標系オフセット」または「ローカル座標系オフセット」という 2 つの概念的に異なる方法で使用されます。 G92 のコマンドセットには次のものが含まれます。

- G92-このコマンドを軸名とともに使用すると、値をオフセット変数に設定します。
- G92.1-このコマンドは、G92 変数にゼロ値を設定します。
- G92.2-このコマンドは一時停止しますが、G92 変数をゼロにしません。
- G92.3-このコマンドは、一時停止されたオフセット値を適用します。

グローバルオフセットとして、G92 はすべてのワーク座標系 G54 から G59.3 をシフトするために使用されます。使用例の例は、パレット上の既知の位置にあるフィクスチャで複数の同一部品を加工する場合ですが、パレットの位置は実行間またはマシン間で変わる可能性があります。 パレット上の基準点に対する各フィクスチャ位置オフセットは、ワーク座標系の 1 つである G54 から G59.3 に事前設定されており、G92 はパレットの基準点を「タッチオフ」するために使用されます。 次に、パーツごとに、対応するワーク座標系が選択され、パーツプログラムが実行されます。

Note

G10 R-ワーク座標系の回転は rs274ngc インタープリターに固有であり、G92 オフセットは回転後に適用されます。 G92 をグローバルオフセットとして使用すると、ワーク座標系の回転によって予期しない結果が生じる可能性があります。

ローカル座標系として、G92 はワーク座標系内の一時的なオフセットとして使用されます。 使用例の例は、異なる場所でいくつかの同一のフィーチャーを持つ部品を機械加工する場合です。 各フィーチャーについて、G92 を使用してローカル参照ポイントを設定し、サブプログラムを呼び出して、そのポイントからフィーチャーを加工します。

Note

パートプログラムでローカル座標系を使用してプログラミングする場合は、G92 の使用をお勧めしません。 代わりに、G52 を参照してください。ワークピースに対する目的のオフセットはわかっているが、現在の工具位置がわからない場合は、ローカル座標系のオフセットがより直感的になります。

プログラミング G92X0 Y0 Z0 は、現在の工具位置を、モーションなしで座標 X0、Y0、および Z0 に設定します。 G92 は絶対的な機械座標からは機能しません。 現在の場所から動作します。

G92 は、G92 コマンドが呼び出されたときに有効な他のオフセットによって変更された現在の場所からも機能します。 作業オフセットと実際のオフセットの違いをテストしているときに、G54 オフセットが G92 をキャンセルし、オフセットが有効になっていないように見えることがわかりました。 ただし、G92 はすべての座標で引き続き有効であり、他の座標系で予想される作業オフセットを生成しました。

デフォルトでは、G92 オフセットはマシンの起動後に復元されます。 マシンの起動時およびリセットまたはプログラムの終了後に G92 オフセットがクリアされるファナックの動作を希望するプログラマーは、.ini ファイルの[RS274NGC]セクションで DISABLE_G92_PERSISTENCE = 1 を設定することにより、G92 の永続性を無効にすることができます。

Note

G92.1 または G92.2 での使用の最後に、G92 オフセットをクリアすることをお勧めします。 G92 永続性を有効にして（デフォルト）LinuxCNC を起動すると、軸がホームになっているときに G92 変数のオフセットが適用されます。 以下の G92 永続性に関する注意を参照してください。

4.1.4.2 G92 値の設定

G92 コマンドは、現在の軸の位置から機能し、正しく加算および減算して、現在の軸の位置に G92 コマンドによって割り当てられた値を与えます。 以前のオフセットがにある場合でも、エフェクトは機能します。

したがって、X 軸が現在その位置として 2.0000 を示している場合、G92 X0 は-2.0000 のオフセットを設定して、X の現在の位置がゼロになるようにします。 G92 X2 は 0.0000 のオフセットを設定し、表示される位置は変更されません。 G92 X5.0000 は、現在表示されている位置が 5.0000 になるように、3.0000 のオフセットを設定します。

4.1.4.3 G92 永続性に関する注意

デフォルトでは、G92 オフセットの値は VAR ファイルに保存され、マシンのリセットまたは起動後に復元されます。

G92 パラメータは次のとおりです。

- 5210-有効化/無効化フラグ（1.0 / 0.0）
- 5211-X 軸オフセット
- 5212-Y 軸オフセット
- 5213-Z 軸オフセット
- 5214-A 軸オフセット
- 5215-B 軸オフセット
- 5216-C 軸オフセット
- 5217-U 軸オフセット
- 5218-V 軸オフセット
- 5219-W 軸オフセット

ここで、5210 は G92 イネーブルフラグ（有効の場合は 1、無効の場合は 0）であり、5211～5219 は軸オフセットです。 前のプログラムにオフセットを保存し、最後にそれらをクリアしなかった結果として、コマンドされた移動の結果として予期しない位置が表示される場合は、MDI ウィンドウで G92.1 を発行して、保存されたオフセットをクリアします。

LinuxCNC の起動時に VAR ファイルに G92 値が存在する場合、var ファイルの G92 値は、各軸の現在の位置の値に適用されます。 これがホームポジションであり、ホームポジションがマシンゼロとして設定されている場合、すべてが正しくなります。 実際のマシンスイッチを使用して、または各軸を既知のホーム位置に移動して軸ホームコマンドを発行することにより、ホームが確立されると、G92 オフセットが適用されます。 X 軸をホームにしたときに G92X1 が有効になっている場合、G92 がマシンの原点に適用されているため、DRO は予想される X：0.000 ではなく X：1.000 を読み取ります。 G92.1 を発行し、DRO がすべてゼロを読み取る場合は、最後に LinuxCNC を実行したときに G92 オフセットが有効になっています。

次のプログラムで同じ G92 オフセットを使用するつもりでない限り、ベストプラクティスは、G92 オフセットを使用する G コードファイルの最後に G92.1 を発行することです。

G92 オフセットが有効になっている処理中にプログラムが中止されると、起動するとプログラムが再びアクティブになります。 安全策として、環境を期待どおりに設定するための前文を常に用意してください。 さらに、.ini ファイルの[RS274NGC]セクションで DISABLE_G92_PERSISTENCE = 1 を設定すると、G92 の永続性を無効にできます。

4.1.4.4 G92 と G52 の相互作用に関する注意

G52 と G92 は同じオフセットレジスタを共有します。 .ini ファイルで G92 の永続性が無効になっていない限り（G92 コマンドを参照）、G52 オフセットは、マシンのリセット後、M02 または M30 でも永続化します。 プログラムの中止中に有効な G52 オフセットは、次のプログラムの実行時に意図しないオフセットを引き起こす可能性があることに注意してください。 上記の G92 永続性に関する注意を参照してください。

4.1.5 オフセットを使用したサンプルプログラム

1.1.1.1 ワーク座標オフセットを使用したサンプルプログラム

このサンプル彫刻プロジェクトは、中心の円の周りにほぼ星の形をした 4 つの.1 半径の円のセットをフライス加工します。 このように個別のサークルパターンを設定できます。

```
G10 L2 P1 X0 Y0 Z0 (ensure that G54 is set to machine zero)
G0 X-0.1 Y0 Z0
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0
M2
```

このように、他の 4 つの円のオフセットを作成する一連のコマンドを発行できます。

```
G10 L2 P2 X0.5 (offsets G55 X value by 0.5 inch)
G10 L2 P3 X-0.5 (offsets G56 X value by -0.5 inch)
G10 L2 P4 Y0.5 (offsets G57 Y value by 0.5 inch)
G10 L2 P5 Y-0.5 (offsets G58 Y value by -0.5 inch)
```

これらを次のプログラムにまとめました。

```
(a program for milling five small circles in a diamond shape)
G10 L2 P1 X0 Y0 Z0 (ensure that G54 is machine zero)
G10 L2 P2 X0.5 (offsets G55 X value by 0.5 inch)
G10 L2 P3 X-0.5 (offsets G56 X value by -0.5 inch)
G10 L2 P4 Y0.5 (offsets G57 Y value by 0.5 inch)
G10 L2 P5 Y-0.5 (offsets G58 Y value by -0.5 inch)
G54 G0 X-0.1 Y0 Z0 (center circle)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0
G55 G0 X-0.1 Y0 Z0 (first offset circle)
G1 F1 Z-0.25
```

```
G3 X-0.1 Y0 I0.1 J0
G0 Z0
G56 G0 X-0.1 Y0 Z0 (second offset circle)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0
G57 G0 X-0.1 Y0 Z0 (third offset circle)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0
G58 G0 X-0.1 Y0 Z0 (fourth offset circle)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G54 G0 X0 Y0 Z0
M2
```

さて、このプログラムに一連の G92 オフセットを適用する時が来ました。 いずれの場合も Z0 で実行されていることがわかります。 ミルがゼロ位置にある場合、プログラムの先頭で発行された G92 Z1.0000 は、すべてを 1 インチシフトします。 また、G92 で X および Y オフセットを追加することにより、XY 平面内でパターン全体をシフトすることもできます。 これを行う場合は、プログラムを終了する M2 の直前に G92.1 コマンドを追加する必要があります。 そうしないと、このプログラムの後に実行する可能性のある他のプログラムもその G92 オフセットを使用します。 さらに、LinuxCNC をシャットダウンすると G92 値が保存され、再起動するとそれらが呼び出されます。

4.1.5.1 G52 オフセットを使用したサンプルプログラム

（書かれる予定）

4.2 G コードの概要

4.2.1 概要

LinuxCNC G コード言語は、RS274 / NGC 言語に基づいています。 G コード言語は、コード行に基づいています。 各行（ブロックとも呼ばれます）には、いくつかの異なることを行うためのコマンドが含まれている場合があります。 プログラムを作成するために、コード行をファイルに収集することができます。

一般的なコード行は、先頭のオプションの行番号とそれに続く 1 つ以上の単語で構成されます。 単語は、文字とそれに続く数字（または数字に評価されるもの）で構成されます。 単語は、コマンドを与えるか、コマンドに引数を提供することができます。 たとえば、G1 X3 は、2 ワードの有効なコード行です。 G1 は、プログラムされた送り速度でプログラムされた終点まで直線的に移動することを意味するコマンドであり、X3 は引数値を提供

します（移動の終了時にXの値は3である必要があります）。 ほとんどのLinuxCNCGコードコマンドは、GまたはM（一般およびその他）で始まります。 これらのコマンドの単語は、Gコードおよび Mコードと呼ばれます。

LinuxCNC 言語には、プログラムの開始を示すインジケーターがありません。 ただし、インタープリターはファイルを扱います。 単一のプログラムが単一のファイルに含まれている場合もあれば、プログラムが複数のファイルに分散している場合もあります。 ファイルは、次の方法でパーセントで区切られる場合があります。 ファイルの最初の非空白行には、パーセント記号%のみが含まれている可能性があり、空白で囲まれている可能性があります。ファイルの後半（通常はファイルの最後）にも同様の行が含まれている可能性があります。 ファイルにM2またはM30が含まれている場合、パーセントでファイルを区切ることはオプションですが、そうでない場合は必須です。 ファイルの最初にパーセント行があり、最後にない場合、エラーが通知されます。 パーセントで区切られたファイルの有用な内容は、2番目のパーセント行の後で停止します。 それ以降は無視されます。

LinuxCNC Gコード言語には2つのコマンド（M2またはM30）があり、どちらもプログラムを終了します。 プログラムは、ファイルが終了する前に終了する場合があります。 プログラムの終了後に発生するファイルの行は実行されません。 通訳はそれらを読みさえしません。

4.2.2 行の形式

入力コードの許容行は、順番に次の行で構成されますが、1行に許可される文字数の最大数（現在は256文字）に制限があります。

1. オプションのブロック削除文字。スラッシュ/です。
2. オプションの行番号。
3. 任意の数の単語、パラメーター設定、およびコメント。
4. 行末マーカー（キャリッジリターンまたはラインフィード、あるいはその両方）。

明示的に許可されていない入力はすべて違法であり、インタプリタがエラーを通知する原因になります。

スペースとタブはコード行のどこでも使用でき、コメント内を除いて行の意味を変更しません。 これにより、奇妙に見える入力が合法になります。 線G0X + 0。 12 34Y 7は、たとえばG0 x + 0.1234Y7と同等です。

入力には空白行を使用できます。 それらは無視されます。

コメントを除いて、入力では大文字と小文字が区別されません。つまり、コメントの外側の文字は、行の意味を変更せずに大文字または小文字にすることができます。

1.1.1.1 ブロック削除

一部のユーザーインターフェイスでは、オプションのブロック削除文字であるスラッシュ/を行の最初に配置すると、必要に応じてコード行をスキップできます。 Axisでは、キーの組み合わせAlt-m- /でブロック削除のオンとオフを切り替えます。 ブロック削除がスラッシュ/で始まる行にある場合、スキップされます。

4.2.2.1 行番号

行番号は、文字Nの後に符号なし整数が続き、オプションでピリオドと別の符号なし整数が続きます。 たとえば、N1234とN56.78は有効な行番号です。 通常の慣行はそのような使用を避けることですが、それらは繰り返

されるか、順不同で使用される可能性があります。 行番号もスキップされる場合があり、これは通常の方法です。 行番号を使用する必要はありませんが、使用する場合は適切な場所に配置する必要があります。

4.2.2.2 ワード

単語は、N 以外の文字の後に実際の値が続く文字です。

ワードは、次の表に示されている文字のいずれかで始まる場合があります。 上で定義したように、行番号はワードではありませんが、表には完全を期すために N が含まれています。 いくつかの文字（I、J、K、L、P、R）は、コンテキストによって意味が異なる場合があります。 軸名を参照する文字は、対応する軸がないマシンでは無効です。

アドレス	意味
A	機械の A 軸
B	機械の B 軸
C	機械の C 軸
D	工具径補正番号
F	送り速度
G	準備機能（モーダルグループの表を参照）
H	工具長オフセット番号
I	円弧および G87 固定サイクルの X オフセット
J	円弧および G87 固定サイクルの Y オフセット
K	円弧および G87 固定サイクルの Z オフセット。
	G33 同期動作のスピンドルモーション比。
L	G10、M66 などのジェネリックパラメータワード
M	その他の機能（モーダルグループの表を参照）
N	行番号
P	ドゥエルサイクルおよび G4 での滞留時間。
	G10 で使用されるキー。
Q	G73、G83 の缶詰サイクルでの送り増分
R	円弧半径または固定サイクル平面
S	主軸速度
T	工具番号
U	機械の U 軸
V	機械の V 軸
W	機械の W 軸
X	機械の X 軸
Y	機械の Y 軸
Z	機械の Z 軸

4.2.2.3 番号

次のルールは（明示的な）数値に使用されます。 これらの規則では、数字は 0 から 9 までの 1 文字です。

- 数値は、（1）オプションのプラス記号またはマイナス記号、（2）0 から数桁、場合によっては（3）小数点以下 1 桁、（4）0 から数桁で構成されます。 番号のどこかに少なくとも 1 桁。
- 数値には、整数と小数の 2 種類があります。 整数には小数点がありません。 小数はありません。
- 数字は、行の長さの制限に従い、任意の桁数にすることができます。 ただし、有効数字は約 17 桁しか保持されません（すべての既知のアプリケーションに十分です）。
- 最初の文字が正であると見なされる、符号のないゼロ以外の数値。

初期（小数点の前と最初の非ゼロ桁）と末尾（小数点と最後の非ゼロ桁の後）のゼロは許可されますが、必須ではないことに注意してください。 初期ゼロまたは後続ゼロで書き込まれた数値は、余分なゼロが存在しないかのように読み取られたときに同じ値になります。

RS274 / NGC で特定の目的に使用される数値は、多くの場合、ある有限の値のセットまたはある範囲の値に制限されています。 多くの用途では、10 進数は整数に近くなければなりません。 これには、インデックス（パラメーターやカルーセルスロット番号など）、M コード、および G コードに 10 を掛けた値が含まれます。 整数を表すことを目的とした 10 進数は、整数値の 0.0001 以内であれば、十分に近いと見なされます。

4.2.3 パラメーター

RS274 / NGC 言語は、パラメーター（他のプログラミング言語では変数と呼ばれるもの）をサポートします。目的と外観が異なるいくつかのタイプのパラメーターがあり、それぞれについて次のセクションで説明します。パラメータでサポートされている唯一の値型は浮動小数点です。 他のプログラミング言語のように、G コードには文字列、ブール、または整数型はありません。 ただし、論理式はブール演算子（AND、OR、XOR、および比較演算子 EQ、NE、GT、GE、LT、LE）を使用して定式化でき、MOD、ROUND、FUP、および FIX 演算子は整数演算をサポートします。

パラメータは、構文、スコープ、まだ初期化されていないときの動作、モード、永続性、および使用目的が異なります。

構文

構文上の外観には次の 3 種類があります。

- 番号付き-＃4711
- 名前付きローカル-＃<localvalue>
- 名前付きグローバル-＃<_ globalvalue>

範囲

パラメータのスコープは、グローバルまたはサブルーチン内のローカルのいずれかです。 サブルーチンパラメータとローカルの名前付き変数にはローカルスコープがあります。 グローバルな名前付きパラメーターと 31 番から始まる番号付きパラメーターは、スコープがグローバルです。 RS274 / NGC は、字句スコープを使用します。サブルーチンでは、そこで定義されているローカル変数のみが表示され、グローバル変数は

すべて表示されます。 呼び出し元のプロシージャのローカル変数は、呼び出されたプロシージャには表示されません。

初期化されていないパラメータの動作

- 初期化されていないグローバルパラメータ、および未使用のサブルーチンパラメータは、式で使用されると値ゼロを返します。
- 初期化されていない名前付きパラメーターは、式で使用されるとエラーを通知します。

モード

ほとんどのパラメーターは読み取り/書き込み可能であり、割り当てステートメント内で割り当てることができます。 ただし、多くの事前定義されたパラメーターの場合、これは意味がないため、読み取り専用です。式に表示される場合がありますが、代入ステートメントの左側には表示されません。

永続性

LinuxCNC がシャットダウンされると、揮発性パラメーターはその値を失います。 現在の永続範囲[1]の番号付きパラメーターを除くすべてのパラメーターは揮発性です。 永続パラメータは.var ファイルに保存され、LinuxCNC が再度起動されたときに以前の値に復元されます。 揮発性の番号付きパラメーターはゼロにリセットされます。

1)永続的なパラメータの範囲は、開発が進むにつれて変わる可能性があります。 この範囲は現在 5161～5390 です。これは、ファイル src / emc / rs274ngc /interp_array.cc の_required_parameters 配列で定義されています。

使用目的

- ユーザーパラメーター:: 31..5000 の範囲の番号付きパラメーター、および事前定義されたパラメーターを除く名前付きのグローバルパラメーターとローカルパラメーター。 これらは、プログラムの実行中、中間結果、フラグなどの浮動小数点値の汎用ストレージに使用できます。 それらは読み取り/書き込みです（値を割り当てることができます）。
- サブルーチンパラメータ-これらは、サブルーチンに渡される実際のパラメータを保持するために使用されます。
- 番号付きパラメーター-これらのほとんどは、座標系のオフセットにアクセスするために使用されます。
- システムパラメータ-現在実行中のバージョンを判別するために使用されます。 それらは読み取り専用です。

1.1.1.1 番号付きパラメーター

番号付きパラメーターは、ポンド文字＃の後に 1 から（現在）5602 [2] までの整数が続きます。パラメーターはこの整数によって参照され、その値はパラメーターに格納されている任意の数です。

値は、=演算子を使用してパラメーターに格納されます。 例えば：

2) RS274 / NGC インタープリターは、番号付きパラメーターの配列を維持します。 そのサイズは、ファイル src / emc / rs274ngc /interp_internal.hh のシンボル RS274NGC_MAX_PARAMETERS によって定義されます。 この数値パラメータの数は、開発によって新しいパラメータのサポートが追加されるにつれて増加する可能性もあります。

```
#3 = 15 (set parameter 3 to 15)
```

パラメータ設定は、同じ行のすべてのパラメータ値が見つかるまで有効になりません。 たとえば、パラメータ 3 が以前に 15 に設定されていて、線＃3 = 6 G1 X＃3 が解釈された場合、X が 15 に等しいポイントへの直線移動が発生し、パラメータ 3 の値は 6 になります。

＃文字は他の操作よりも優先されるため、たとえば、＃1 + 2 は、パラメーター 3 で見つかった値ではなく、パラメーター 1 の値に 2 を加算して見つかった数値を意味します。もちろん＃[1 + 2] パラメータ 3 で見つかった値を意味します。＃文字を繰り返すことができます。 たとえば、## 2 は、インデックスがパラメータ 2 の（整数）値であるパラメータの値を意味します。

- 31-5000-G コードのユーザーパラメータ。 これらのパラメータは G コードファイルでグローバルであり、一般的に使用できます。 揮発性。
- 5061-5069-G38 プローブ結果の座標（X、Y、Z、A、B、C、U、V、および W）。 座標は、G38 が発生した座標系にあります。 揮発性。
- 5070-G38 プローブの結果：成功した場合は 1、プローブを閉じることができなかった場合は 0。 G38.3 および G38.5 で使用されます。 揮発性。
- 5161-5169-X、Y、Z、A、B、C、U、V、W の「G28」ホーム。永続的。
- 5181-5189-X、Y、Z、A、B、C、U、V、W の「G30」ホーム。永続的。
- 5210-「G52」または「G92」オフセットが現在適用されている場合は 1、それ以外の場合は 0。 デフォルトでは揮発性。 .ini ファイルの[RS274NGC]セクションで DISABLE_G92_PERSISTENCE = 1 の場合は永続的です。
- 5211-5219-X、Y、Z、A、B、C、U、V＆W の共有「G52」および「G92」オフセット。 デフォルトでは揮発性。 .ini ファイルの[RS274NGC]セクションで DISABLE_G92_PERSISTENCE = 1 の場合は永続的です。
- 5220-G54 の座標系番号 1〜9-G59.3。 持続的に。

- 5221-5230-座標系 1、X、Y、Z、A、B、C、U、V、W、R の G54。R は Z 軸を中心とした XY 回転角を示します。 持続的に。
- 5241-5250-座標系 2、X、Y、Z、A、B、C、U、V、W、R の G55。永続的。
- 5261-5270-座標系 3、X、Y、Z、A、B、C、U、V、W、R の G56。永続的。
- 5281-5290-座標系 4、X、Y、Z、A、B、C、U、V、W、R の G57。永続的。
- 5301-5310-座標系 5、X、Y、Z、A、B、C、U、V、W、R の G58。永続的。
- 5321-5330-座標系 6、X、Y、Z、A、B、C、U、V、W、R の G59。永続的。
- 5341-5350-座標系 7、X、Y、Z、A、B、C、U、V、W、R の G59.1。永続的。
- 5361-5370-座標系 8、X、Y、Z、A、B、C、U、V、W、R の G59.2。永続的。
- 5381-5390-座標系 9、X、Y、Z、A、B、C、U、V、W、R の G59.3。永続的。
- 5399-M66 の結果-入力を確認または待機します。 揮発性。
- 5400-ツール番号。 揮発性。
- 5401-5409-X、Y、Z、A、B、C、U、V、および W のツールオフセット。揮発性。
- 5410-工具径。 揮発性。
- 5411-ツールのフロントアングル。 揮発性。
- 5412-ツールバックアングル。 揮発性。
- 5413-ツールの向き。 揮発性。
- 5420-5428-すべてのオフセットを含むアクティブな座標系および X、Y、Z、A、B、C、U、V、W の現在のプログラム単位での現在の相対位置。
- 5599-（DEBUG、）ステートメントの出力を制御するためのフラグ。 1 =出力、0 =出力なし。 デフォルト= 1。 揮発性。
- 5600-ツールチェンジャー障害インジケーター。 iocontrol-v2 コンポーネントとともに使用されます。 1：ツールチェンジャーに障害が発生しました。0：正常です。 揮発性。
- 5601-ツールチェンジャーの障害コード。 iocontrol-v2 コンポーネントとともに使用されます。 障害が発生した場合、toolchanger-reasonHAL ピンの値を反映します。 揮発性。

番号付きパラメータの永続性マシニングセンターの電源がオフになっていても、永続的な範囲のパラメータの値は時間の経過とともに保持されます。 LinuxCNC は、永続性を確保するためにパラメータファイルを使用します。 それは通訳によって管理されています。 インタプリタは、起動時にファイルを読み取り、終了時にファイルを書き込みます。

パラメータファイルの形式は、表パラメータファイル形式に示されています。

インタープリターは、ファイルに2つの列があることを想定しています。 正確に2つの数値を含まない行はスキップされます。 最初の列には整数値（パラメーターの数値）が含まれている必要があります。 2番目の列には、浮動小数点数（このパラメーターの最後の値）が含まれています。 値は、インタープリター内で倍精度浮動小数点数として表されますが、ファイルに小数点は必要ありません。

ユーザー定義の範囲（31～5000）のパラメーターをこのファイルに追加できます。 このようなパラメーターは、インタープリターによって読み取られ、ファイルが終了するときにファイルに書き込まれます。

永続範囲内の欠落しているパラメーターはゼロに初期化され、次の保存操作で現在の値で書き込まれます。

パラメータ番号は昇順で配置する必要があります。 昇順でない場合、パラメータファイルの順序が正しくないというエラーが通知されます。

元のファイルは、新しいファイルが書き込まれるときにバックアップファイルとして保存されます。

表5.3：パラメータファイル形式

パラメータ番号	パラメータ値
5161	0.0
5162	0.0

4.2.3.1 サブルーチンパラメータ

- 1-30呼び出し引数のサブルーチンローカルパラメータ。 これらのパラメーターは、サブルーチンに対してローカルです。 揮発性。 Oコードの章も参照してください。

4.2.3.2 名前付きパラメーター

名前付きパラメーターは番号付きパラメーターと同じように機能しますが、読みやすくなっています。 すべてのパラメーター名は小文字に変換され、スペースとタブが削除されているため、<param>と<P a Ram>は同じパラメーターを参照します。 名前付きパラメーターは、<>マークで囲む必要があります。

#<名前付きパラメーター>は、ローカルの名前付きパラメーターです。 デフォルトでは、名前付きパラメーターは、それが割り当てられているスコープに対してローカルです。 サブルーチンの外部でローカルパラメータにアクセスすることはできません。 これは、1つのサブルーチンが別のサブルーチンの値を上書きすることを恐れずに、2つのサブルーチンが同じパラメーター名を使用できることを意味します。

#<_グローバル名前付きパラメーター>はグローバル名前付きパラメーターです。 それらは、呼び出されたサブルーチン内からアクセス可能であり、呼び出し元がアクセス可能なサブルーチン内に値を設定できます。 スコープに関する限り、これらは通常の数値パラメーターと同じように機能します。 それらはファイルに保存されません。

例：

名前付きグローバル変数の宣言

```
#<_endmill_dia> = 0.049
```

以前に宣言されたグローバル変数への参照

```
#<_endmill_rad> = [#<_endmill_dia>/2.0]
```

リテラルと名前付きパラメーターの混合

```
o100 call [0.0] [0.0] [#<_inside_cutout>-#<_endmill_dia>] [#<_Zcut>] [#<_feedrate>]
```

名前付きパラメーターは、初めて値が割り当てられたときに発生します。 ローカルの名前付きパラメーターは、スコープが離れると消えます。サブルーチンが戻ると、そのローカルパラメーターはすべて削除され、参照できなくなります。

式内、または割り当ての右側に存在しない名前付きパラメーターを使用するとエラーになります。 存在しない名前付きパラメーターの値を DEBUG ステートメント（（DEBUG、<no_such_parameter>）のように）で出力すると、文字列＃が表示されます。

グローバルパラメータ、およびグローバルレベルで割り当てられたローカルパラメータは、プログラムが終了した場合でも一度割り当てられた値を保持し、プログラムが再度実行されたときにこれらの値を保持します。

EXISTS 関数は、指定された名前付きパラメーターが存在するかどうかをテストします。

4.2.3.3 事前定義された名前付きパラメーター

次のグローバル読み取り専用名前付きパラメーターは、インタープリターの内部状態とマシン状態にアクセスするために使用できます。 これらは、たとえば if-then-else ステートメントを使用してプログラムのフローを制御するために、任意の式で使用できます。 新しい定義済みの名前付きパラメーターは、ソースコードを変更せずに簡単に追加できることに注意してください。

- ＃<_ vmajor>-メジャーパッケージバージョン。 現在のバージョンが 2.5.2 の場合、2.5 が返されます。
- ＃<_ vminor>-マイナーパッケージバージョン。 現在のバージョンが 2.6.2 の場合、0.2 が返されます。
- ＃<_ line>-シーケンス番号。 G コードファイルを実行している場合、これは現在の行番号を返します。
- ＃<_ motion_mode>-インタプリタの現在のモーションモードを返します。

モーションモード	戻り値
G1	10

G2	20
G3	30
G33	330
G38.2	382
G38.3	383
G38.4	384
G38.5	385
G5.2	52
G73	730
G76	760
G80	800
G81	810
G82	820
G83	830
G84	840
G85	850
G86-	860
G87	870
G89	890

- #<_ plane>-現在の平面を指定する値を返します。

平面	戻り値
G17	170
G18	180
G19	190
G17.1	171
G18.1	181
G19.1	191

- #<_ ccomp>-カッター補正のステータス。 戻り値：

モード	戻り値
G40	400
G41	410
G41.1	411

G42	420
G42.1	421

- #<_ metric> -G21 がオンの場合は 1 を返し、そうでない場合は 0 を返します。
- #<_ imperial> -G20 がオンの場合は 1 を返し、そうでない場合は 0 を返します。
- #<_ absolute> -G90 がオンの場合は 1 を返し、そうでない場合は 0 を返します。
- #<_ incremental> -G91 がオンの場合は 1 を返し、そうでない場合は 0 を返します。
- #<_ inverse_time>-逆送りモード（G93）がオンの場合は 1 を返し、そうでない場合は 0 を返します。
- #<_ units_per_minute>-単位/分送りモード（G94）がオンの場合は 1 を返し、それ以外の場合は 0 を返します。
- #<_ units_per_rev>-単位/回転モード（G95）がオンの場合は 1 を返し、それ以外の場合は 0 を返します。
- #<_ coord_system>-現在の座標系名（G54..G59.3）の浮動小数点数を返します。 たとえば、G55 座標系の場合、戻り値は 550.000000 であり、G59.1 の場合、戻り値は 591.000000 です。

モード	戻り値
G54	540
G55	550
G56	560
G57	570
G58	580
G59	590
G59.1	591
G59.2	592
G59.3	593

- #<_ tool_offset>-ツールオフセット（G43）がオンの場合は 1 を返し、それ以外の場合は 0 を返します。
- #<_ retract_r_plane> -G98 が設定されている場合は 1 を返し、そうでない場合は 0 を返します。
- #<_ retract_old_z> -G99 がオンの場合は 1 を返し、そうでない場合は 0 を返します。

4.2.3.4 システムパラメータ

- #<_ spindle_rpm_mode>-スピンドル rpm モード（G97）がオンの場合は 1 を返し、そうでない場合は 0 を返します。

- #<_ spindle_css_mode>-一定の表面速度モード（G96）がオンの場合は 1 を返し、そうでない場合は 0 を返します。
- #<_ ijk_absolute_mode>-絶対円弧距離モード（G90.1）がオンの場合は 1 を返し、それ以外の場合は 0 を返します。
- #<_ lathe_diameter_mode>-これが旋盤構成で直径（G7）モードがオンの場合は 1 を返し、それ以外の場合は 0 を返します。
- #<_ lathe_radius_mode>-これが旋盤構成で半径（G8）モードがオンの場合は 1 を返し、それ以外の場合は 0 を返します。
- #<_ spindle_on>-スピンドルが現在実行中の場合（M3 または M4）は 1 を返し、それ以外の場合は 0 を返します。
- #<_ spindle_cw>-スピンドル方向が時計回り（M3）の場合は 1 を返し、それ以外の場合は 0 を返します。
- #<_ mist>-ミスト（M7）がオンの場合、1 を返します。
- #<_ Flood>-フラッド（M8）がオンの場合は 1 を返します。
- #<_ speed_override>-フィードオーバーライド（M48 または M50 P1）がオンの場合は 1 を返し、それ以外の場合は 0 を返します。
- #<_ feed_override>-フィードオーバーライド（M48 または M51 P1）がオンの場合は 1 を返し、それ以外の場合は 0 を返します。
- #<_ adaptive_feed>-アダプティブフィード（M52 または M52 P1）がオンの場合は 1 を返し、それ以外の場合は 0 を返します。
- #<_ feed_hold>-フィードホールドスイッチが有効になっている場合は 1 を返し（M53 P1）、それ以外の場合は 0 を返します。
- #<_ feed>-実際の送り速度ではなく、F の現在の値を返します。
- #<_ rpm>-実際のスピンドル速度ではなく、S の現在の値を返します。
- #<_ x>-すべてのオフセットを含む現在の相対 X 座標を返します。 #5420 と同じ。
- #<_ y>-すべてのオフセットを含む現在の相対 Y 座標を返します。 #5421 と同じ。
- #<_ z>-すべてのオフセットを含む現在の相対 Z 座標を返します。 #5422 と同じ。
- #<_ a>-すべてのオフセットを含む現在の相対 A 座標を返します。 #5423 と同じ。
- #<_ b>-すべてのオフセットを含む現在の相対 B 座標を返します。 #5424 と同じ。
- #<_ c>-すべてのオフセットを含む現在の相対 C 座標を返します。 #5425 と同じ。
- #<_ u>-すべてのオフセットを含む現在の相対 U 座標を返します。 #5426 と同じ。
- #<_ v>-すべてのオフセットを含む現在の相対 V 座標を返します。 #5427 と同じ。

- #<_ w>-すべてのオフセットを含む現在の相対 W 座標を返します。 #5428 と同じ。
- #<_ current_tool>-スピンドル内の現在のツールの番号を返します。 #5400 と同じ。
- #<_ current_pocket>-現在のツールのポケット番号を返します。
- #<_ selected_tool> -T コードを投稿する選択されたツールの番号を返します。 デフォルト-1。
- #<_ selected_pocket> -T コードを投稿する選択されたポケットの番号を返します。 デフォルト-1（ポケットは選択されていません）。
- #<_ value>-最後の O ワード return または endsub からの戻り値。 return または endsub の後に式がない場合、デフォルト値は 0 です。
- プログラム開始時に 0 に初期化されます。
- #<_ value_returned>-最後の O ワード return または endsub が値を返した場合は 1.0、それ以外の場合は 0。次の O-word 呼び出しでクリアされます。
- #<_ task>-実行中のインタプリタインスタンスが milltask の一部である場合は 1.0、それ以外の場合は 0.0。場合によっては、適切なプレビューを維持するためにこのケースを特別に処理する必要があります。たとえば、プレビューインタープリター（Axis など）で常に失敗する #5070 を検査して、プローブ（G38.n）の成功をテストする場合です。
- #<_ call_level> -O-word プロシージャの現在のネストレベル。 デバッグ用。
- #<_ remap_level>-リマップスタックの現在のレベル。 ブロック内の各リマップは、リマップレベルに 1 つ追加します。 デバッグ用。

4.2.4 HAL ピンと INI 値

INI ファイルで有効になっている場合、G コードは INI ファイルエントリと HAL ピンの値にアクセスできます。

- #<_ ini [section] name> INI ファイル内の対応するアイテムの値を返します。 たとえば、ini ファイルが次のようになっている場合：

```
[SETUP]
XPOS = 3.145
YPOS = 2.718
```

G コード内で名前付きパラメーター #<_ ini [setup] xpos>および #<_ ini [setup] ypos>を参照できます。

EXISTS を使用して、特定の ini ファイル変数の存在をテストできます。

```
o100 if [EXISTS[#<_ini[setup]xpos>]]
(debug, [setup]xpos exists: #<_ini[setup]xpos>)
o100 else
(debug, [setup]xpos does not exist)
o100 endif
```

値は inifile から 1 回読み取られ、インタープリターにキャッシュされます。 これらのパラメータは読み取り専用です。値を割り当てると、ランタイムエラーが発生します。 名前では大文字と小文字は区別されません。ini ファイルを参照する前に大文字に変換されます。

- #<_ hal [Hal item]> G コードプログラムが HAL ピンの値を読み取れるようにする可変アクセスは読み取り専用であり、G コードから HAL ピンを設定する唯一の方法は M62-M65、M67、M68 およびカスタム M100-のままです。 M199 コード。 読み取られた値はリアルタイムで更新されないことに注意してください。通常、G コードプログラムの開始時にピンにあった値が返されます。 状態の同期を強制することで、これを回避することができます。 これを行う 1 つの方法は、ダミーの M66 コマンドを使用することです：M66E0L0

例：

```
(debug, #<_hal[motion-controller.time]>)
```

HAL アイテムへのアクセスは読み取り専用です。 現在、この方法でアクセスできるのはすべて小文字の HAL 名のみです。

EXISTS は、特定の HAL アイテムの存在をテストするために使用できます。

```
o100 if [EXISTS[#<_hal[motion-controller.time]>]]
(debug, [motion-controller.time] exists: #<_hal[motion-controller.time]>)
o100 else
(debug, [motion-controller.time] does not exist)
o100 endif
```

この機能は、GladeVCP や PyVCP などのユーザーインターフェイスコンポーネント間のより強力な結合が、NGC ファイルの動作を駆動するためのパラメーターソースとして機能することを望んでいることが動機となっています。 別の方法（M6x ピンを経由してそれらを配線する）には、限定された非ニーモニック名前空間があり、UI /インタープリター通信メカニズムと同じように不必要に面倒です。

4.2.5 式

式は、左角かっこ[で始まり、バランスをとる右角かっこ]で終わる文字のセットです。 角かっこの間には、数値、パラメーター値、数学演算、およびその他の式があります。 式が評価されて数値が生成されます。 行の式は、その行が読み取られるときに、その行の何かが実行される前に評価されます。 式の例は、[1 + acos [0]-[＃3 ** [4.0 / 2]]]です。

4.2.6 二項演算子

二項演算子は、式の内部にのみ表示されます。 基本的な数学演算には、加算（+）、減算（-）、乗算（*）、除算（/）の 4 つがあります。 論理演算には、非排他的論理和（OR）、排他的論理和（XOR）、論理積（AND）の 3 つがあります。 8 番目の演算はモジュラス演算（MOD）です。 9 番目の演算は、演算の左側の数値を右側の累乗に増やす累乗演算（**）です。 関係演算子は、等式（EQ）、不等式（NE）、厳密に（GT）より大きい、（GE）以上、厳密に（LT）より小さい、および（LE）以下です。

二項演算は、優先順位に従っていくつかのグループに分けられます。 異なる優先順位グループの操作が一緒につながれている場合（たとえば、式[2.0 / 3 * 1.5-5.5 / 11.0]）、上位グループの操作は、下位グループの操作の前に実行されます。 式に同じグループの複数の操作（例の最初の/と*など）が含まれている場合は、左側の操作が最初に実行されます。 したがって、この例は次と同等です。[[[2.0 / 3] * 1.5]-[5.5 / 11.0]]、これは[1.0-0.5]、つまり 0.5 と同等です。

論理演算と絶対値は、整数だけでなく、任意の実数に対して実行されます。 ゼロという数字は論理的な偽と同等であり、ゼロ以外の数字は論理的な真と同等です。

表 5.4：演算子の優先順位

演算子	優先順位
**	高い
* / MOD	
+ -	
EQ NE GT GE LT LE	
AND OR XOR	低い

4.2.7 等式と浮動小数点値

RS274 / NGC 言語は、有限精度の浮動小数点値のみをサポートします。 したがって、2 つの浮動小数点値の等式または不等式をテストすることは本質的に問題があります。 インタプリタは、絶対差が 0.0001 未満の場合に等しい値を考慮することによってこの問題を解決します（この値は、src / emc / rs274ngc / interp_internal.hh で TOLERANCE_EQUAL として定義されています）。

4.2.8 関数

使用可能な機能を次の表に示します。 角度測度（COS、SIN、および TAN）をとる単項演算の引数は度単位です。 角度メジャー（ACOS、ASIN、および ATAN）を返す単項演算によって返される値も度単位です。

表 5.5：関数

関数名	演算結果
ATAN[arg]/[arg]	4 象限逆正接
ABS [arg]	絶対値
ACOS [arg]	逆余弦
ASIN [arg]	逆正弦
COS [arg]	余弦
EXP [arg]	e は与えられた力に上げられます
FIX [arg]	整数に切り捨てます
FUP [arg]	整数に切り上げ
ROUND[arg]	最も近い整数に丸める
LN [arg]	基数 e の対数

SIN [arg]	正弦
SQRT [arg]	平方根
TAN [arg]	正接
EXISTS [arg]	名前付きパラメータを確認してください

FIX 関数は、数直線上で左に向かって丸められるため（正の値が小さい、または負の値が大きい）、FIX [2.8] = 2 および FIX [-2.8] = -3 になります。

FUP 演算は、数直線上で右に向かって丸められます（正または負が少なくなります）。 FUP [2.8] = 3 および FUP [-2.8] = -2。

EXISTS 関数は、単一の名前付きパラメーターの存在をチェックします。 名前付きパラメーターを 1 つだけ受け取り、存在する場合は 1 を返し、存在しない場合は 0 を返します。 番号付きのパラメータまたは式を使用するとエラーになります。 EXISTS 関数の使用例を次に示します。

```
o<test> sub
o10 if [EXISTS[#<_global>]]
(debug, _global exists and has the value #<_global>)
o10 else
(debug, _global does not exist)
o10 endif
o<test> endsub
o<test> call
#<_global> = 4711
o<test> call
m2
```

4.2.9 繰り返しアイテム

行には任意の数の G ワードを含めることができますが、同じモーダルグループの 2 つの G ワードが同じ行に表示されない場合があります。詳細については、モーダルグループのセクションを参照してください。

行には、0〜4 個の M ワードが含まれる場合があります。 同じモーダルグループからの 2 つの M 単語は、同じ行に表示されない場合があります。 他のすべての合法的な文字の場合、1 行にその文字で始まる単語を 1 つだけ含めることができます。

同じパラメータのパラメータ設定が 1 行で繰り返される場合、たとえば、＃3 = 15＃3 = 6 の場合、最後の設定のみが有効になります。 同じ行に同じパラメータを 2 回設定するのはばかげていますが、違法ではありません。

1 行に複数のコメントが表示される場合は、最後のコメントのみが使用されます。 他の各コメントが読み取られ、その形式がチェックされますが、それ以降は無視されます。 1 行に複数のコメントを付けることは非常にまれであると予想されます。

4.2.10 アイテムの注文

（このセクションの冒頭で説明したように）1 行で順序が異なる可能性のある 3 種類の項目は、単語、パラメーター設定、およびコメントです。 これらの 3 つのタイプのアイテムがタイプごとに 3 つのグループに分けられていると想像してください。

最初のグループ（単語）は、行の意味を変更せずに、任意の方法で並べ替えることができます。

2 番目のグループ（パラメーター設定）を並べ替えても、同じパラメーターを複数回設定しない限り、行の意味は変わりません。 この場合、パラメータの最後の設定のみが有効になります。 たとえば、行＃3 = 15＃3 = 6 が解釈された後、パラメーター 3 の値は 6 になります。順序が＃3 = 6＃3 = 15 に逆になり、行が解釈された場合、パラメータ 3 は 15 になります。

3 番目のグループ（コメント）に複数のコメントが含まれていて並べ替えられている場合は、最後のコメントのみが使用されます。

各グループが行の意味を変更せずに順序付けまたは並べ替えられている場合、3 つのグループは行の意味を変更せずに任意の方法でインターリーブできます。 たとえば、行 g40 g1＃3 = 15（foo）＃4 = -7.0 には 5 つの項目があり、120 の可能な注文のいずれでもまったく同じことを意味します（＃4 = -7.0 g1＃3 = 15 g40（ foo））5 つのアイテム。

4.2.11 コマンドとマシンモード

多くのコマンドにより、コントローラーは 1 つのモードから別のモードに変更され、他のコマンドが暗黙的または明示的に変更するまで、モードはアクティブのままになります。 このようなコマンドはモーダルと呼ばれます。 たとえば、クーラントがオンになっている場合、明示的にオフになるまでオンのままになります。 モーションの G コードもモーダルです。 たとえば、G1（直線移動）コマンドが 1 行で指定された場合、その行で 1 つ以上の軸ワードが使用可能であれば、次の行で明示的なコマンドが指定されていない限り、次の行で再度実行されます。 軸ワードまたはキャンセルモーション。

非モーダルコードは、それらが発生する行にのみ影響します。 たとえば、G4（ドウェル）は非モーダルです。

4.2.12 極座標

極座標を使用して、移動の XY 座標を指定できます。 @n は距離、^n は角度です。 これの利点は、ボルト穴の円のように、円の中心の点に移動し、オフセットを設定してから最初の穴に移動してからドリルサイクルを実行するだけで実行できることです。 極座標は常に現在の XY ゼロ位置からのものです。 極座標をマシンゼロからシフトするには、オフセットを 使用するか、座標系を選択します。

絶対モードでは、距離と角度は XY ゼロ位置からであり、角度は X 正軸の 0 から始まり、Z 軸を中心に CCW 方向に増加します。 コード G1 @ 1^90 は G1Y1 と同じです。

相対モードでは、距離と角度も XY ゼロ位置からですが、累積されます。 これは、インクリメンタルモードでこれがどのように機能するかを最初は混乱させる可能性があります。

たとえば、次のプログラムがある場合、正方形のパターンであると予想される場合があります。

```
F100 G1 @.5 ^90
G91 @.5 ^90
@.5 ^90
@.5 ^90
@.5 ^90
G90 G0 X0 Y0 M2
```

次の図から、出力が期待したものではないことがわかります。 XY ゼロ位置からの距離が線ごとに増加するたびに距離に 0.5 を追加したためです。

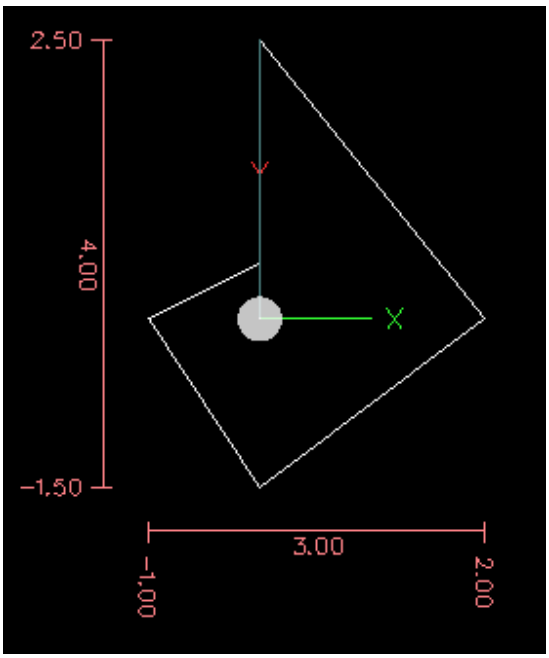


図 5-50

次のコードは、正方形のパターンを生成します。

```
F100 G1 @.5 ^90
G91 ^90
^90
^90
^90
G90 G0 X0 Y0 M2
```

ご覧のとおり、終点の距離が各線で同じになるたびに、角度に 90 度を追加するだけです。

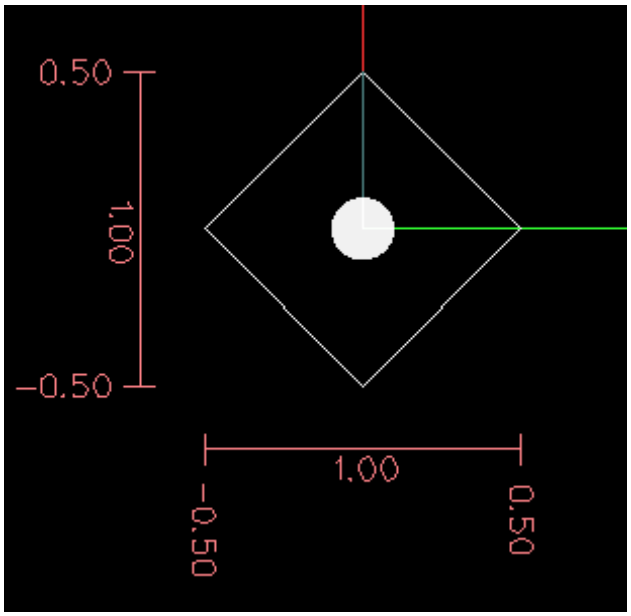


図 5-51

次の場合はエラーになります。

- 原点からインクリメンタル移動を開始
- Polar と X または Y の単語が混在して使用されている

4.2.13 モーダルグループ

モーダルコマンドはモーダルグループと呼ばれるセットに配置され、モーダルグループの 1 つのメンバーのみが常に有効になります。 一般に、モーダルグループには、2 つのメンバーを同時に有効にすることが論理的に不可能なコマンドが含まれています。たとえば、インチ単位の測定とミリメートル単位の測定などです。 マシニングセンターは同時に多くのモードになり、各モーダルグループから 1 つのモードが有効になります。 モーダルグループを次の表に示します。

表 5.6：G コードモーダルグループ

モーダルグループの意味	メンバーのワード
非モーダルコード（グループ 0）	G4、G10 G28、G30、G52、G53、G92、G92.1、G92.2、G92.3、
モーション（グループ 1）	G0、G1、G2、G3、G33、G38.n、G73、G76、G80、G81
	G82、G83、G84、G85、G86、G87、G88、G89
平面の選択（グループ 2）	G17、G18、G19、G17.1、G18.1、G19.1
距離モード（グループ 3）	G90、G91
アーク IJK 距離モード（グループ 4）	G90.1、G91.1
送り速度モード（グループ 5）	G93、G94、G95
単位ト（グループ 6）	G20、G21
カッター径補正（グループ 7）	G40、G41、G42、G41.1、G42.1

工具長オフセット（グループ 8）	G43、G43.1、G49
固定サイクルリターンモード（グループ 10）	G98、G99
座標系（グループ 12）	G54、G55、G56、G57、G58、G59、G59.1、G59.2、G59.3
制御モード（グループ 13）	G61、G61.1、G64
スピンドル速度モード（グループ 14）	G96、G97
旋盤径モード（グループ 15）	G7、G8

表 5.7：M コードモーダルグループ

モーダルグループの意味	メンバーのワード
停止（グループ 4）	M0、M1、M2、M30、M60
スピンドル（グループ 7）	M3、M4、M5
クーラント（グループ 8）	（M7 M8 は両方ともオンにすることができます）、M9
オーバーライドスイッチ（グループ 9）	M48、M49
ユーザー定義（グループ 10）	M100-M199

いくつかのモーダルグループでは、マシニングセンターがコマンドを受け入れる準備ができている場合、グループの 1 つのメンバーが有効になっている必要があります。 これらのモーダルグループにはデフォルト設定があります。 マシニングセンターの電源を入れるか、再初期化すると、デフォルト値が自動的に有効になります。

表の最初のグループであるグループ 1 は、モーション用の G コードのグループです。 これらの 1 つは常に有効です。 これを現在のモーションモードと呼びます。

グループ 1 の G コードとグループ 0 の G コードの両方が軸ワードを使用している場合、それらを同じ行に配置するとエラーになります。 グループ 1 の軸ワードを使用する G コードが（前の行でアクティブ化されているために）行に暗黙的に有効であり、軸ワードを使用するグループ 0 G コードが行に表示される場合、 グループ 1 の G コードはその回線で一時停止されています。 グループ 0 の軸ワードを使用する G コードは、
G10、G28、G30、G52、および G92 です。

O-フロー制御の行に無関係な単語を含めるとエラーになります。

4.2.14 コメント

プログラマーの意図を明確にするために、G コードの行にコメントを追加できます。 コメントは、括弧（）を使用して行に埋め込むことも、セミコロンを使用して行の残りの部分に埋め込むこともできます。 セミコロンは、括弧で囲まれている場合、コメントの開始として扱われません。

コメントは単語の間に表示される場合がありますが、単語とそれに対応するパラメーターの間に表示されることはありません。 したがって、S（設定速度）F200（送り）は OK ですが、S（速度）100F（送り）は OK ではありません。

```
G0 (Rapid to start) X1 Y1
G0 X1 Y1 (Rapid to start; but don't forget the coolant)
M2 ; End of program.
```

コメントのように見えますが、（debug、..）や（print、..）などのアクションを引き起こすアクティブなコメントがいくつかあります。 1 行に複数のコメントがある場合、最後のコメントのみがこれらのルールに従って解釈されます。 したがって、アクティブなコメントに続く通常のコメントは、アクティブなコメントを事実上無効にします。 たとえば、（foo）（debug、＃1）はパラメーター＃1 の値を出力しますが、（debug、＃1）（foo）は出力しません。

セミコロンによって導入されたコメントは、定義上、その行の最後のコメントであり、常にアクティブなコメント構文として解釈されます。

Note

O ワードのインラインコメントは使用しないでください。詳細については、O コードのコメントセクションを参照してください。

4.2.15 メッセージ

- （MSG、）-MSG が左括弧の後、他の印刷文字の前に表示される場合、メッセージを表示します。 空白と小文字を含む MSG のバリアントが許可されます。 右括弧の前の残りの文字はメッセージと見なされます。 メッセージが提供されている場合は、ユーザーインターフェイスのメッセージ表示デバイスに表示する必要があります。

メッセージの例

```
(MSG, This is a message)
```

4.2.16 プローブロギング

- （PROBEOPEN filename.txt）-filename.txt を開き、成功した各ストレートプローブの XYZABCUVW で構成される 9 桁の座標を格納します。
- （PROBECLOSE）-開いているプローブログファイルを閉じます。 プロービングの詳細については、G38 セクションを参照してください。

4.2.17 ロギング

- （LOGOPEN、filename.txt）-指定されたログファイルを開きます。 ファイルがすでに存在する場合は、切り捨てられます。
- （LOGAPPEND、filename）-指定されたログファイルを開きます。 ファイルがすでに存在する場合は、データが追加されます。
- （LOGCLOSE）-開いているログファイルを閉じます。
- （LOG、）-が開いている場合、を超えるすべてのものがログファイルに書き込まれます。 以下に説明するパラメータの拡張をサポートします。

ロギングの例は、nc_files / examples /smartprobe.ngc および nc_files / ngcgui_lib /rectange_probe.ngc サンプル G コードファイルにあります。

4.2.18 デバッグメッセージ

- （DEBUG、）-（MSG、）のようなメッセージを表示し、以下に説明するようにコメントパラメータの特別な処理を追加します。

4.2.19 メッセージを印刷する

- （PRINT、）-メッセージは、以下に説明するように、コメントパラメータの特別な処理とともに stderr に出力されます。

4.2.20 コメントパラメータ

DEBUG、PRINT、および LOG コメントでは、メッセージ内のパラメーターの値が展開されます。

例：名前付きグローバル変数を stderr（デフォルトのコンソールウィンドウ）に出力します。

パラメータの例

```
(print,endmill dia = #<_endmill_dia>)
(print,value of variable 123 is: #123)
```

上記のタイプのコメント内では、＃123 のようなシーケンスはパラメーター 123 の値に置き換えられます。＃<namedparameter>のようなシーケンスは named パラメーターの値に置き換えられます。 名前付きパラメーターでは、空白が削除されます。 したがって、＃<名前付きパラメーター>は＃<名前付きパラメーター>に変換されます。

4.2.21 ファイル要件

G コードファイルには、1 行以上の G コードが含まれ、プログラム終了で終了する必要があります。 プログラムの終わりを過ぎた G コードは評価されません。

プログラム終了コードが使用されていない場合は、ファイルの最初の行に最初のパーセント記号があり、その後に1行以上のGコードと2番目のパーセント記号が続くパーセント記号%のペアが使用されます。 2番目のパーセント記号を超えるコードは評価されません。

%を使用してGコードファイルをラップしても、プログラムの終了を使用するのと同じことはできません。 マシンは、プログラムが%を使用して残した状態になり、スピンドルとクーラントがまだオンになっている可能性があり、G90 / 91などが最後のプログラムで設定されたままになります。 適切な前文を使用しないと、次のプログラムが危険な状態で開始される可能性があります。

Note

ファイルは、Open Office Word Processorのようなワードプロセッサではなく、Geditのようなテキストエディタで作成する必要があります。

4.2.22 ファイルサイズ

インタプリタとタスクは注意深く書かれているので、パートプログラムのサイズの制限はディスク容量だけです。 ただし、TkLinuxCNCとAxisインターフェイスはどちらもプログラムテキストをロードしてユーザーに表示するため、RAMが制限要因になります。 Axisでは、プレビュープロットはデフォルトで描画されるため、再描画時間もプログラムサイズの実際的な制限になります。 Axisでプレビューをオフにして、大部分のプログラムのロードを高速化できます。 軸では、プレビューコントロールコメントを使用してプレビューのセクションをオフにすることができます。

4.2.23 Gコードの実行順序

行上のアイテムの実行順序は、行上の各アイテムの位置ではなく、次のリストによって定義されます。

- O-wordコマンド（オプションでコメントが続きますが、同じ行に他の単語は許可されません）
- コメント（メッセージを含む）
- 送り速度モード（G93、G94）を設定します。
- 送り速度（F）を設定します。
- スピンドル速度（S）を設定します。
- ツール（T）を選択します。
- HALピンI / O（M62-M68）。
- 工具交換（M6）と工具番号設定（M61）。
- スピンドルのオンまたはオフ（M3、M4、M5）。
- 状態の保存（M70、M73）、状態の復元（M72）、状態の無効化（M71）。

- クーラントのオンまたはオフ（M7、M8、M9）。
- オーバーライドを有効または無効にします（M48、M49、M50、M51、M52、M53）。
- ユーザー定義コマンド（M100-M199）。
- ドゥエル（G4）。
- アクティブプレーン（G17、G18、G19）を設定します。
- 長さの単位（G20、G21）を設定します。
- カッター半径補正のオンまたはオフ（G40、G41、G42）
- カッター長さ補正のオン/オフ（G43、G49）
- 座標系の選択（G54、G55、G56、G57、G58、G59、G59.1、G59.2、G59.3）。
- パス制御モードの設定（G61、G61.1、G64）
- 距離モード（G90、G91）を設定します。
- リトラクトモード（G98、G99）を設定します。
- 基準位置（G28、G30）に移動するか、座標系データ（G10）を変更するか、軸オフセット（G52、G92、G92.1、G92.2、G94）を設定します。
- G53 によって（おそらく）変更されたモーション（G0 から G3、G33、G38.n、G73、G76、G80 から G89）を実行します。
- 停止（M0、M1、M2、M30、M60）。

4.2.24 G コードのベストプラクティス

適切な小数精度を使用するミリメートル単位でフライス盤を作成する場合は、小数点以下 3 桁以上を使用し、インチ単位でフライス盤を作成する場合は、小数点以下 4 桁以上を使用します。

一貫した空白を使用する G コードは、単語の前に少なくとも 1 つのスペースが表示されている場合に最も読みやすくなります。 数字の途中に空白を挿入することは許可されていますが、そうする理由はありません。

センターフォーマットのアークを使用するセンターフォーマットのアーク（R-の代わりに I- J- K-を使用）は、特に 180 度または 360 度に近い夾角の場合、R フォーマットのアークよりも一貫して動作します。

プリアンブルセットのモーダルグループを使用するプログラムの正しい実行がモーダル設定に依存する場合は、必ずパートプログラムの最初に設定してください。 モードは、以前のプログラムおよび MDI コマンドから引き継ぐことができます。

工場の前文の例

```
G17 G20 G40 G49 G54 G80 G90 G94
```

G17 は XY 平面を使用、G20 インチモード、G40 は直径補正をキャンセル、G49 は長さオフセットをキャンセル、G54 は座標系 1 を使用、G80 は固定サイクルをキャンセル、G90 絶対距離モード、G94 送り/分モードを使用します。

おそらく最も重要なモーダル設定は距離の単位です。G20 または G21 を含めない場合、さまざまなマシンがさまざまなスケールでプログラムをミリングします。 固定サイクルのリターンモードなど、その他の設定も重要な場合があります。

1 行に多くのことを入れないでください。セクション実行順序のすべてを無視し、代わりに少しあいまいなコード行を記述しないでください。

同じ行にパラメータを設定および使用しないでくださいセマンティクスが明確に定義されている場合でも、同じ行にパラメータを使用および設定しないでください。 変数を＃1 = [＃1 +＃2]などの新しい値に更新しても問題ありません。

行番号を使用しないでください行番号には利点がありません。 行番号がエラーメッセージで報告される場合、番号は N ワード値ではなく、ファイル内の行番号を参照します。

4.2.25 直線軸と回転軸

1 分あたりの送りモードでの F ワードの意味は、移動するように命令された軸によって異なり、除去される材料の量は送り速度だけに依存しないため、G93 逆時間を使用する方が簡単な場合があります。 必要な材料除去率を達成するためのフィードモード。

4.2.26 一般的なエラーメッセージ

- 範囲外の G コード-G99 より大きい G コードが使用されました。LinuxCNC の G コードのスコープは 0〜99 です。0〜99 のすべての数値が有効な G コードであるとは限りません。
- 不明な g コードが使用されました-LinuxCNCG コード言語の一部ではない G コードが使用されました。
- 使用するには Gx のない i、j、k ワード-i、j、k ワードは G コードと同じ行で使用する必要があります。
- 軸値は、それらを使用する g コードなしでは使用できません-軸値は、有効なモーダル G コードまたは同じ行の G コードがない行では使用できません。
- ファイルがパーセント記号またはプログラム終了なしで終了した-すべての G コードファイルは、M2 または M30 で終了するか、パーセント記号%でラップする必要があります。

4.3 G コード

4.3.1 コンベンション

このセクションで使用される規則

G コードのプロトタイプでは、ハイフン（-）は実際の値を表し、（<>）はオプションの項目を表します。

L-がプロトタイプで書かれている場合、-は L 番号と呼ばれることが多く、他の文字についても同様です。

G コードのプロトタイプでは、axes という単語は構成で定義されている任意の軸を表します。

オプションの値はこの<L->のように記述されます。

実際の値は次のとおりです。

- 明示的な番号、4
- 式、[2 + 2]
- パラメータ値、＃88
- 単項関数値 acos [0]

ほとんどの場合、軸ワードが指定されている場合（X Y Z A B C U V W のいずれかまたはすべて）、宛先ポイントを指定します。

軸番号は、絶対座標系であると明示的に説明されていない限り、現在アクティブな座標系にあります。

軸ワードがオプションの場合、省略された軸は元の値を保持します。

オプションとして明示的に説明されていない G コードプロトタイプの項目はすべて必須です。

文字に続く値は、多くの場合、明示的な数値として示されます。 特に明記されていない限り、明示的な数値は実数値にすることができます。 たとえば、G10L2 は G [2 * 5] L [1 +1]と書くこともできます。 パラメータ 100 の値が 2 の場合、G10 L＃100 も同じ意味になります。

L-がプロトタイプで書かれている場合、-は L 番号と呼ばれることが多く、他の文字についても同様です。

4.3.2 G コードクイックリファレンステーブル

コード	説明
G0	高速での協調運動
G1	送り速度での協調運動
G2,G3	送り速度での調整されたらせん運動
G4	ドゥエル
G5	キュービックスプライン
G5.1	二次 B スプライン

G5.2	NURBS、コントロールポイントを追加
G7	直径モード（旋盤）
G8	半径モード（旋盤）
G10 L1	ツールテーブルエントリの設定
G10 L10	セットツールテーブル、計算済み、ワークピース
G10 L11	ツールテーブル、計算、フィクスチャを設定します
G10 L2	座標系の原点設定
G10 L20	計算された座標系の原点設定
G17-G19.1	平面選択
G20 G21	測定単位を設定する
G28-G28.1	事前定義された位置に移動
G30-G30.1	事前定義された位置に移動
G33	スピンドル同期モーション
G33.1	リジッドタッピング
G38.2-G38.5	プロービング
G40	カッター補正をキャンセルする
G41 G42	カッター補正
G41.1 G42.1	動的カッター補正
G43	工具テーブルからの工具長オフセットを使用
G43.1	動的工具長オフセット
G43.2	追加の工具長オフセットを適用する
G49	工具長オフセットのキャンセル
G52	ローカル座標系オフセット
G53	機械座標で移動
G54-G59.3	座標系の選択（1〜9）
G61	正確なパスモード
G61.1	正確な停止モード
G64	オプションの公差を備えたパス制御モード
G73	切りくず処理を伴う穴あけサイクル
G74	ドウェルを使用した左側のタッピングサイクル
G76	マルチパススレッディングサイクル（旋盤）
G80	モーションモードをキャンセルする
G81	ドリルサイクル
G82	ドゥエル付きドリルサイクル
G83	ペック付きドリルサイクル
G84	ドウェルを使用した右側のタッピングサイクル
G85	ボーリングサイクル、ドゥエル無し、待避付き
G86	ボーリングサイクル、停止、早送り待避
G89	ボーリングサイクル、ドゥエル、待避付き

G90 G91	距離モード
G90.1　G91.1	アーク距離モード
G92	座標系オフセット
G92.1 G92.2	G92 オフセットをキャンセルする
G92.3	G92 オフセットを復元
G93 G94 G95	フィードモード
G96	スピンドル制御モード
G98 G99	缶詰サイクル Z リトラクトモード

4.3.3 G0 早送り

```
G0 axes
```

早送りの場合は、G0 軸をプログラムします。ここで、すべての軸ワードはオプションです。 現在のモーションモードが G0 の場合、G0 はオプションです。 これにより、目的のポイントへの協調モーションが最大の高速（または低速）で生成されます。 G0 は通常、位置決め移動として使用されます。

1.1.1.1

ini ファイル[TRAJ]セクションの MAX_VELOCITY 設定は、最大早送り速度を定義します。 最大早送り速度は、協調移動中に個々の軸の MAX_VELOCITY 設定よりも高くなる可能性があります。 軸の MAX_VELOCITY または軌道の制約によって制限されている場合、最大早送り速度は[TRAJ]セクションの MAX_VELOCITY 設定よりも遅くなる可能性があります。

G0 の例

```
G90 (set absolute distance mode)
G0 X1 Y-2.3 (Rapid linear move from current location to X1 Y-2.3)
M2 (end program)
```

- 詳細については、G90 および M2 のセクションを参照してください。

カッター補正がアクティブな場合、モーションは上記とは異なります。 カッター補正のセクションを参照してください。

G53 が同じラインにプログラムされている場合、モーションも異なります。 詳細については、G53 セクションを参照してください。

G0 の高速モーションのパスは、方向の変更時に丸めることができ、軌道制御設定と軸の最大加速度によって異なります。

次の場合はエラーになります。

- 軸文字には実際の値がありません。
- 構成されていない軸文字が使用されている

4.3.4 G1 リニアムーブ

G1 axes

プログラムされた送り速度（切削の有無にかかわらず）での直線（直線）モーションの場合、G1 の「軸」をプログラムします。ここで、すべての軸ワードはオプションです。 現在のモーションモードが G1 の場合、G1 はオプションです。 これにより、現在の送り速度（またはそれより遅い速度）で目的地への協調モーションが生成されます。

G1 の例

```
G90 (set absolute distance mode)
G1 X1.2 Y-3 F10 (linear move at a feed rate of 10 from current position to X1.2 Y-3)
Z-2.3 (linear move at same feed rate from current position to Z-2.3)
Z1 F25 (linear move at a feed rate of 25 from current position to Z1)
M2 (end program)
```

- 詳細については、G90＆F＆M2 のセクションを参照してください。

カッター補正がアクティブな場合、モーションは上記とは異なります。 カッター補正のセクションを参照してください。

G53 が同じラインにプログラムされている場合、モーションも異なります。 詳細については、G53 セクションを参照してください。 次の場合はエラーになります。

- 送り速度は設定されていません。
- 軸文字には実際の値がありません。
- 構成されていない軸文字が使用されている

4.3.5 G2、G3 アークムーブ

G2 or G3 axes offsets (center format)
G2 or G3 axes R- (radius format)
G2 or G3 offsets|R- <P-> (full circles)

円弧またはらせん円弧は、現在の送り速度で G2（時計回りの円弧）または G3（反時計回りの円弧）のいずれかを使用して指定されます。 方向（CW、CCW）は、円運動が発生する軸の正の端から見たものです。

円またはらせんの軸は、機械座標系の X、Y、または Z 軸に平行である必要があります。 軸（または、同等に、軸に垂直な平面）は、G17（Z 軸、XY 平面）、G18（Y 軸、XZ 平面）、または G19（X 軸、YZ 平面）で選択されます。 。 プレーン 17.1、18.1、および 19.1 は現在サポートされていません。 円弧が円形の場合、選択した平面に平行な平面にあります。

らせんをプログラムするには、円弧平面に垂直な軸ワードを含めます。たとえば、G17 平面にある場合は、Z ワードを含めます。 これにより、XY 円運動中に、Z 軸がプログラムされた値に移動します。

複数のフルターンを与えるアークをプログラムするには、フルターン数とプログラムされたアークを指定する P ワードを使用します。 P ワードは整数でなければなりません。 P が指定されていない場合、動作は P1 が与えられたかのようになります。つまり、完全または部分的なターンが 1 回だけ発生します。 たとえば、180 度の円弧が P2 でプログラムされている場合、結果のモーションは 11/2 回転になります。 1 を超える P 増分ごとに、プログラムされた円弧に追加の完全な円が追加されます。 マルチターンヘリカルアークがサポートされており、穴やねじ山のフライス盤に役立つモーションを提供します。

警告

らせんのピッチが非常に小さい場合（ナイーブな CAM 許容値よりも小さい場合、らせんは直線に変換される可能性があります。バグ＃222

コード行が円弧を描き、回転軸の動きが含まれている場合、回転軸は一定の速度で回転するため、XYZ の動きが開始および終了すると、回転の動きが開始および終了します。 この種の行はほとんどプログラムされていません。

カッター補正がアクティブな場合、モーションは上記とは異なります。 カッター補正のセクションを参照してください。

円弧の中心は、それぞれ G90.1 または G91.1 で設定されている絶対値または相対値です。

円弧を指定するには、中心形式と半径形式の 2 つの形式を使用できます。

次の場合はエラーになります。

- 送り速度は設定されていません。
- P ワードは整数ではありません。

1.1.1.1 センターフォーマットアーク

中心フォーマットの円弧は、半径形式の円弧よりも正確であり、使用するのに適した形式です。

円弧の終点と現在の位置からの円弧の中心へのオフセットは、完全な円よりも小さい円弧をプログラムするために使用されます。 円弧の終点が現在の位置と同じであれば問題ありません。

現在の位置から円弧の中心までのオフセットと、オプションで回転数を使用して、完全な円をプログラムします。

アークをプログラミングする場合、インチの場合は小数点以下 4 桁（0.0000）未満、ミリメートルの場合は小数点以下 3 桁（0.000）未満の精度を使用すると、丸めによるエラーが発生する可能性があります。

インクリメンタルアーク距離モードアーク中心オフセットは、アークの開始位置からの相対距離です。 インクリメンタルアーク距離モードがデフォルトです。

360 度未満の円弧に対して、1 つ以上の軸ワードと 1 つ以上のオフセットをプログラムする必要があります。

軸ワードおよび 1 つ以上のオフセットを完全な円にプログラムする必要はありません。 P ワードのデフォルトは 1 で、オプションです。

'インクリメンタルアーク距離モードの詳細については、G91.1 セクションを参照してください。

絶対円弧距離モード円弧中心オフセットは、軸の現在の 0 位置からの絶対距離です。 1 つまたは複数の軸ワードと両方のオフセットは、360 度未満の円弧用にプログラムする必要があります。

軸ワードはなく、両方のオフセットを完全な円にプログラムする必要があります。 P ワードのデフォルトは 1 で、オプションです。

'絶対アーク距離モードの詳細については、G90.1 セクションを参照してください。

XY 平面（G17）

```
G2 or G3 <X- Y- Z- I- J- P->
```

- Z-らせん
- I-X オフセット
- J-Y オフセット
- •P-ターン数

XZ 平面（G18）

```
G2 or G3 <X- Z- Y- I- K- P->
```

- Y-らせん
- I-X オフセット
- K-Z オフセット
- P-ターン数

YZ 平面（G19）

```
G2 or G3 <Y- Z- X- J- K- P->
```

- X-らせん
- J-Y オフセット
- K-Z オフセット
- P-ターン数

次の場合はエラーになります。

- F ワードでは送り速度は設定されていません。
- オフセットはプログラムされていません。
- 選択した平面に円弧を投影すると、現在の点から中心までの距離が、終点から中心までの距離と（.05 インチ/.5mm）または（（.0005 インチ/。 005mm）および半径の.1%）。

エラーメッセージの解読半径から弧の終わりまでは、始点までの半径とは異なります。

- 開始-現在の位置
- center-i、j、またはkワードを使用して計算された中心位置
- end-プログラムされたエンドポイント
- r1-開始位置から中心までの半径
- r2-終了位置から中心までの半径

4.3.5.1 センターフォーマットの例

手作業でアークを計算するのは難しい場合があります。 1つのオプションは、CADプログラムを使用して円弧を描画し、座標とオフセットを取得することです。 上記の許容誤差に留意してください。目的の結果を得るには、CADプログラムの精度を変更する必要がある場合があります。 別のオプションは、数式を使用して座標とオフセットを計算することです。 次の図でわかるように、現在の位置、終了位置、および円弧の中心から三角形を形成できます。

次の図では、開始位置がX0 Y0、終了位置がX1Y1であることがわかります。 円弧の中心位置はX1Y0です。 これにより、X軸が1、Y軸が0の開始位置からのオフセットが得られます。 この場合、Iオフセットのみが必要です。

G2サンプルライン

```
G0 X0 Y0
G2 X1 Y1 I1 F10 (clockwise arc in the XY planeggif
```

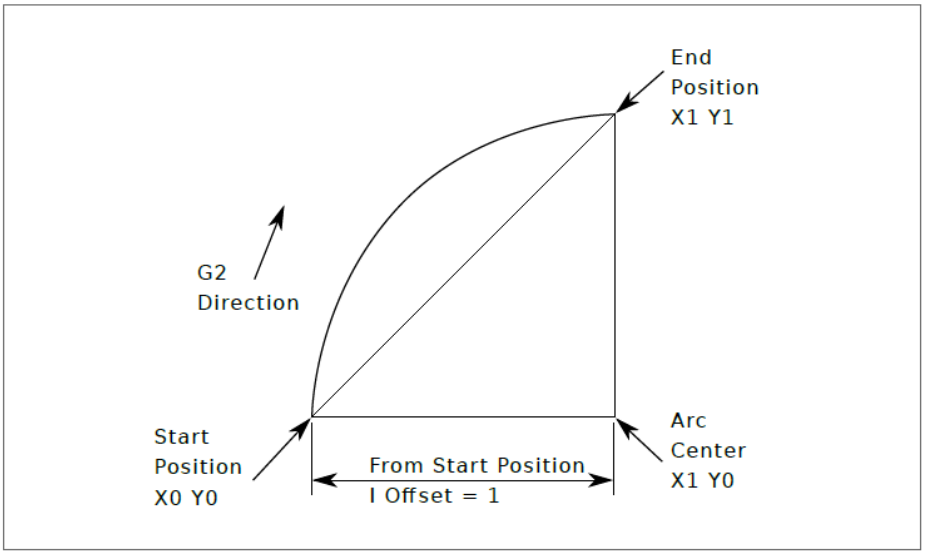


図 5-52

次の例では、G2またはG3の移動を行っている場合、Yのオフセットの違いを確認します。 G2移動の場合、開始位置はX0 Y0であり、G3移動の場合、開始位置はX0Y1です。 アークの中心は、両方の動きでX1Y0.5にあります。 G2のJオフセットの移動は0.5で、G3のJオフセットの移動は-0.5です。

G2-G3 サンプルライン

```
G0 X0 Y0
G2 X0 Y1 I1 J0.5 F25 (clockwise arc in the XY plane)
G3 X0 Y0 I1 J-0.5 F25 (counterclockwise arc in the XY plane)
```

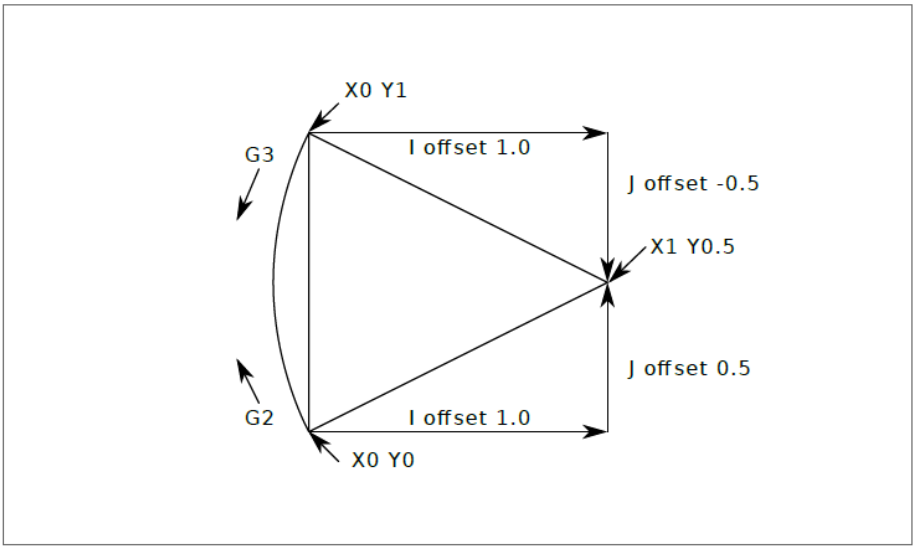


図 5-53

次の例では、Z ワードを追加することにより、円弧が Z 軸にらせんを作成する方法を示します。

G2 の例のらせん

```
G0 X0 Y0 Z0
G17 G2 X10 Y16 I3 J4 Z-1 (helix arc with Z added)
```

次の例では、P ワードを使用して複数のターンを行う方法を示します。

P ワードの例

```
G0 X0 Y0 Z0
G2 X0 Y1 Z-1 I1 J0.5 P2 F25
```

中心形式では、円弧の半径は指定されていませんが、円の中心から現在の点または円弧の終点までの距離として簡単に見つけることができます。

4.3.5.2 半径フォーマットアーク

```
G2 or G3 axes R- <P->
```

- R-現在の位置からの半径

ほぼ完全な円またはほぼ半円である半径形式の円弧をプログラムすることはお勧めできません。これは、終点の位置を少し変更すると、円の中心の位置が大幅に変更されるためです（したがって、 弧の真ん中）。 拡大効果

は十分に大きいため、数値の丸め誤差によって許容範囲外のカットが生成される可能性があります。 たとえば、180 度の円弧の端点の 1%の変位は、円弧に沿って 90 度の点の 7%の変位を生成しました。 ほぼ完全な円はさらに悪いです。 他のサイズの円弧（小さいから 165 度または 195 から 345 度の範囲）でも問題ありません。

半径形式では、選択した平面内の円弧の終点の座標が、円弧の半径とともに指定されます。 G2 軸 R-をプログラムします（または G2 の代わりに G3 を使用します）。 R は半径です。 軸ワードはすべてオプションですが、選択した平面の軸の 2 つのワードのうち少なくとも 1 つを使用する必要があります。 R 番号は半径です。 正の半径は、円弧の回転が 180 度未満であることを示し、負の半径は、180 度を超える回転を示します。 円弧がらせんの場合、らせんの軸に平行な座標軸上の円弧の終点の値も指定されます。

次の場合はエラーになります。

- 選択した平面の軸の両方の軸ワードが省略されます
- 円弧の終点は現在の点と同じです。

G2 サンプルライン

```
G17 G2 X10 Y15 R20 Z5 (radius format with arc)
```

上記の例では、軸が Z 軸に平行で、半径 20 の X = 10、Y = 15、Z = 5 で終わる、時計回り（正の Z 軸から見て）の円弧またはらせん状の円弧を作成します。 。Z の開始値が 5 の場合、これは XY 平面に平行な円弧です。 それ以外の場合は、らせん状の円弧です。

4.3.6 G4 ドウェル

```
G4 P-
```

- P-滞留する秒数（浮動小数点）

P 番号は、すべての軸が動かないままになる秒単位の時間です。 P 数は浮動小数点数であるため、1 秒未満を使用できます。 G4 は、スピンドル、クーラント、および I / O には影響しません。

G4 サンプルライン

```
G4 P0.5 (wait for 0.5 seconds before proceeding)
```

次の場合はエラーになります。

- P 番号が負であるか、指定されていません。

4.3.7 G5 キュービックスプライン

```
G5 X- Y- <I- J-> P- Q-
```

- I-X 開始点から最初の制御点までの増分オフセット
- J-Y 開始点から最初の制御点までの増分オフセット
- P-X エンドポイントから 2 番目のコントロールポイントへの増分オフセット

- Q-Y エンドポイントから 2 番目のコントロールポイントへの増分オフセット

G5 は、X 軸と Y 軸のみを使用して XY 平面に 3 次 B スプラインを作成します。 すべての G5 コマンドに対して P と Q の両方を指定する必要があります。

一連の G5 コマンドの最初の G5 コマンドには、I と J の両方を指定する必要があります。 後続の G5 コマンドでは、I と J の両方を指定するか、どちらも指定しない必要があります。 I と J が指定されていない場合、このキュービックの開始方向は、前のキュービックの終了方向と自動的に一致します（I と J が前の P と Q の否定であるかのように）。

たとえば、曲線の N 形状をプログラムするには、次のようにします。

G5 サンプルの初期 3 次スプライン

```
G90 G17
G0 X0 Y0
G5 I0 J3 P0 Q-3 X1 Y1
```

これにスムーズに接続する 2 番目の曲線 N は、I と J を指定せずに作成できるようになりました。

G5 後続の 3 次スプラインのサンプル

```
G5 P0 Q-3 X2 Y2
```

次の場合はエラーになります。

- P と Q の両方が指定されていない
- I または J のいずれか 1 つだけが指定されています
- I または J は、一連の G5 コマンドの最初のコマンドで指定されていません
- X または Y 以外の軸が指定されている
- アクティブな平面は G17 ではありません

4.3.8 G5.1 2 次スプライン

```
G5.1 X- Y- I- J-
```

- I-X 開始点から制御点までの増分オフセット
- J-Y 開始点から制御点までの増分オフセット

G5.1 は、X 軸と Y 軸のみを使用して XY 平面に 2 次 B スプラインを作成します。 I または J を指定しないと、指定されていない軸のオフセットがゼロになるため、一方または両方を指定する必要があります。

たとえば、原点を介して X-2Y4 から X2Y4 に放物線をプログラムするには、次のようにします。

```
G90 G17
G0 X-2 Y4
G5.1 X2 I2 J-8
```

次の場合はエラーになります。

- I と J の両方のオフセットが指定されていないかゼロです
- X または Y 以外の軸が指定されている
- X | YNEXT の軸が指定された

4.3.9 G5.2 G5.3NURBS ブロック

```
G5.2 <P-> <X- Y-> <L->
X- Y- <P->
...
G5.3
```

警告：G5.2、G5.3 は実験的なものであり、完全にはテストされていません。

G5.2 は NURBS を定義するデータブロックを開くためのものであり、G5.3 はデータブロックを閉じるためのものです。 これらの 2 つのコード間の線では、曲線の制御点は、関連する重み（P）と曲線の順序を決定するパラメーター（L）の両方で定義されます。

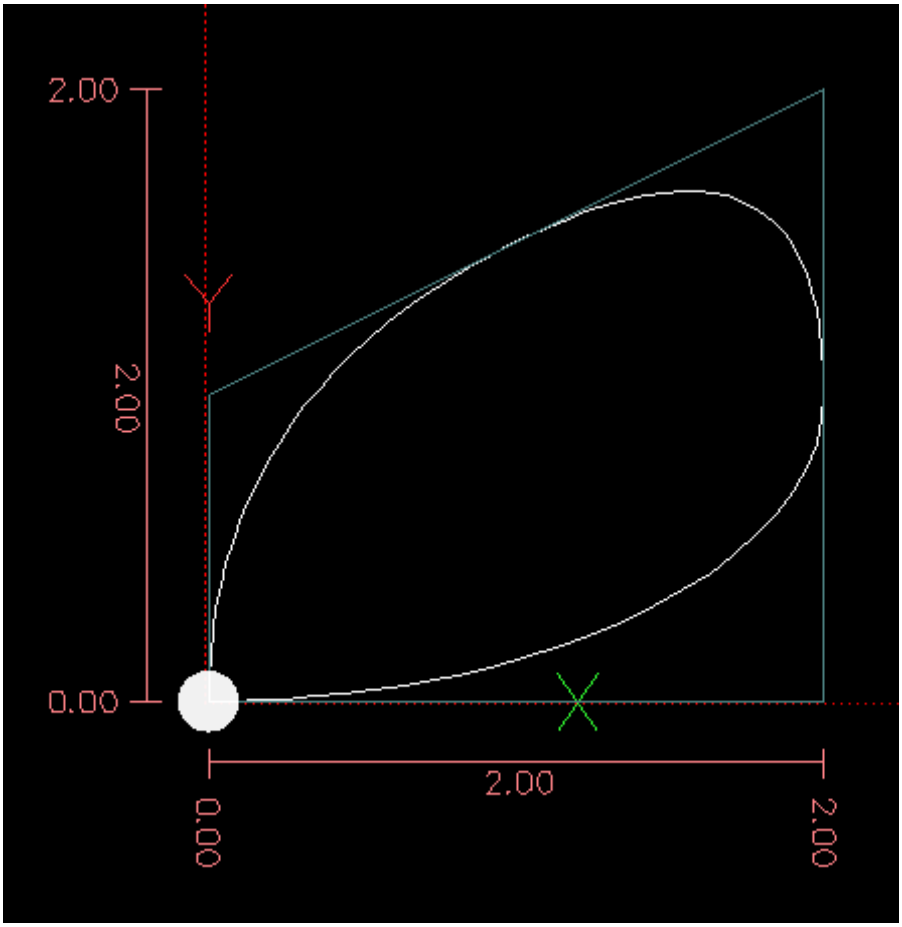
最初の G5.2 コマンドの前の現在の座標は、常に最初の NURBS コントロールポイントとして使用されます。この最初の制御点の重みを設定するには、最初に XY を指定せずに G5.2P-をプログラムします。

P が指定されていない場合のデフォルトの重みは 1 です。L が指定されていない場合のデフォルトの順序は 3 です。

G5.2 の例

```
G0 X0 Y0 (rapid move)
F10 (set feed rate)
G5.2 P1 L3
X0 Y1 P1
X2 Y2 P1
X2 Y0 P1
X0 Y0 P2
G5.3
; The rapid moves show the same path without the NURBS Block
G0 X0 Y1
X2 Y2
X2 Y0
X0 Y0
```

M2



NURBS 出力のサンプル NURBS の詳細については、次を参照してください。

http://wiki.linuxcnc.org/cgi-bin/wiki.pl?NURBS

4.3.10 G7 旋盤径モード

G7

G7 をプログラムして、旋盤の軸 X の直径モードに入ります。 直径モードの場合、X 軸は旋盤上を移動し、旋盤の中心までの距離の 1/2 になります。 たとえば、X1 は、カッターを旋盤の中心から 0.500 インチに移動し、直径 1 インチの部品を作成します。

4.3.11 G8 旋盤半径モード

G8 をプログラムして、旋盤の軸 X の半径モードに入ります。 半径モードの場合、X 軸は旋盤上を移動し、中心からの距離になります。 したがって、X1 でのカットは、直径 2 "のパーツになります。電源投入時のデフォルトは G8 です。

4.3.12 G10L1 セットツールテーブル

```
G10 L1 P- axes <R- I- J- Q->
```

- P-工具番号
- R-工具の半径
- I-フロントアングル（旋盤）
- J-バックアングル（旋盤）
- Q-オリエンテーション（旋盤）

G10 L1 は、P 工具番号の工具テーブルをワードの値に設定します。

有効な G10L1 は、ツールテーブルを書き換えてリロードします。

G10L1 サンプルライン

```
G10 L1 P1 Z1.5 (set tool 1 Z offset from the machine origin to 1.5)
G10 L1 P2 R0.015 Q3 (lathe example setting tool 2 radius to 0.015 and orientation to 3)
```

次の場合はエラーになります。

- カッター補正がオンになっています
- P 番号は指定されていません
- P 番号が工具テーブルの有効な工具番号ではありません
- P 番号は 0 です

Q ワードで使用されるカッターの向きの詳細については、旋盤工具の向きの図を参照してください。

4.3.13 G10L2 セット座標系

```
G10 L2 P- <axes R->
```

- P-座標系（0-9）
- R-Z 軸を中心とした回転

G10 L2 は、指定された座標系の軸の原点を軸ワードの値にオフセットします。 オフセットは、ホーミング中に確立された機械原点からのものです。 オフセット値は、指定された座標系で有効な現在のオフセットを置き換えます。 使用されていない軸ワードは変更されません。

P0～P9 をプログラムして、変更する座標系を指定します。

表 5.8：座標系

P 値	座標系	G コード
0	現在の座標系	n/a
1	1	G54
2	2	G55
3	3	G56
4	4	G57
5	5	G58
6	6	G59
7	7	G59.1
8	8	G59.2
9	9	G59.3

オプションで、R をプログラムして、Z 軸を中心とした XY 軸の回転を示します。 回転方向は、Z 軸の正の端から見て CCW です。

すべての軸ワードはオプションです。

インクリメンタルディスタンスモード（G91）であっても、G10L2 には影響しません。

重要な概念：

- G10 L2 Pn は、現在の座標系から P で指定された座標系に変更されません。座標系を選択するには、G54-59.3 を使用する必要があります。
- 回転が実際にジョギングしている場合、軸はその軸を正または負の方向にのみ移動し、回転した軸に沿って移動しません。
- G52 ローカルオフセットまたは G92 原点オフセットが G10L2 の前に有効であった場合、その後も引き続き有効になります。
- R を使用して座標系をプログラミングする場合、回転後に G52 または G92 が適用されます。
- G10 コマンドによって原点が設定されている座標系は、G10 の実行時にアクティブまたは非アクティブになります。 現在アクティブな場合、新しい座標はすぐに有効になります。

次の場合はエラーになります。

- P 数は、0 から 9 の範囲の整数には評価されません。
- 構成で定義されていない軸がプログラムされています。

G10L2 サンプルライン

```
G10 L2 P1 X3.5 Y17.2
```

上記の例では、最初の座標系（G54 によって選択されたもの）の原点は X = 3.5 および Y = 17.2 に設定されています。 X と Y のみが指定されているため、原点は X と Y でのみ移動します。 他の座標は変更されません。

G10L2 サンプルライン

```
G10 L2 P1 X0 Y0 Z0 (clear offsets for X,Y & Z axes in coordinate system 1)
```

上記の例では、座標系 1 の XYZ 座標を機械の原点に設定します。

座標系については、「座標系」のセクションで説明しています。

4.3.14 G10L10 セットツールテーブル

```
G10 L10 P- axes <R- I- J- Q->
```

- P-工具番号
- R-工具の半径
- I-フロントアングル（旋盤）
- J-バックアングル（旋盤）
- Q-オリエンテーション（旋盤）

G10 L10 は、工具 P の工具テーブルエントリを変更して、工具オフセットがリロードされ、マシンが現在の位置にあり、現在の G5x および G52 / G92 オフセットがアクティブな場合、指定された軸の現在の座標が指定された値になるようにします。 。 G10L10 コマンドで指定されていない軸は変更されません。 これは、G38 のセクションで説明されているプローブの移動に役立つ可能性があります。

G10L10 の例

```
T1 M6 G43 (load tool 1 and tool length offsets)
G10 L10 P1 Z1.5 (set the current position for Z to be 1.5)
G43 (reload the tool length offsets from the changed tool table)
M2 (end program)
```

- 詳細については、T&M6、および G43 /G43.1 のセクションを参照してください。

次の場合はエラーになります。

- カッター補正がオンになっています
- P 番号は指定されていません
- P 番号が工具テーブルの有効な工具番号ではありません
- P 番号は 0 です

4.3.15 G10L11 セットツールテーブル

```
G10 L11 P- axes <R- I- J- Q->
```

- P-工具番号
- R-工具の半径
- I-フロントアングル（旋盤）
- J-バックアングル（旋盤）
- Q-オリエンテーション（旋盤）

G10L11 は G10L10 と同じですが、現在のオフセットに従ってエントリを設定する代わりに、新しいツールオフセットがリロードされ、マシンが G59.3 に配置された場合に、現在の座標が指定された値になるように設定されます。 G52 / G92 オフセットがアクティブでない座標系。

これにより、ユーザーはマシン上の固定点に従って G59.3 座標系を設定し、そのフィクスチャを使用して、現在アクティブな他のオフセットに関係なくツールを測定できます。

次の場合はエラーになります。

- カッター補正がオンになっています
- P 番号は指定されていません
- P 番号が工具テーブルの有効な工具番号ではありません
- P 番号は 0 です

4.3.16 G10L20 セット座標系

```
G10 L20 P- axes
```

- P-座標系（0-9）

G10L20 は G10L2 と似ていますが、オフセット/エントリを指定された値に設定する代わりに、現在の座標が指定された値になるように計算された値に設定される点が異なります。

G10L20 サンプルライン

```
G10 L20 P1 X1.5 (set the X axis current location in coordinate system 1 to 1.5)
```

次の場合はエラーになります。

- P 数は、0 から 9 の範囲の整数には評価されません。
- 構成で定義されていない軸がプログラムされています。

4.3.17 G17-G19.1 平面選択

これらのコードは、現在の平面を次のように設定します。

- G17-XY（デフォルト）

- G18-ZX
- G19-YZ
- G17.1-UV
- G18.1-WU
- G19.1-VW

UV、WU、VW プレーンはアークをサポートしていません。

各 G コードファイルのプリアンブルに平面選択を含めることをお勧めします。

平面を選択した場合の影響については、セクション G2G3 アークおよびセクション G81G89 で説明しています。

4.3.18 G20、G21 ユニット

- G20-長さの単位にインチを使用します。
- G21-長さの単位にミリメートルを使用します。

各 G コードファイルのプリアンブルに単位を含めることをお勧めします。

4.3.19 G28、G28.1 Go / Set Predefined Position

警告

マシンが繰り返し可能な位置にホームされ、目的の G28 位置が G28.1 に保存されている場合にのみ、G28 を使用してください。

G28 は、パラメータ 5161-5169 に格納されている値を、移動先の X Y Z A B C UVW 最終点として使用します。 パラメータ値は、ini ファイルで指定されているネイティブマシンユニットの絶対マシン座標です。 G28 が発行されると、ini ファイルで定義されているすべての軸が移動されます。 G28.1 で位置が保存されていない場合、すべての軸が機械の原点に移動します。

- G28-現在の位置からパラメータ 5161-5166 の値の絶対位置にすばやく移動します。
- G28 軸-オフセットを含む軸で指定された位置にすばやく移動し、次に、指定されたすべての軸のパラメータ 5161-5166 の値の絶対位置にすばやく移動します。 指定されていない軸は移動しません。
- G28.1-現在の絶対位置をパラメータ 5161-5166 に格納します。

G28 サンプルライン

```
G28 Z2.5 (rapid to Z2.5 then to Z location specified in #5163)
```

次の場合はエラーです：

- カッター補正がオンになっている

4.3.20 G30、G30.1 Go / Set Predefined Position

警告

G30 は、マシンが繰り返し可能な位置にホームされ、目的の G30 位置が G30.1 に保存されている場合にのみ使用してください。

G30 は G28 と同じように機能しますが、パラメータ 5181-5189 に格納されている値を、移動先の X Y Z A B C U VW 最終ポイントとして使用します。 パラメータ値は、ini ファイルで指定されているネイティブマシンユニットの絶対マシン座標です。 G30 が発行されると、ini ファイルで定義されているすべての軸が移動されます。 G30.1 で位置が保存されていない場合、すべての軸が機械の原点に移動します。

Note

TOOL_CHANGE_AT_G30 = 1 が ini ファイルの[EMCIO]セクションにある場合、M6 がプログラムされているときに、G30 パラメータを使用してツールを移動します。

- G30-現在の位置からパラメータ 5181-5189 の値の絶対位置にすばやく移動します。
- G30 軸-オフセットを含む軸で指定された位置にすばやく移動し、次に、指定されたすべての軸のパラメータ 5181-5189 の値の絶対位置にすばやく移動します。 指定されていない軸は移動しません。
- G30.1-現在の絶対位置をパラメータ 5181-5186 に格納します。

G30 サンプルライン

```
G30 Z2.5 (rapid to Z2.5 then to the Z location specified in #5i83)
```

次の場合はエラーです：

- カッター補正がオンになっている

4.3.21 G33 スピンドル同期モーション

```
G33 X- Y- Z- K- $-
```

- K-1 回転あたりの距離

一方向のスピンドル同期モーションの場合、コード G33 X- Y- Z- K-ここで、K は、スピンドルの各回転に対して XYZ で移動した距離を示します。 たとえば、Z = 0 で開始する場合、G33 Z-1 K.0625 は、スピンドルの 16 回転にわたって Z で 1 インチのモーションを生成します。 このコマンドは、16TPI スレッドを生成するプログラムの一部である可能性があります。 メートル法の別の例として、G33 Z-15 K1.5 は 15mm の動きを生成し、スピンドルは 1.5mm のねじ山に対して 10 回回転します。

（オプションの）$引数は、モーションが同期されるスピンドルを設定します（デフォルトはゼロです）。 たとえば、G33 Z10 K1 $ 1 は、スピンドルと同期してスピンドルを移動します。N.revsHAL ピン値。

スピンドル同期モーションは、スピードピンでスピンドルインデックスとスピンドルを待機するため、複数のパスが整列します。 G33 は、プログラムされたエンドポイントで終了します。 G33 は、テーパースレッドまたはヒューズの切断に使用できます。

少なくとも 1 つを使用する必要があることを除いて、すべての軸ワードはオプションです。

Note

K は、X- Y-Z-で表されるドライブラインに従います。 たとえばテーパーねじを切断するときに X または Y 端点が使用される場合、K は Z 軸に平行ではありません。

技術情報各 G33 パスの開始時に、LinuxCNC はスピンドル速度とマシン加速制限を使用して、インデックスパルス後に Z が加速するのにかかる時間を計算し、その間にスピンドルが回転する角度を決定します。 次に、その角度をインデックス位置に追加し、修正されたスピンドル角度を使用して Z 位置を計算します。 つまり、Z は適切な速度まで加速し終えるとすぐに正しい位置に到達し、すぐに適切な糸を切り始めることができます。

HAL 接続モーションを開始するには、ピンスピンドル.N.at-speed を true に設定または駆動する必要があります。 さらに、スピンドル.N.revs は、スピンドルとスピンドルの回転ごとに 1 ずつ増加する必要があります。N.index-enable ピンは、回転ごとに 1 回 index-enable をリセットするエンコーダー（またはリゾルバー）カウンターに接続する必要があります。

主軸同期動作の詳細については、インテグレータのマニュアルを参照してください。

G33 の例

```
G90 (absolute distance mode)
G0 X1 Z0.1 (rapid to position)
S100 M3 (start spindle turning)
G33 Z-2 K0.125 (move Z axis to -2 at a rate to equal 0.125 per revolution)
G0 X1.25 (rapid move tool away from work)
Z0.1 (rapid move to starting Z position)
M2 (end program)
```

- 詳細については、G90＆G0＆M2 のセクションを参照してください。

次の場合はエラーになります。

- すべての軸ワードが省略されています。
- このコマンドを実行しても、スピンドルが回転していません
- 要求された線形運動は、スピンドル速度のために機械速度制限を超えています

4.3.22 G33.1 リジッドタッピング

```
G33.1 X- Y- Z- K- I- $-
```

- K-1 回転あたりの距離
- I-より速いリターン移動のためのオプションのスピンドル速度乗数
- $-オプションのスピンドルセレクター

警告

Z の場合、G33.1 を呼び出す前に XY 位置を前置詞のみでタップし、G33.1 では Z ワードのみを使用します。指定された座標が G33.1 を呼び出して移動をタップしたときに現在の座標ではない場合、移動は Z 軸に沿っていませんが、現在の場所から指定された場所への調整されたスピンドル同期移動になります。

リジッドタッピング（リターン付きのスピンドル同期モーション）の場合、コード G33.1 X- Y- Z- K-ここで、K-はスピンドルの各回転ごとに移動した距離を示します。

リジッドタッピングムーブは、次のシーケンスで構成されます。

1. 現在の座標から指定された座標への移動。選択されたスピンドルと指定された比率で同期され、現在の座標からスピンドルインデックスパルスで開始されます。
2. エンドポイントに到達したら、スピンドルを反転し、乗数によって設定された係数で速度を上げるコマンド（時計回りから反時計回りなど）。
3. スピンドルが実際に停止して反転するまで、指定された終了座標を超えて同期モーションを継続します。
4. 元の座標に戻る同期モーションを続行しました。
5. 元の座標に達したときに、スピンドルをもう一度反転させるコマンド（たとえば、反時計回りから時計回りに）。
6. 元の座標に達されます、スピンドルを反転反転するコマンド（本当に、反時計回りから時計回りに）。
7. 非同期で元の座標に戻ります。

スピンドル同期モーションはスピンドルインデックスを待機するため、複数のパスが整列します。G33.1 の移動は元の座標で終了します。 少なくとも 1 つを使用する必要があることを除いて、すべての軸ワードはオプションです。

G33.1 例

```
G90 (set absolute mode)
G0 X1.000 Y1.000 Z0.100 (rapid move to starting position)
S100 M3 (turn on the spindle, 100 RPM)
G33.1 Z-0.750 K0.05 (rigid tap a 20 TPI thread 0.750 deep)
M2 (end program)
```

- 詳細については、G90＆G0＆M2 のセクションを参照してください。

次の場合はエラーになります。

- すべての軸ワードが省略されています。
- このコマンドを実行しても、スピンドルが回転していません
- 要求された線形運動は、スピンドル速度のために機械速度制限を超えています

4.3.23 G38.n ストレートプローブ

G38.n axes

- G38.2-ワークピースに向かってプローブし、接触で停止し、故障した場合はエラーを通知します
- G38.3-ワークピースに向かってプローブし、接触で停止します
- G38.4-プローブをワークピースから離し、接触が失われると停止し、故障した場合は信号エラー
- G38.5-プローブをワークピースから離し、接触がなくなったら停止します

重要

プローブ入力信号を提供するようにマシンが設定されるまで、プローブ移動を使用することはできません。 プローブ入力信号は、.hal ファイルの motion.probe-input に接続する必要があります。 G38.n は、motion.probe-input を使用して、プローブがいつ接触した（または接触を失った）かを判断します。 プローブ接点が閉じている（接触している）場合は TRUE、プローブ接点が開いている場合は FALSE。

ストレートプローブ操作を実行するように G38.n 軸をプログラムします。 軸ワードはオプションですが、少なくとも 1 つを使用する必要があります。 軸ワードは一緒に、現在の場所から開始して、プローブが向かって移動する宛先ポイントを定義します。 宛先に到達する前にプローブがトリップしない場合、G38.2 および G38.4 はエラーを通知します。

スピンドルの工具はプローブであるか、プローブスイッチに接触している必要があります。

このコマンドに応答して、マシンは制御されたポイント（プローブボールの中心にある必要があります）を現在の送り速度でプログラムされたポイントに向かって直線的に移動します。 逆時間送りモードでは、送り速度は、現在のポイントからプログラムされたポイントまでの全体の動きが指定された時間かかるようなものです。 プログラムされたポイントに到達したとき、またはプローブ入力で要求された変更が行われたときのいずれか早い方で、移動は（マシンの加速制限内で）停止します。

プローブが成功すると、パラメータ＃5061～＃5069 が、プローブの状態が変化したときの制御点の位置の X、Y、Z、A、B、C、U、V、W 座標に設定されます（ 現在の作業座標系）。 プロービングに失敗すると、プログラムされたポイントの座標に設定されます。 パラメータ 5070 は、プローブが成功した場合は 1 に設定され、プローブが失敗した場合は 0 に設定されます。 プローブ操作が失敗した場合、選択した GUI がそれをサポートしていれば、G38.2 および G38.4 は画面にメッセージを投稿してエラーを通知します。 そして、プログラムの実行を停止することによって。

フォーム（PROBEOPEN filename.txt）のコメントは、filename.txt を開き、成功した各ストレートプローブの XYZABCUVW で構成される 9 桁の座標を格納します。 ファイルは（PROBECLOSE）で閉じる必要があります。 詳細については、コメントセクションを参照してください。

プローブ移動を使用してパーツの座標をファイルに記録する方法を示すために、サンプルファイル smartprobe.ngc が（examples ディレクトリに）含まれています。 プログラム smartprobe.ngc は、最小限の変更で ngcgui とともに使用できます。

次の場合はエラーになります。

- 現在のポイントは、プログラムされたポイントと同じです。
- 軸ワードは使用されません
- カッター補正が有効になっている
- 送り速度はゼロです
- プローブはすでにターゲット状態にあります

4.3.24 G40 補償オフ

- G40-カッター補正をオフにします。 工具補正が次の移動であった場合は、直線移動であり、工具の直径よりも長くなければなりません。 すでにオフになっている場合は、補正をオフにしても問題ありません。

G40 の例

```
; current location is X1 after finishing cutter compensated move
G40 (turn compensation off)
G0 X1.6 (linear move longer than current cutter diameter)
M2 (end program)
```

詳細については、G0 および M2 のセクションを参照してください。

次の場合はエラーになります。

- G2 / G3 アーク移動は、G40 の次にプログラムされます。
- 補正をオフにした後の直線移動は、工具径よりも小さくなります。

4.3.25 G41、G42 カッター補正

```
G41 <D-> (left of programmed path)
G42 <D-> (right of programmed path)
```

- D-工具番号

D ワードはオプションです。 D ワードがない場合は、現在ロードされているツールの半径が使用されます（ツールがロードされておらず、D ワードが指定されていない場合、半径 0 が使用されます）。

指定されている場合、D ワードは使用する工具番号です。 これは通常、スピンドル内の工具の番号です（この場合、D ワードは冗長であり、指定する必要はありません）が、任意の有効な工具番号である可能性があります。

Note

G41 / G42D0 は少し特別です。 その動作は、ランダムツールチェンジャーマシンと非ランダムツールチェンジャーマシンで異なります（ツール変更のセクションを参照）。 非ランダム工具交換機では、G41 / G42 D0 は、現在主軸にある工具の TLO を適用し、主軸に工具がない場合は 0 の TLO を適用します。 ランダムツールチェンジャーマシンでは、G41 / G42 D0 はツールテーブルファイルで定義されたツール T0 の TLO を適用します（または T0 がツールテーブルで定義されていない場合はエラーを引き起こします）。

部品プロファイルの左側でカッター補正を開始するには、G41 を使用します。 G41 は、平面に垂直な軸の正の端から見て、プログラムされた線の左側でカッター補正を開始します。

部品プロファイルの右側でカッター補正を開始するには、G42 を使用します。 G42 は、平面に垂直な軸の正の端から見て、プログラムされた線の右側でカッター補正を開始します。

移動中のリードは、少なくとも工具半径と同じ長さである必要があります。 移動のリードは、迅速な移動である可能性があります。

XY 平面または XZ 平面がアクティブな場合、カッター補正を実行できます。

カッター補正がオンの場合、ユーザー M100-M199 コマンドが許可されます。

カッター補正がオンのときのマシニングセンターの動作は、コード例とともにカッター補正セクションで説明されています。

次の場合はエラーになります。

- D 番号は有効な工具番号または 0 ではありません。
- YZ 平面がアクティブです。
- カッター補正は、すでにオンになっているときにオンになるように命令されます。

4.3.26 G41.1、G42.1 動的カッター補正

```
G41.1 D- <L-> (left of programmed path)
G42.1 D- <L-> (right of programmed path)
```

- D-カッター直径
- L-工具の向き（旋盤の工具の向きを参照）

G41.1 および G42.1 は、G41 および G42 と同じように機能しますが、工具径をプログラムできる範囲が追加されています。 指定されていない場合、L ワードのデフォルトは 0 です。

次の場合はエラーになります。

- YZ 平面がアクティブです。
- L 番号は 0 から 9 までの範囲ではありません。
- L 番号は、XZ 平面がアクティブでない場合に使用されます。
- カッター補正は、すでにオンになっているときにオンになるように命令されます。

4.3.27 G43 工具長オフセット

```
G43 <H->
```

- H-工具番号（オプション）

G43 は工具長補正を可能にします。 G43 は、オフセットの長さだけ軸座標をオフセットすることにより、後続のモーションを変更します。 G43 は動きを引き起こしません。 次に補正された軸が移動されたとき、その軸の端点が補正された位置になります。

H ワードのない G43 は、最後の TnM6 から現在ロードされているツールを使用します。

G43 Hn は、工具 n のオフセットを使用します。

Note

G43H0 は少し特別です。 その動作は、ランダムツールチェンジャーマシンと非ランダムツールチェンジャーマシンで異なります（ツールチェンジャーのセクションを参照）。 非ランダム工具交換機では、G43 H0 は現在主軸にある工具の TLO を適用し、主軸に工具がない場合は 0 の TLO を適用します。 ランダム工具交換機では、G43 H0 は工具テーブルファイルで定義された工具 T0 の TLO を適用します（または T0 が工具テーブルで定義されていない場合はエラーを引き起こします）。

G43H-ライン例

```
G43 H1 (set tool offsets using the values from tool 1 in the tool table)
```

次の場合はエラーになります。

- H 番号が整数ではない、または
- H 数が負、または
- H 番号は有効な工具番号ではありません（0 は非ランダム工具交換機では有効な工具番号ですが、「現在スピンドルにある工具」を意味します）。

4.3.28 G43.1 動的工具長オフセット

```
G43.1 axes
```

- G43.1 軸-軸の現在のオフセットを置き換えることにより、後続のモーションを変更します。 G43.1 は動きを引き起こしません。 次に補正された軸が移動されたとき、その軸の端点が補正された位置になります。

G43.1 例

```
G90 (set absolute mode)
T1 M6 G43 (load tool 1 and tool length offsets, Z is at machine 0 and DRO shows Z1.500)
G43.1 Z0.250 (offset current tool offset by 0.250, DRO now shows Z1.250)
M2 (end program)
```

- 詳細については、G90＆T＆M6 のセクションを参照してください。

次の場合はエラーになります。

- モーションは G43.1 と同じ行で指令されます

Note

G43.1 はツールテーブルに書き込みません。

4.3.29 G43.2 追加の工具長オフセットを適用する

```
G43.2 H-
```

- H-工具番号

G43.2 は、追加の同時工具オフセットを適用します。

G43.2 の例

```
G90 (set absolute mode)
T1 M6 (load tool 1)
G43 (or G43 H1 - replace all tool offsets with T1’s offset)
G43.2 H10 (also add in T10’s tool offset)
M2 (end program)
```

G43.2 をさらに呼び出すことで、任意の数のオフセットを合計できます。 どの数値がジオメトリオフセットで、どの数値が摩耗オフセットであるか、またはそれぞれを 1 つだけ持つ必要があるという組み込みの仮定はありません。

他の G43 コマンドと同様に、G43.2 はモーションを発生させません。 次に補正された軸が移動されたとき、その軸の端点が補正された位置になります。

次の場合はエラーになります。

- H が指定されていない、または
- 指定された工具番号が工具テーブルに存在しません

> **Note**
>
> G43.2 はツールテーブルに書き込みません。

4.3.30 G49 工具長補正のキャンセル

- G49-工具長補正をキャンセルします

すでに使用されているのと同じオフセットを使用してプログラムしても問題ありません。 現在使用されている工具長オフセットがない場合は、工具長オフセットを使用せずにプログラムすることもできます。

4.3.31 G52 ローカル座標系オフセット

```
G53 axes
```

G52 は、ワーク座標系内の一時的な「ローカル座標系オフセット」としてパートプログラムで使用されます。 G52 の詳細については、「ローカルおよびグローバルオフセット」セクションを参照してください。

4.3.32 G53 機械座標での移動

```
G53 axes
```

機械座標系で移動するには、直線移動と同じ行に G53 をプログラムします。 G53 はモーダルではないため、各行でプログラムする必要があります。 G0 または G1 が現在アクティブな場合は、同じ行にプログラムする必要はありません。

たとえば、G53 G0 X0 Y0 Z0 は、現在選択されている座標系に有効なオフセットがある場合でも、軸をホームポジションに移動します。

G53 の例

```
G53 G0 X0 Y0 Z0 (rapid linear move to the machine origin)
G53 X2 (rapid linear move to absolute coordinate X2)
```

- 詳細については、G0 セクションを参照してください。

次の場合はエラーになります。

- G53 は、G0 または G1 がアクティブでない状態で使用されます。
- または、カッター補正がオンのときに G53 が使用されます。

4.3.33 G54-G59.3 座標系の選択

- G54-座標系 1 を選択
- G55-座標系 2 を選択
- G56-座標系 3 を選択
- G57-座標系を選択 4
- G58-座標系を選択 5
- G59-座標系を選択 6
- G59.1-座標系の選択 7
- G59.2-座標系の選択 8
- G59.3-座標系の選択 9

座標系は、軸の値と Z 軸を中心とした XY 回転角を次の表に示すパラメータに保存します。

表 5.9：座標系パラメーター

	CS	X	Y	Z	A	B	C	U	V	W	R
G54	1	5221	5222	5223	5224	5225	5226	5227	5228	5229	5230
G55	2	5241	5242	5243	5244	5245	5246	5247	5248	5249	5250
G56	3	5261	5262	5263	5264	5265	5266	5267	5268	5269	5270
G57	4										

G58	5										
G59	6										
G59.1	7										
G59.2	8										
G59.3	9										

次の場合はエラーになります。

- 座標系の選択は、カッター補正がオンのときに使用されます。

座標系の概要については、座標系のセクションを参照してください。

4.3.34 G61 正確なパスモード

- G61-正確なパスモード、プログラムどおりの動き。 プログラムされたすべてのポイントに到達するために、必要に応じて移動が遅くなるか停止します。 2 つの連続した動きが正確に同一直線上にある場合、動きは停止しません。

4.3.35 G61.1 完全停止モード

- G61.1-正確な停止モード。プログラムされた各セグメントの終わりで移動が停止します。

4.3.36 G64 パスブレンディング

```
G64 <P- <Q->>
```

- P-モーションブレンディング許容値
- Q-ナイーブカムトレランス
- G64-可能な限り最高の速度。
- G64 P- <Q->公差のあるブレンド。
- G64-P なしは、プログラムされたポイントからどれだけ離れていても、可能な限り最高の速度を維持することを意味します。
- G64 P- Q--は、システムを微調整して、速度と精度の間で最適な妥協点を見つける方法です。 P-許容値は、実際のパスがプログラムされたエンドポイントから P-以内にあることを意味します。パスを維持するために必要な場合は、速度が低下します。さらに、G64 P- Q-をアクティブにすると、ナイーブカム検出器がオンになります。同じ送り速度で同一線上から離れる Q-未満の一連の線形 XYZ 送り移動がある場合、それらは単一の線形移動に折りたたまれます。 G2 / G3 では、直線からの円弧の最大偏差が G64 P-公差よりも小さい場合、G17（XY）平面内を移動します。円弧は、2 本の線に分割されます（円弧の始点から中点まで、および中点から終わり）。これらのラインは、ラインのナイーブカムアルゴリズムの対象になります。したがって、ラインアーク、アークアーク、およびアークラインの場合、およびラインラインは、ナイーブカム検出器の恩恵を受けます。これにより、パスが単純化され、輪郭のパフォーマンスが向上します。すでにアクティブ

になっているモードでプログラムしても問題ありません。これらのモードの詳細については、軌道制御のセクションも参照してください。 Q が指定されていない場合、Q は以前と同じ動作をし、P-の値を使用します。

G64P-ライン例

```
G64 P0.015 (set path following to be within 0.015 of the actual path)
```

各 G コードファイルのプリアンブルにパス制御仕様を含めることをお勧めします。

4.3.37 切りくずを伴う G73 穴あけサイクル

```
G73 X- Y- Z- R- Q- <L->
```

- R-Z 軸に沿って位置を後退させます。
- Q-Z 軸に沿ったデルタ増分。
- L-繰り返す

73 サイクルは、切りくずを破壊する穴あけまたはフライス盤です。 このサイクルは、Z 軸に沿ったデルタ増分を表す Q 番号を取ります。

1. 予備的な動き。
   - 現在の Z 位置が R 位置より下にある場合、Z 軸は R 位置に急速に移動します。
   - XY 座標に移動します
2. 現在の送り速度でのみ Z 軸をデルタだけ下に移動するか、Z 位置のどちらか浅い方に移動します。
3. 少し急上昇します。
4. 手順 2 で Z 位置に到達するまで、手順 2 と 3 を繰り返します。
5. Z 軸は R 位置に急速に移動します。

次の場合はエラーになります。

- Q 数が負またはゼロです。
- R 番号が指定されていません

4.3.38 G74 左側タッピングサイクル、ドウェル

```
G74 (X- Y- Z-) or (U- V- W-) R- L- P- $-
```

G74 サイクルは、フローティングチャックでタッピングし、穴の底に留まるように設計されています。

1. 予備動作と中間動作のセクションで説明されているように、予備動作。
2. フィードと速度のオーバーライドを無効にします。
3. 現在の送り速度で Z 軸を Z 位置に移動します。

4. 選択したスピンドルを停止します（$パラメーターで選択）
5. スピンドルの回転を時計回りに開始します。
6. P 秒間滞留します。
7. Z 軸を現在の送り速度で移動して、Z をクリアします
8. フィードと速度のオーバーライドを元に戻すと、以前の状態になります

ドウェルの長さは、G74 ブロックの P ワードで指定されます。 ねじ山のピッチは F を S で割ったものです。例では、S100F125 は 1 回転あたり 1.25MM のピッチを与えます。

4.3.39 G76 スレッディングサイクル

G76 P- Z- I- J- R- K- Q- H- E- L- $-

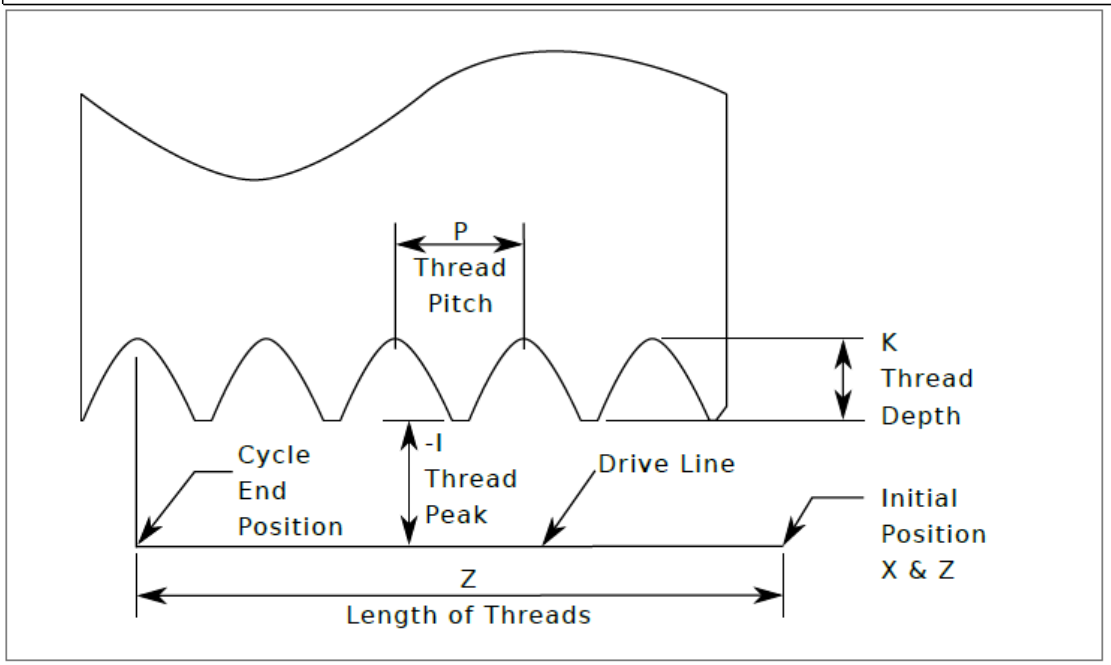


図 5-54

- ドライブライン-Z に平行な最初の X 位置を通る線。
- P --- 1 回転あたりの距離でのねじピッチ。
- Z--スレッドの最終位置。 サイクルの終わりに、ツールはこの Z 位置になります。

> Note
>
> G7 旋盤直径モードが有効な場合、I、J、および K の値は直径の測定値です。 G8 旋盤半径モードが有効な場合、I、J、および K の値は半径の測定値です。

- I--ドライブラインからのスレッドピークオフセット。 負の I 値は外部スレッドであり、正の I 値は内部スレッドです。 通常、材料は G76 サイクルの前にこのサイズになっています。

- J--初期カット深さを指定する正の値。 最初のスレッディングカットは、スレッドのピーク位置を超えた J になります。
- K--ねじ山の深さ全体を指定する正の値。 最終的なスレッディングカットは、スレッドのピーク位置を超えて K になります。

オプション設定

- $-モーションが同期されるスピンドル番号（デフォルトは 0）。 たとえば、$ 1 がプログラムされている場合、モーションはスピンドル 1.インデックスのリセットで開始され、スピンドル 1.revs の値と同期して進行します。
- R--深さの低下。 R1.0 は、連続するスレッドパスで一定の深さを選択します。 R2.0 は一定の領域を選択します。 1.0 から 2.0 の間の値は、深さを減らし、面積を増やすことを選択します。 2.0 を超える値は、減少する領域を選択します。 不必要に高いデグレッション値を使用すると、多数のパスが使用されることに注意してください。 （退行=段階またはステップによる降下。）
- Q--複合スライド角度は、連続するパスをドライブラインに沿ってどの程度オフセットする必要があるかを表す角度（度単位）です。 これは、ツールの一方の側がもう一方の側よりも多くの材料を除去するために使用されます。 正の Q 値を指定すると、工具の前縁がより大きく切断されます。 一般的な値は 29、29.5、または 30 です。
- H--スプリングパスの数。 スプリングパスは、ねじ山の深さ全体での追加パスです。 追加のパスが必要ない場合は、H0 をプログラムします。
- E--テーパーに使用されるドライブラインに沿った距離を指定します。 テーパーの角度は、E で指定された距離にわたって最後のパスがねじ山の頂上に向かってテーパーするようになります。「E0.2」は、ねじに沿った最初/最後の 0.2 長さの単位のテーパーを与えます。 45 度テーパープログラムの場合、E は K と同じです。
- L--スレッドのどちらの端がテーパーを取得するかを指定します。 テーパーなし（デフォルト）の場合は L0、入口テーパーの場合は L1、出口テーパーの場合は L2、入口テーパーと出口テーパーの両方の場合は L3 をプログラムします。 入口テーパーはドライブラインで一時停止してインデックスパルスと同期し、送り速度でテーパーの先頭に移動します。 入口テーパーはなく、工具は切削深さまで急速に進み、同期して切削を開始します。

G76 を発行する前に、ツールは最初の X および Z 位置に移動されます。 X 位置はドライブラインであり、Z 位置はねじ山の始点です。

ツールは、各ねじ切りパスの前に同期のために一時停止するため、ねじ山の始点が材料の終点を超えているか、入口テーパーが使用されていない限り、入口にレリーフ溝が必要になります。

出口テーパーを使用しない限り、出口の動きはスピンドル速度に同期せず、急速な動きになります。 スピンドルが遅い場合、出口の動きはほんのわずかな回転しかかからないかもしれません。 数回のパスが完了した後にスピンドル速度を上げると、その後の出口移動には 1 回転の大部分が必要になり、出口移動中に非常に重いカット

が発生します。 これは、出口に逃げ溝を設けるか、ねじ切り中にスピンドル速度を変更しないことで回避できます。

ツールの最終的な位置は、ドライブラインの端になります。 穴から工具を取り外すには、めねじを使用して安全な Z 移動が必要になります。

次の場合はエラーになります。

- アクティブな平面は ZX 平面ではありません
- X-や Y-などの他の軸ワードが指定されている
- R-degression 値は 1.0 未満です。
- 必要な単語がすべて指定されているわけではありません
- P-、J-、K-または H-は負です
- E-がドライブラインの長さの半分より大きい

HAL 接続 G76 が機能する前に、スピンドルのピンスピンドル.N.at-speed とエンコーダー.n.phase-Z を HAL ファイルに接続する必要があります。 詳細については、モーションセクションのスピンドルピンを参照してください。

技術情報 G76 の固定サイクルは、G33 スピンドル同期モーションに基づいています。 詳細については、G33 技術情報を参照してください。

サンプルプログラム g76.ngc は、G76 固定サイクルの使用法を示しており、sim /lathe.ini 構成を使用して任意のマシンでプレビューおよび実行できます。

G76 の例

```
G0 Z-0.5 X0.2
G76 P0.05 Z-1 I-.075 J0.008 K0.045 Q29.5 L2 E0.045
```

この図では、G76 サイクルが完了した後、ツールは最終位置にあります。 Q29.5 から右側に入口パス、L2E0.045 から左側に出口パスが表示されます。 白い線はカッティングムーブです。

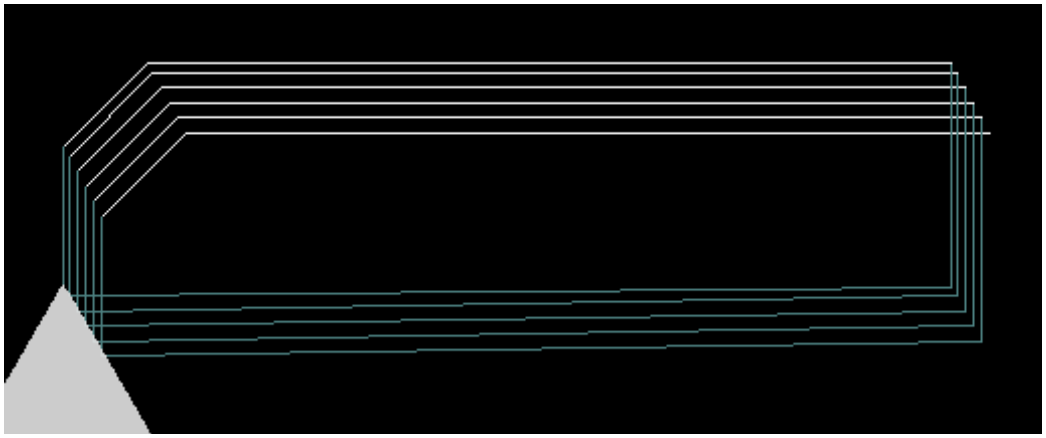


図 5-55

4.3.40 固定サイクル

このセクションでは、定型サイクル G81 から G89 および定型サイクルストップ G80 について説明します。

すべての固定サイクルは、現在選択されている平面に対して実行されます。 9 つの平面のいずれかを選択できます。 このセクション全体を通して、ほとんどの説明は XY 平面が選択されていることを前提としています。 別の平面が選択された場合の動作は類似しており、正しい単語を使用する必要があります。 たとえば、G17.1 平面では、固定サイクルのアクションは W に沿っており、位置または増分は U と V で指定されます。この場合、命令の X、Y、Z を U、V、W に置き換えます。 未満。

回転軸ワードは、定型サイクルでは使用できません。 アクティブプレーンが XYZ ファミリのいずれかである場合、UVW 軸ワードは許可されません。 同様に、アクティブな平面が UVW ファミリのいずれかである場合、XYZ 軸ワードは許可されません。

1.1.1.1 普通の言葉

すべての固定サイクルは、選択された平面と R ワードに応じて、X、Y、Z、または U、V、W グループを使用します。 R（通常はリトラクトを意味します）位置は、現在選択されている平面に垂直な軸（XY 平面の場合は Z 軸など）に沿っています。一部の固定サイクルでは、追加の引数が使用されます。

4.3.40.1 スティッキーワード

固定サイクルの場合、同じサイクルが連続する複数行のコードで使用されるときに、その番号を最初に使用する必要があるが、残りの行ではオプションである場合、番号をスティッキーと呼びます。 スティッキー番号は、異なるように明示的にプログラムされていない場合、残りの行でその値を保持します。 R 番号は常にスティッキーです。

増分距離モードでは、X、Y、および R の数値は現在の位置からの増分として扱われ、Z は Z を含む移動が行われる前の Z 軸の位置からの増分として扱われます。 絶対距離モードでは、X、Y、R、および Z の数値は現在の座標系での絶対位置です。

4.3.40.2 サイクルを繰り返す

L 番号はオプションであり、繰り返しの数を表します。 L = 0 は許可されていません。 リピート機能を使用する場合、通常はインクリメンタル距離モードで使用されるため、同じ一連のモーションが直線に沿って等間隔に配置された複数の場所で繰り返されます。 XY 平面が選択されたインクリメンタルモードで L-が 1 より大きい場合、X と Y の位置は、指定された X と Y の数値を現在の X と Y の位置（最初のゴーアラウンド）またはに加算することによって決定されます。 前のゴーアラウンドの終了時の X 位置と Y 位置（繰り返し）。 したがって、L10 をプログラムすると、10 サイクルが得られます。 最初のサイクルは、元の場所からの距離 X、Y になります。 R と Z の位置は繰り返し中に変化しません。 L 番号は粘着性がありません。 絶対距離モードでは、L> 1 は、同じ場所で同じサイクルを数回実行することを意味します。L ワードを省略することは、L = 1 を指定することと同じです。

4.3.40.3 リトラクトモード

各リピートの終了時のリトラクト移動の高さ（以下の説明ではクリアZと呼ばれます）は、元のZ位置（R位置より上でリトラクトモードがG98、OLD_Z）、またはそれ以外の場合はR位置に。 G98G99セクションを参照してください。

4.3.40.4 固定サイクルエラー

次の場合はエラーになります。

- 固定サイクル中に軸ワードがすべて欠落している、
- 異なるグループ（XYZ）（UVW）の軸ワードが一緒に使用され、
- P番号が必要であり、負のP番号が使用されます。
- 正の整数に評価されないL数が使用されます。
- 回転軸の動きは、固定のサイクル中に使用されます。
- 逆時間送り速度は、固定サイクル中にアクティブになります。
- または、カッター補正は、固定サイクル中にアクティブになります。

XY平面がアクティブな場合、Z番号はスティッキーであり、次の場合はエラーになります。

- Z番号が欠落しており、同じ固定サイクルがまだアクティブではありませんでした。
- または、R番号がZ番号よりも小さい。

他の平面がアクティブな場合、エラー条件は上記のXY条件に類似しています。

4.3.40.5 予備的および中間の動き

予備動作は、すべてのフライス盤の缶詰サイクルに共通する一連の動作です。 現在のZ位置がR位置より下にある場合、Z軸はR位置に急速に移動します。 これは、Lの値に関係なく、1回だけ発生します。

さらに、最初のサイクルの開始時と各繰り返しで、次の1つまたは2つの動きが行われます。

1. XY平面に平行な、指定されたXY位置への急速な移動。
2. Z軸は、まだR位置にない場合は、R位置にすばやく移動します。

別の平面がアクティブな場合、予備動作と中間動作は類似しています。

4.3.40.6 なぜ固定サイクルを使用するのですか？

固定サイクルを使用する理由は少なくとも2つあります。 1つ目は、コードの経済性です。 1つのボアを実行するには、数行のコードが必要です。

G81の例1は、固定サイクルを使用して、固定サイクルモード内で10行のGコードを含む8つの穴を生成する方法を示しています。 以下のプログラムは、固定サイクルに5本の線を使用して同じ8つの穴のセットを生成します。 前の例とまったく同じパスをたどったり、同じ順序でドリルしたりすることはありません。 しかし、良い缶詰サイクルのプログラム作成経済は明白であるはずです。

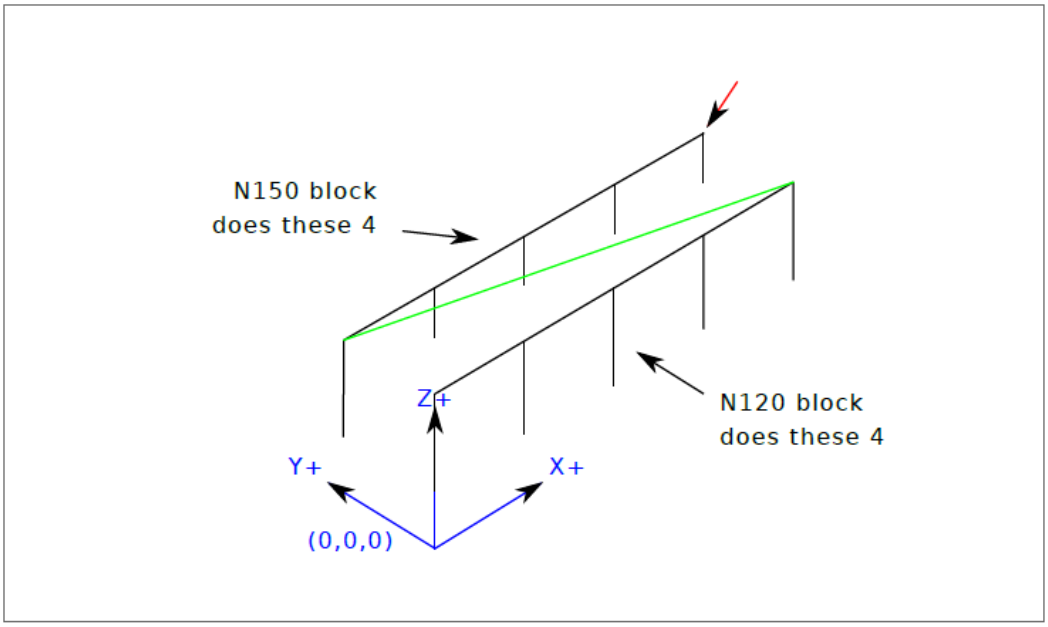
Note

行番号は必要ありませんが、これらの例を明確にするのに役立ちます

8 つの穴

```
N100 G90 G0 X0 Y0 Z0 (move coordinate home)
N110 G1 F10 X0 G4 P0.1
N120 G91 G81 X1 Y0 Z-1 R1 L4(canned drill cycle)
N130 G90 G0 X0 Y1
N140 Z0
N150 G91 G81 X1 Y0 Z-0.5 R1 L4(canned drill cycle)
N160 G80 (turn off canned cycle)
N170 M2 (program end)
```

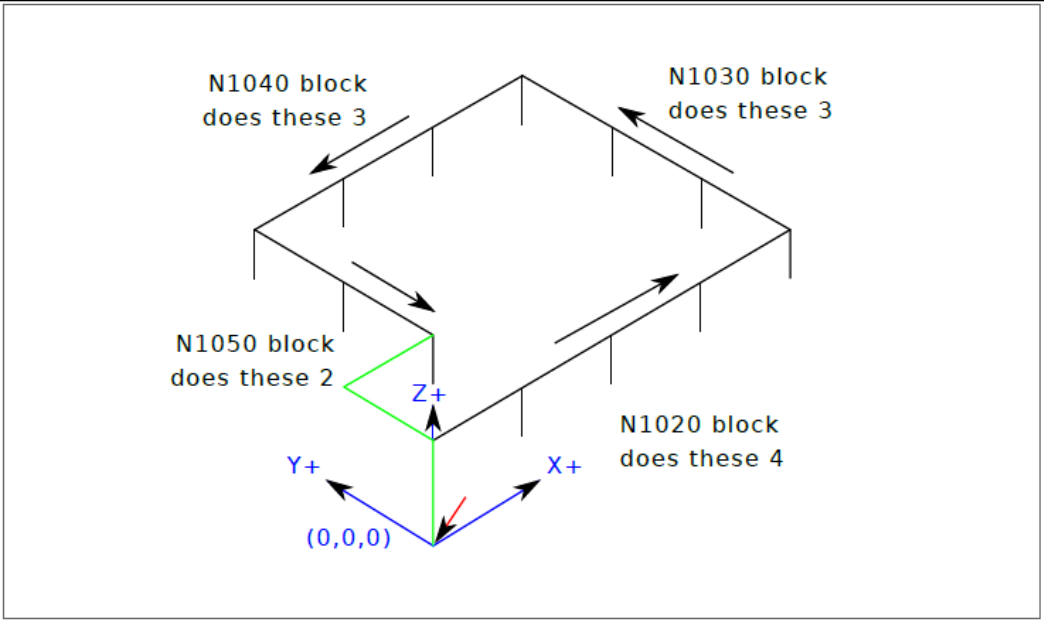
上記の 2 行目の G98 は、指定された R 値よりも高いため、最初の行の Z の値に戻ることを意味します。



正方形の 12 個の穴この例は、L ワードを使用して、同じ G81 モーションモード内の連続するコードブロックに対して一連の増分ドリルサイクルを繰り返す方法を示しています。 ここでは、キャンドモーションモードで 5 行のコードを使用して 12 個の穴を作成します。

```
N1000 G90 G0 X0 Y0 Z0 (move coordinate home)
N1010 G1 F50 X0 G4 P0.1
N1020 G91 G81 X1 Y0 Z-0.5 R1 L4 (canned drill cycle)
N1030 X0 Y1 R0 L3 (repeat)
N1040 X-1 Y0 L3 (repeat)
N1050 X0 Y-1 L2 (repeat)
N1060 G80 (turn off canned cycle)
N1070 G90 G0 X0 (rapid move home)
N1080 Y0
```

```
N1090 Z0
N1100 M2 (program end)
```



固定サイクルを使用する 2 つ目の理由は、固定サイクルの開始点に関係なく、それらすべてが事前の移動とリターンを生成し、予測および制御できることです。

4.3.41 G80 固定サイクルのキャンセル

G80-固定サイクルモーダルモーションをキャンセルします。 G80 はモーダルグループ 1 の一部であるため、モーダルグループ 1 から他の G コードをプログラミングすると、固定サイクルもキャンセルされます。

次の場合はエラーになります。

- 軸ワードは、G80 がアクティブなときにプログラムされます。

G80 の例

```
G90 G81 X1 Y1 Z1.5 R2.8 (absolute distance canned cycle)
G80 (turn off canned cycle motion)
G0 X0 Y0 Z0 (rapid move to coordinate home)
```

次のコードは、前のコードと同じ最終位置とマシン状態を生成します。

G0 の例

```
G90 G81 X1 Y1 Z1.5 R2.8 (absolute distance canned cycle)
G0 X0 Y0 Z0 (rapid move to coordinate home)
```

最初のセットの利点は、G80 ラインが G81 缶詰サイクルを明らかにオフにすることです。 ブロックの最初のセットで、プログラマーは次の行で行われるように、G0 または他のモーションモード G ワードでモーションをオンに戻す必要があります。

固定サイクルが G80 または別のモーションワードでオフにされていない場合、固定サイクルは、X、Y、または Z ワードを含むコードの次のブロックを使用して繰り返されます。 次のファイルは、次のキャプションに示すように、8 つの穴のセットをドリル（G81）します。

G80 の例 1

```
N100 G90 G0 X0 Y0 Z0 (coordinate home)
N110 G1 X0 G4 P0.1
N120 G81 X1 Y0 Z0 R1 (canned drill cycle)
N130 X2
N140 X3
N150 X4
N160 Y1 Z0.5
N170 X3
N180 X2
N190 X1
N200 G80 (turn off canned cycle)
N210 G0 X0 (rapid move home)
N220 Y0
N230 Z0
N240 M2 (program end)
```

Note

最初の 4 つの穴の後で z 位置が変化することに注意してください。 また、これは行番号に何らかの価値がある数少ない場所の 1 つであり、読者に特定のコード行を示すことができます。

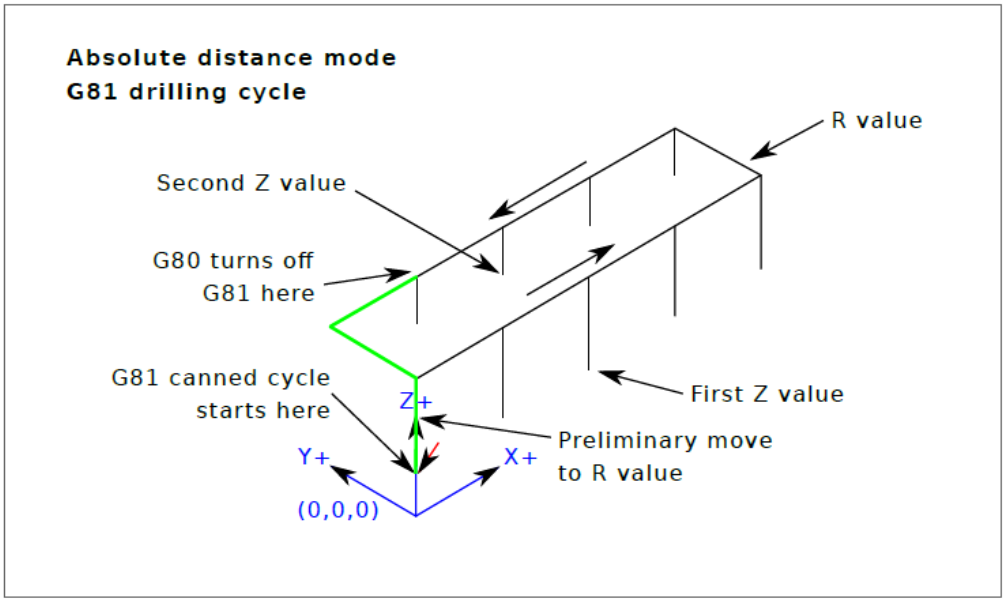


図 5-56

次の行の G0 は G81 サイクルをオフにするため、行 N200 での G80 の使用はオプションです。 ただし、例 1 に示すように G80 を使用すると、缶詰のサイクルが読みやすくなります。 それがなければ、N120 と N200 の間のすべてのブロックが固定サイクルに属していることはそれほど明白ではありません。

4.3.42 G81 ドリルサイクル

```
G81 (X- Y- Z-) or (U- V- W-) R- L-
```

G81 サイクルは、穴あけを目的としています。

サイクルは次のように機能します。

1. 予備動作と中間動作のセクションで説明されているように、予備動作。
2. 現在の送り速度で Z 軸を Z 位置に移動します。
3. Z 軸は、Z をクリアするために急速に移動します。

例 1-絶対位置 G81 現在の位置が（X1、Y2、Z3）であり、NC コードの次の行が解釈されるとします。

```
G90 G98 G81 X4 Y5 Z1.5 R2.8
```

これには、絶対距離モード（G90）と OLD_Z リトラクトモード（G98）が必要であり、G81 ドリルサイクルを 1 回実行する必要があります。

X 値と X 位置は 4 です。

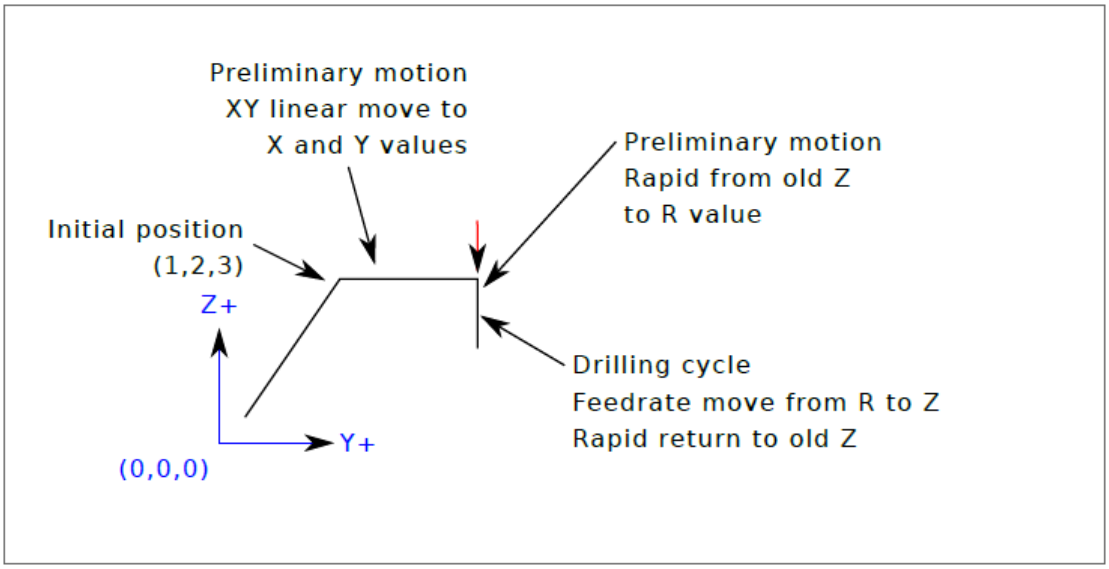
Y 値と Y 位置は 5 です。

Z 値と Z 位置は 1.5 です。

R 値とクリア Z は 2.8 です。 OLD_Z は 3 です。

次の動きが起こります：

1. XY 平面に平行な（X4、Y5）への急速な移動
2. 早送りは Z 軸に平行に（Z2.8）に移動します。
3. （Z1.5）への送り速度で Z 軸に平行に移動します
4. Z 軸に平行な（Z3）への急速な移動



例 2-相対位置 G81 現在の位置が（X1、Y2、Z3）であり、NC コードの次の行が解釈されるとします。

```
G91 G98 G81 X4 Y5 Z-0.6 R1.8 L3
```

これには、増分距離モード（G91）と OLD_Z リトラクトモード（G98）が必要です。 また、G81 ドリルサイクルを 3 回繰り返す必要があります。 X 値は 4、Y 値は 5、Z 値は-0.6、R 値は 1.8 です。 初期の X 位置は 5（= 1 + 4）、初期の Y 位置は 7（= 2 + 5）、クリアな Z 位置は 4.8（= 1.8 + 3）、Z 位置は 4.2（= 4.8-0.6）です。 ）。 OLD_Z は 3 です。

OLD_Z <クリア Z であるため、最初の予備移動は、Z 軸に沿った（X1、Y2、Z4.8）への最大急速移動です。

最初の繰り返しは 3 回の動きで構成されます。

1. XY 平面に平行な（X5、Y7）への急速な移動
2. （Z4.2）への送り速度で Z 軸に平行に移動します
3. Z 軸に平行な（X5、Y7、Z4.8）への急速な移動

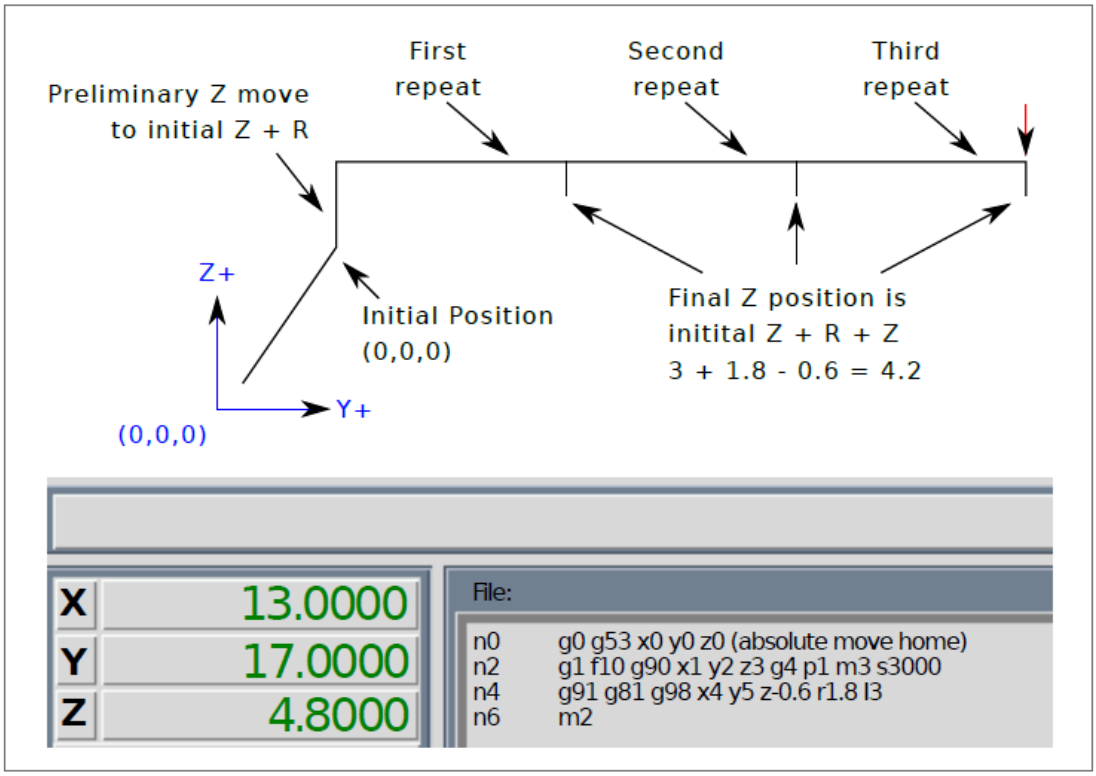
2 回目の繰り返しは 3 回の動きで構成されます。 X 位置は 9（= 5 + 4）にリセットされ、Y 位置は 12（= 7 + 5）にリセットされます。

1. XY 平面に平行な（X9、Y12、Z4.8）への急速な移動
2. （X9、Y12、Z4.2）への送り速度で Z 軸に平行に移動します
3. Z 軸に平行な（X9、Y12、Z4.8）への急速な移動

3 回目の繰り返しは 3 回の動きで構成されます。 X 位置は 13（= 9 + 4）にリセットされ、Y 位置は 17（= 12 + 5）にリセットされます。

1. XY 平面に平行な（X13、Y17、Z4.8）への急速な移動
2. （X13、Y17、Z4.2）への送り速度で Z 軸に平行に移動します

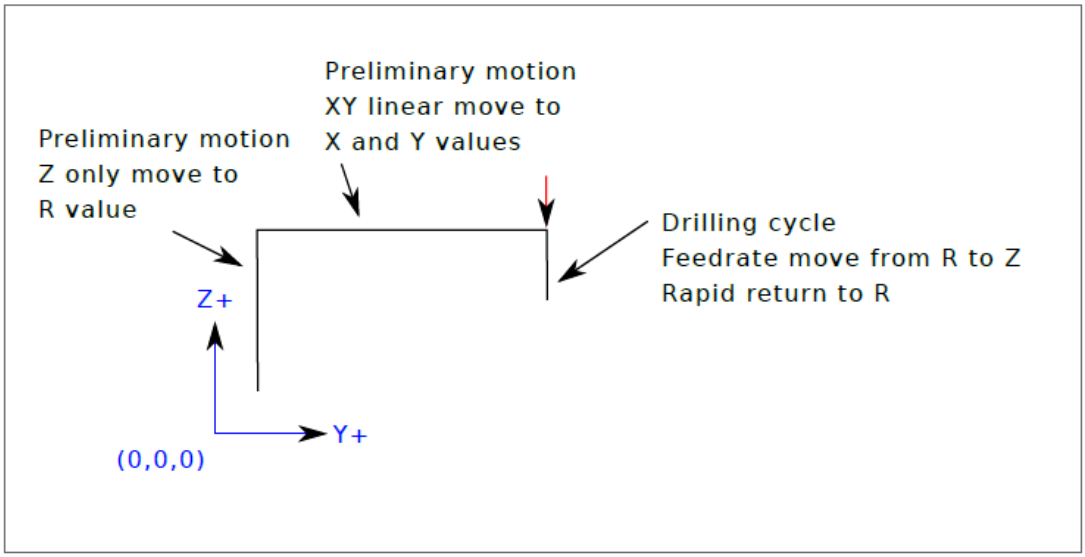
3. Z 軸に平行な（X13、Y17、Z4.8）への急速な移動



例 3-相対位置 G81 ここで、コードの最初の G81 ブロックを実行するとしますが、（X1、Y2、Z3）からではなく（X0、Y0、Z0）から実行します。

```
G90 G98 G81 X4 Y5 Z1.5 R2.8
```

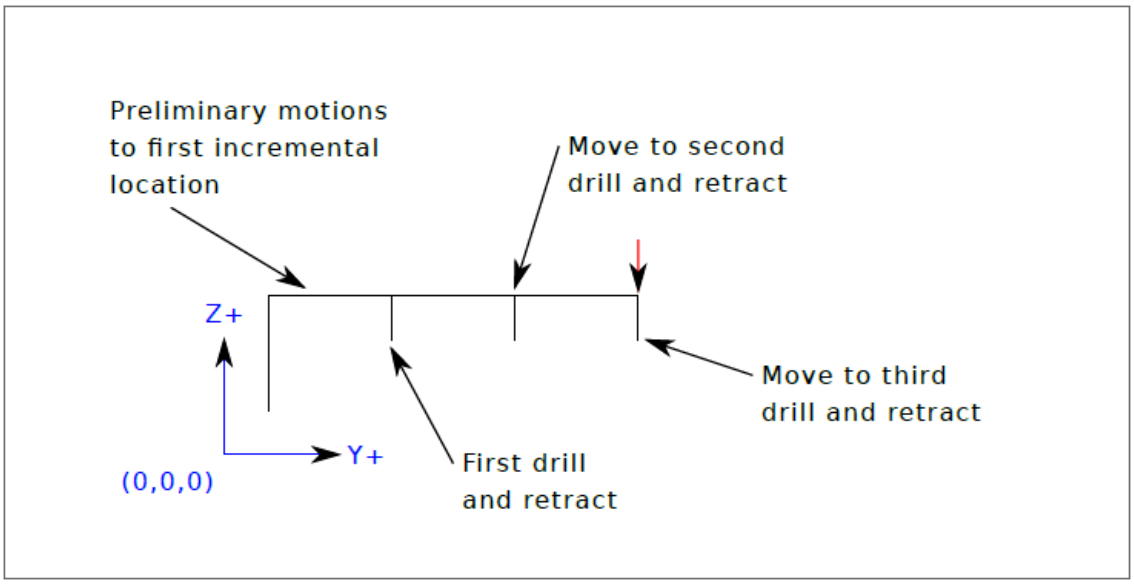
OLD_Z は R 値を下回っているため、モーションには何も追加されませんが、Z の初期値が R で指定された値よりも小さいため、予備移動中に最初の Z 移動が発生します。



例 4-絶対 G81R> Z これは、コードの 2 番目の g81 ブロックのモーションパスのプロットです。

```
G91 G98 G81 X4 Y5 Z-0.6 R1.8 L3
```

このプロットは（X0、Y0、Z0）で始まるため、インタープリターは最初の Z0 と R1.8 を追加し、その場所にすばやく移動します。 最初の Z 移動後、繰り返し機能は例 3 と同じように機能し、最終的な Z 深度は R 値より 0.6 低くなります。



例 5-相対位置 R> Z

```
G90 G98 G81 X4 Y5 Z-0.6 R1.8
```

このプロットは（X0、Y0、Z0）で始まるため、インタープリターは最初の Z0 と R1.8 を追加し、例 4 のようにその場所に高速移動します。最初の Z 移動の後、X4Y5 への高速移動が行われます。 次に、最終的な Z 深度は R 値より 0.6 低くなります。 繰り返し関数は、Z を同じ場所に再び移動させます。

4.3.43 G82 ドリルサイクル、ドゥエル付き

G82 (X- Y- Z-) or (U- V- W-) R- L- P-

G82 サイクルは、穴の底にドウェルを備えた穴あけを目的としています。

1. 予備動作と中間動作のセクションで説明されているように、予備動作。
2. 現在の送り速度で Z 軸を Z 位置に移動します。
3. P 秒間滞留します。
4. Z 軸は、Z をクリアするために急速に移動します。

G82 固定サイクルの動きは G81 と同じように見えますが、Z ムーブの下部にドウェルが追加されています。 ドウェルの長さは、G82 ブロックの P ワードで指定されます。

4.3.44 G83 ペックドリルサイクル

G83 (X- Y- Z-) or (U- V- W-) R- L- Q-

G83 サイクル（多くの場合、ペックドリルと呼ばれます）は、切りくず破壊を伴う深穴加工またはフライス盤加工を目的としています。 このサイクルでのリトラクトは、切りくずの穴を取り除き、長いストリンガー（アルミニウムを穴あけするときに一般的です）を切断します。 このサイクルは、Z 軸に沿ったデルタ増分を表す Q 番号を取ります。 G98 が有効な場合でも、最終的な深さの前のリトラクトは常にリトラクト平面になります。 最後の撤回は、有効な G98 / 99 を尊重します。 G83 は G81 と同じように機能しますが、穴あけ作業中にリトラクトが追加されます。

1. 予備動作と中間動作のセクションで説明されているように、予備動作。
2. 現在の送り速度で Z 軸をデルタだけ下に移動するか、Z 位置のどちらか浅い方に移動します。
3. R ワードで指定されたリトラクト平面にすばやく戻ります。
4. 少し後退して、現在の穴の底にすばやく戻ります。
5. 手順 2 で Z 位置に到達するまで、手順 2、3、および 4 を繰り返します。
6. と 2 で Z 座に集するます、他 2、3、お 4 を到ます。

次の場合はエラーになります。

- Q 数が負またはゼロです。

4.3.45 G84 右側タッピングサイクル、ドウェル

```
G84 (X- Y- Z-) or (U- V- W-) R- L- P- $-
```

G84 サイクルは、フローティングチャックでタッピングし、穴の底に留まるように設計されています。

1. 予備動作と中間動作のセクションで説明されているように、予備動作。
2. フィードと速度のオーバーライドを無効にします。
3. 現在の送り速度で Z 軸を Z 位置に移動します。
4. 選択したスピンドルを停止します（$パラメーターで選択）
5. スピンドルの回転を反時計回りに開始します。
6. P 秒間滞留します。
7. Z 軸を現在の送り速度で移動して、Z をクリアします
8. フィードと速度のオーバーライドを元に戻すと、以前の状態になります

ドウェルの長さは、G84 ブロックの P ワードで指定されます。 ねじ山のピッチは F を S で割ったものです。例では、S100F125 は 1 回転あたり 1.25MM のピッチを与えます。

4.3.46 G85 ボーリングサイクル、フィードアウト

```
G85 (X- Y- Z-) or (U- V- W-) R- L-
```

G85 サイクルは、ボーリングまたはリーマ加工を目的としていますが、穴あけまたはフライス盤に使用することもできます。

1. 予備動作と中間動作のセクションで説明されているように、予備動作。
2. 現在の送り速度でのみ Z 軸を Z 位置に移動します。
3. 初期 Z よりも低い場合は、現在の送り速度で R 平面への Z 軸を後退させます。
4. トラバース速度で後退して Z をクリアします。

4.3.47 G86 ボーリングサイクル、スピンドルストップ、ラピッドムーブアウト

```
G86 (X- Y- Z-) or (U- V- W-) R- L- P- $-
```

G86 サイクルは退屈を目的としています。 このサイクルでは、滞留する秒数に P 番号を使用します。

1. 予備動作と中間動作のセクションで説明されているように、予備動作。
2. 現在の送り速度でのみ Z 軸を Z 位置に移動します。
3. 速度のパッキング速度でざ軸を Z 位置に移動します。
4. 選択したスピンドルの回転を停止します。 （$パラメーターで選択）
5. Z 軸は、Z をクリアするために急速に移動します。

6. スピンドルを進行方向に再起動します。

次の場合はエラーになります。

- このサイクルが実行される前に、スピンドルは回転していません。

4.3.48 G87 バックボーリングサイクル

このコードは現在、LinuxCNC では実装されていません。 受け入れられますが、動作は定義されていません。

4.3.49 G88 ボーリングサイクル、スピンドルストップ、マニュアルアウト

このコードは現在、LinuxCNC では実装されていません。 受け入れられますが、動作は定義されていません。

4.3.50 G89 ボーリングサイクル、ドウェル、フィードアウト

```
G89 (X- Y- Z-) or (U- V- W-) R- L- P-
```

G89 サイクルは退屈を目的としています。 このサイクルでは、P 番号を使用します。ここで、P は滞留する秒数を指定します。

1. 予備動作と中間動作のセクションで説明されているように、予備動作。
2. 現在の送り速度でのみ Z 軸を Z 位置に移動します。
3. P 秒間滞留します。
4. Z をクリアするには、現在の送り速度で Z 軸を後退させます。

4.3.51 G90、G91 距離モード

- G90-絶対距離モード絶対距離モードでは、軸番号（X、Y、Z、A、B、C、U、V、W）は通常、現在アクティブな座標系での位置を表します。 その規則の例外は、G80G89 セクションで明示的に説明されています。
- G91-増分距離モード増分距離モードでは、軸番号は通常、現在の座標からの増分を表します。

G90 の例

```
G90 (set absolute distance mode)
G0 X2.5 (rapid move to coordinate X2.5 including any offsets in effect)
```

G91 の例

```
G91 (set incremental distance mode)
G0 X2.5 (rapid move 2.5 from current position along the X axis)
```

- 詳細については、G0 セクションを参照してください。

4.3.52 G90.1、G91.1 アーク距離モード

- G90.1-I、J、および K オフセットの絶対距離モード。 G90.1 が有効な場合、I と J の両方を XY 平面の場合は G2 / 3 で、XZ 平面の場合は J と K で指定する必要があります。そうしないとエラーになります。
- G91.1-I、J、および K オフセットの増分距離モード。 G91.1 I、J、および K をデフォルトの動作に戻します。

4.3.53 G92 座標系オフセット

G92 axes

警告

マシンが目的のポイントに配置された後でのみ、G92 を使用してください。

G92 は、現在のポイントに必要な座標（モーションなし）を持たせます。軸ワードには、必要な軸番号が含まれます。 少なくとも 1 つを使用する必要があることを除いて、すべての軸ワードはオプションです。 特定の軸に軸ワードが使用されていない場合、その軸のオフセットはゼロになります。

G92 を実行すると、すべての座標系の原点が移動します。 それらは、現在アクティブな座標系で現在制御されているポイントの値が指定された値になるように移動します。 座標系のすべての原点（G53-G59.3）は、この同じ距離だけオフセットされています。

G92 は、パラメータ 5211-5219 に格納されている値を、各軸の X Y Z A B C U VW オフセット値として使用します。 パラメータ値は、ini ファイルで指定されているネイティブマシンユニットの絶対マシン座標です。 G92 がアクティブな場合、ini ファイルで定義されているすべての軸がオフセットされます。 G92 に続いて軸が入力されなかった場合、その軸のオフセットはゼロになります。

たとえば、現在のポイントが X = 4 にあり、現在アクティブな G92 オフセットがないとします。 次に、G92X7 がプログラムされます。 これにより、すべての原点が X 内で-3 移動し、現在のポイントが X = 7 になります。 この-3 はパラメータ 5211 に保存されます。

インクリメンタルディスタンスモード（G90 ではなく G91）であっても、G92 のアクションには影響しません。

G92 が呼び出されたときに、G92 オフセットがすでに有効になっている可能性があります。 この場合、オフセットは、現在のポイントが指定された値になる新しいオフセットに置き換えられます。

次の場合はエラーになります。

- すべての軸ワードが省略されます。

LinuxCNC は、G92 オフセットを保存し、プログラムの次の実行でそれらを再利用します。 これを防ぐために、G92.1 をプログラムする（それらを消去する）か、G92.2 をプログラムする（それらを削除する-それらはまだ保存されている）ことができます。

Note

G52 コマンドを使用して、このオフセットを変更することもできます。 G92 と G52 の詳細、およびそれらがどのように相互作用するかについては、オフセットのセクションを参照してください。

座標系の概要については、座標系のセクションを参照してください。

詳細については、パラメータのセクションを参照してください。

4.3.54 G92.1、G92.2 リセット G92 オフセット

- G92.1-G92 オフセットをオフにし、パラメータ 5211-5219 をゼロにリセットします。
- G92.2-G92 オフセットをオフにしますが、パラメータ 5211-5219 を使用可能なままにします。

> Note
>
> G92.1 は G92 オフセットのみをクリアします。G コードで G53-G59.3 座標系オフセットを変更するには、G10L2 または G10L20 のいずれかを使用します。

4.3.55 G92.3G92 オフセットの復元

- G92.3-G92 オフセットをパラメータ 5211～5219 に保存されている値に設定します

あるプログラムで軸オフセットを設定し、別のプログラムで同じオフセットを使用することができます。 最初のプログラムのプログラム G92。 これにより、パラメータ 5211～5219 が設定されます。最初のプログラムの残りの部分では G92.1 を使用しないでください。 パラメータ値は、最初のプログラムが終了すると保存され、2 番目のプログラムが起動すると復元されます。 2 番目のプログラムの開始近くで G92.3 を使用します。 これにより、最初のプログラムで保存されたオフセットが復元されます。

4.3.56 G93、G94、G95 送り速度モード

- G93-は逆時間モードです。 逆時間送り速度モードでは、F ワードは、移動が[1 を F 値で割った値]分で完了する必要があることを意味します。 たとえば、F 値が 2.0 の場合、移動は 30 分で完了する必要があります。

  逆時間送り速度モードがアクティブな場合、G1、G2、または G3 モーションがあるすべてのラインに F ワードが表示される必要があり、G1、G2、または G3 がないラインの F ワードは無視されます。 逆時間送り速度モードであっても、G0（早送り）動作には影響しません。
- G94-は単位/分モードです。 単位/分送りモードでは、F ワードは、使用されている長さの単位と 1 つまたは複数の軸に応じて、制御点が特定のインチ/分、ミリメートル/分、または度/分で移動する必要があることを意味すると解釈されます。 動いています。
- G95-は 1 回転あたりの単位モードです 1 回転あたりの単位モードでは、F ワードは、使用されている単位の長さと 1 つまたは複数の軸に応じて、制御点がスピンドルの 1 回転あたり特定のインチ数移動する必要があることを意味すると解釈されます。 移動します。 G95 はスレッディングには適していません。スレッディングには G33 または G76 を使用してください。 G95 では、スピンドル.N。スピードインを接続する必要があります。 フィードが同期される実際のスピンドルは、$パラメータによって選択されます

次の場合はエラーになります。

- 逆タイムフィードモードがアクティブであり、G1、G2、またはG3（明示的または暗黙的）のあるラインにはFワードがありません。
- G94またはG95に切り替えた後、新しい送り速度が指定されない

4.3.57 G96、G97スピンドル制御モード

```
G96 <D-> S- <$-> (Constant Surface Speed Mode)
G97 S- <$-> (RPM Mode)
```

- D-最大スピンドルRPM
- S-表面速度
- '$ "-速度が変化するスピンドル。
- G96 D- S--は、毎分Sフィート（G20が有効な場合）または毎分メートル（G21が有効な場合）の一定の表面速度を選択します。 D-はオプションです。

G96を使用する場合は、現在の座標系（オフセットとツールの長さを含む）のX0が回転の中心であることを確認してください。そうでない場合、LinuxCNCは目的の表面速度を提供しません。 G96は、半径または直径モードの影響を受けません。

選択したスピンドルでCSSモードを実現するには、M3を発行する前に、各スピンドルに対して連続するG96コマンドをプログラムします。

- G97はRPMモードを選択します。

G96サンプルライン

```
G96 D2500 S250 (set CSS with a max rpm of 2500 and a surface speed of 250)
```

次の場合はエラーになります。

- SはG96で指定されていません
- 送り移動は、主軸が回転していないときにG96モードで指定されます

4.3.58 G98、G99固定サイクルリターンレベル

- G98-この一連の1つまたは複数の連続した固定サイクルが開始される直前に軸があった位置に後退します。
- G99-固定サイクルのRワードで指定された位置に後退します。

G98をプログラムすると、固定サイクルは、固定サイクルの前のZ位置が、サイクルで指定されたR値よりも高い場合、Z戻り位置として使用されます。 低い場合は、R値が使用されます。 Rワードは、絶対距離モードと増分距離モードで異なる意味を持ちます。

G98原点への撤回

```
G0 X1 Y2 Z3
G90 G98 G81 X4 Y5 Z-0.6 R1.8 F10
```

上記の 2 行目の G98 は、指定された R 値よりも高いため、最初の行の Z の値に戻ることを意味します。

初期（G98）平面は、明示的（G80）または暗黙的（サイクルではないモーションコード）に関係なく、サイクルモーションモードが破棄されるとリセットされます。 サイクルモード（G81 から G83 など）を切り替えても、初期プレーンはリセットされません。 一連のサイクル中に G98 と G99 を切り替えることができます。

4.4 M コード

4.4.1 M コードクイックリファレンステーブル

コード	説明
M0 M1	プログラムの一時停止
M2 M30	プログラム終了
M60	パレット交換の一時停止
M3 M4 M5	スピンドル制御
M6	工具交換
M7 M8 M9	クーラント制御
M19	スピンドルオリエンテーション
M48 M49	フィードとスピンドルのオーバーライドの有効化/無効化
M50	フィードオーバーライド制御
M51	スピンドルオーバーライド制御
M52	アダプティブフィードコントロール
M53	フィードストップコントロール
M61	現在の工具番号を設定する
M62-M65	出力制御
M66	入力制御
M67	アナログ出力制御
M68	アナログ出力制御
M70	モーダル状態を保存
M71	保存されたモーダル状態を無効にする
M72	モーダル状態を復元
M73	自動復元モーダル状態を保存
M98 M99	サブプログラムからの呼び出しと戻り
M100-M199	ユーザー定義の M コード

4.4.2 M0、M1 プログラムの一時停止

- M0-実行中のプログラムを一時的に一時停止します。 LinuxCNC は自動モードのままであるため、MDI およびその他の手動アクションは有効になりません。 再開ボタンを押すと、次の行でプログラムが再起動します。
- M1-オプションの停止スイッチがオンの場合、実行中のプログラムを一時的に一時停止します。 LinuxCNC は自動モードのままであるため、MDI およびその他の手動アクションは有効になりません。 再開ボタンを押すと、次の行でプログラムが再起動します。

Note

M0 と M1 を MDI モードでプログラムしても問題ありませんが、MDI モードの通常の動作は入力の各行の後で停止するため、効果はおそらく目立たないでしょう。

4.4.3 M2、M30 プログラム終了

- M2-プログラムを終了します。 Cycle Start（Axis GUI の「R」）を押すと、ファイルの先頭からプログラムが再起動します。
- M30-パレットシャトルを交換し、プログラムを終了します。 Cycle Start を押すと、ファイルの先頭からプログラムが開始されます。

これらのコマンドは両方とも次の効果があります。

1. 自動モードから MDI モードに変更します。
2. 原点オフセットはデフォルトに設定されています（G54 など）。
3. 選択した平面が XY 平面に設定されます（G17 など）。
4. 距離モードは絶対モード（G90 など）に設定されます。
5. 送り速度モードは、1 分あたりの単位に設定されます（G94 など）。
6. 送りと速度のオーバーライドはオンに設定されています（M48 など）。
7. カッター補正がオフになっています（G40 など）。
8. スピンドルが停止します（M5 のように）。
9. 現在のモーションモードはフィードに設定されています（G1 など）。
10. クーラントがオフになっています（M9 のように）。

Note

M2 / M30 以降のコード行は実行されません。 Cycle Start を押すと、ファイルの先頭からプログラムが開始されます。

警告

%を使用してGコードをラップしても、プログラム終了と同じことはできません。 %を使用しても実行されないことの詳細については、ファイル要件を参照してください。

4.4.4 M60 パレット変更の一時停止

- M60-パレットシャトルを交換してから、実行中のプログラムを一時的に一時停止します（オプションの停止スイッチの設定に関係なく）。 サイクルスタートボタンを押すと、次の行からプログラムが再起動します。

4.4.5 M3、M4、M5 スピンドル制御

- M3 [$ n]-選択したスピンドルをS速度で時計回りに始動します。
- M4 [$ n]-選択したスピンドルをS速度で反時計回りに始動します。
- M5 [$ n]-選択したスピンドルを停止します。

$を使用して、特定のスピンドルを操作します。 $を省略すると、コマンドはデフォルトでスピンドルで動作します。0$ -1 を使用して、すべてのアクティブなスピンドルで動作します。

この例では、スピンドル0、1、および2を異なる速度で同時に開始します。

```
S100 $0
S200 $1
S300 $2
M3 $-1
```

この例では、スピンドル1を逆にしますが、他のスピンドルは前方に回転させたままにします。

```
M4 $1
```

そして、これによりスピンドル2が停止し、他のスピンドルは回転したままになります。

```
M5 $2
```

$を省略すると、動作はシングルスピンドルマシンの通常とまったく同じになります。

Sスピンドル速度がゼロに設定されている場合は、M3またはM4を使用しても問題ありません。 これが行われた場合（または速度オーバーライドスイッチが有効でゼロに設定されている場合）、スピンドルは回転を開始しません。 後でスピンドル速度がゼロより上に設定された場合（またはオーバーライドスイッチが上にされた場合）、スピンドルは回転を開始します。 主軸がすでに回転している場合はM3またはM4を使用し、主軸がすでに停止している場合はM5を使用しても問題ありません。

4.4.6 M6 工具交換

1.1.1.1 手動工具交換

HAL コンポーネント hal_manualtoolchange がロードされると、M6 はスピンドルを停止し、プログラムされた最後の T 番号に基づいてツールを変更するようにユーザーに促します。 hal_manualtoolchange の詳細については、「手動ツール変更」セクションを参照してください。

4.4.6.1 ツールチェンジャー

主軸の工具を現在主軸にある工具から最後に選択した工具に変更するには（T ワードを使用-セクション選択工具を参照）、M6 をプログラムします。 工具交換が完了したら：

- スピンドルが停止します。
- （同じ行または前回の工具交換後の任意の行の T ワードによって）選択された工具が主軸になります。
- 工具交換前に選択した工具が主軸になかった場合、主軸にあった工具（ある場合）は工具交換マガジンに戻されます。
- .ini ファイルで構成されている場合、M6 が発行されたときに一部の軸位置が移動する可能性があります。ツール変更オプションの詳細については、EMCIO セクションを参照してください。
- その他の変更は行われません。 たとえば、クーラントは、M9 によってオフにされていない限り、工具交換中に流れ続けます。

警告

工具長オフセットは M6 によって変更されません。M6 の後に G43 を使用して、工具長オフセットを変更してください。

工具交換には、軸の動きが含まれる場合があります。 すでにスピンドルにあるツールへの変更をプログラムすることは OK です（しかし有用ではありません）。 選択したスロットにツールがなくても問題ありません。 その場合、工具交換後、スピンドルは空になります。 スロットゼロが最後に選択された場合、工具交換後、スピンドルに工具は確実にありません。 ツールチェンジャーは、半分、場合によってはクラシックラダーでツール交換を実行するように設定する必要があります。

4.4.7 M7、M8、M9 クーラントコントロール

- M7-ミストクーラントをオンにします。 M7 は iocontrol.0.coolant-mist ピンを制御します。
- M8-フラッドクーラントをオンにします。 M8 は iocontrol.0.coolant-flood ピンを制御します。
- M9-M7 と M8 の両方をオフにします。

M7 または M8 が出力を制御する前に、HAL のクーラント制御ピンの一方または両方を接続します。 M7 および M8 を使用して、G コードを介して任意の出力をオンにすることができます。

現在のクーラントの状態に関係なく、これらのコマンドのいずれかを使用しても問題ありません。

4.4.8 M19 スピンドルオリエンテーション

- M19 R- Q- [P-] [$-]
- R 0 から回転する位置。有効な範囲は 0〜360 度です。
- Q オリエントが完了するまで待機する秒数。 スピンドル.N.is 指向が Q タイムアウト内に真にならない場合、エラーが発生します。
- P 位置に回転する方向。
  - 0 回転して最小の角運動量（デフォルト）
  - 1 は常に時計回りに回転します（M3 方向と同じ）
  - 2 は常に反時計回りに回転します（M4 方向と同じ）
- $方向付けするスピンドル（実際には、どの HAL ピンがスピンドル位置コマンドを実行するかを決定するだけです）

M19 は、M3、M4、M5 のいずれかによってクリアされます。

スピンドルの向きには、スピンドルシャフトの位置と回転方向を感知するためのインデックス付きの直交エンコーダが必要です。 [RS274NGC]セクションの INI 設定。

ORIENT_OFFSET = 0-360（M19 R ワードに追加された度単位の固定オフセット）

HAL ピン

- spindle.N.orient-angle (out float)M19 に必要なスピンドルの向き。 M19R ワードパラメータの値と [RS274NGC] ORIENT_OFFSETini パラメータの値。
- spindle.N.orient-mode (out s32)必要なスピンドル回転モード。 M19 P パラメータワードを反映、デフォルト = 0
- spindle.N.orient (out bit)スピンドルオリエントサイクルの開始を示します。 M19 によって設定されます。 M3、M4、M5 のいずれかによってクリアされます。 スピンドルオリエントが true のときにスピンドルオリエントフォールトがゼロでない場合、M19 コマンドはエラーメッセージで失敗します。
- spindle.N.is-oriented (in bit)スピンドル指向の確認ピンです。 オリエントサイクルを完了します。 主軸方向がアサートされたときに主軸方向が true の場合、主軸方向ピンがクリアされ、主軸ロックピンがアサートされます。 また、スピンドルブレーキピンがアサートされます。
- spindle.N.orient-fault (in s32)オリエントサイクルの障害コード入力。 ゼロ以外の値を指定すると、オリエントサイクルが中止されます。
- spindle.N.locked (out bit)スピンドルは完全なピンを方向付けます。 M3、M4、M5 のいずれかによってクリアされます。

4.4.9 M48、M49 速度および送りオーバーライド制御

- M48-スピンドル速度と送り速度のオーバーライド制御を有効にします。
- M49-両方のコントロールを無効にします。

これらのコマンドは、オプションの$パラメーターを使用して、操作するスピンドルを決定します。

コントロールがすでに有効または無効になっている場合は、コントロールを有効または無効にしても問題ありません。 詳細については、送り速度のセクションを参照してください。

4.4.10 M50 フィードオーバーライド制御

- M50 <P1>-送り速度オーバーライド制御を有効にします。 P1 はオプションです。
- M50P0-送り速度制御を無効にします。

無効になっている間、送りオーバーライドは影響を与えず、モーションはプログラムされた送り速度で実行されます。（適応送り速度オーバーライドがアクティブでない限り）。

4.4.11 M51 スピンドル速度オーバーライド制御

- M51 <P1> <$->-選択したスピンドルのスピンドル速度オーバーライド制御を有効にします。 P1 はオプションです。
- M51 P0 <$->-スピンドル速度オーバーライド制御プログラムを無効にします。 無効になっている間、スピンドル速度のオーバーライドは影響を与えず、スピンドル速度は S ワードの正確なプログラム指定値になります（スピンドル速度のセクションで説明）。

4.4.12 M52 アダプティブフィードコントロール

- M52 <P1>-アダプティブフィードを使用します。 P1 はオプションです。
- M52P0-アダプティブフィードの使用を停止します。

アダプティブフィードを有効にすると、外部入力値がユーザーインターフェイスのフィードオーバーライド値およびコマンドされたフィードレートとともに使用され、実際のフィードレートが設定されます。 LinuxCNC では、この目的のために HAL ピン motion.adaptive-feed が使用されます。 motion.adaptive-feed の値は、-1（プログラムされた逆方向の速度）から 1（フルスピード）の範囲である必要があります。 0 はフィードホールドに相当します。

Note

リバースランにネガティブアダプティブフィードを使用することは新しい機能であり、まだ十分にテストされていません。 使用目的はプラズマ切断機とワイヤースパーク侵食器ですが、そのような用途に限定されません。

4.4.13 M53 フィードストップコントロール

- M53 <P1>-送り停止スイッチを有効にします。 P1 はオプションです。 送り停止スイッチを有効にすると、送り停止制御によってモーションを中断できます。 LinuxCNC では、この目的のために HAL ピン motion.feed-hold が使用されます。 真の値を指定すると、M53 がアクティブなときにモーションが停止します。
- M53P0-フィード停止スイッチを無効にします。 M53 がアクティブでない場合、motion.feed-hold の状態はフィードに影響を与えません。

4.4.14 M61 現在のツールを設定

- M61 Q--MDI または手動モードで、工具を変更せずに現在の工具番号を変更します。 1 つの用途は、現在スピンドルにあるツールを使用して LinuxCNC の電源を入れるときに、ツールを変更せずにそのツール番号を設定できることです。

> 警告
>
> 工具長オフセットは M61 によって変更されません。M61 の後に G43 を使用して、工具長オフセットを変更してください。

次の場合はエラーになります。

- Q-が 0 以上ではありません

4.4.15 M62-M65 デジタル出力制御

- M62P--モーションに同期したデジタル出力をオンにします。 P ワードはデジタル出力番号を指定します。
- M63P--モーションに同期したデジタル出力をオフにします。 P ワードはデジタル出力番号を指定します。
- M64P--すぐにデジタル出力をオンにします。 P ワードはデジタル出力番号を指定します。
- M65P--デジタル出力をすぐにオフにします。 P ワードはデジタル出力番号を指定します。

P ワードの範囲は 0 からデフォルト値の 3 です。必要に応じて、モーションコントローラをロードするときに num_dio パラメータを使用して I / O の数を増やすことができます。 詳細については、モーションセクションを参照してください。

M62 および M63 コマンドはキューに入れられます。 同じ出力番号を参照する後続のコマンドは、古い設定を上書きします。 複数の M62 / M63 コマンドを発行することにより、複数の出力変更を指定できます。

指定された出力の実際の変更は、次のモーションコマンドの開始時に行われます。 後続のモーションコマンドがない場合、キューに入れられた出力の変更は行われません。 M62 / 63 の直後にモーション G コード（G0、G1 など）を常にプログラムすることをお勧めします。

M64 と M65 は、モーションコントローラによって受信されるとすぐに発生します。 それらは動きと同期しておらず、ブレンドを壊します。

Note

M62-65 は、適切な motion.digital-out-nn ピンが hal ファイルで出力に接続されていない限り機能しません。

4.4.16 M66 入力を待つ

```
M66 P- | E- <L->
```

- P --- 0 から 3 までのデジタル入力番号を指定します。
- E --- 0 から 3 までのアナログ入力番号を指定します。
- L--待機モードのタイプを指定します。
  - モード 0：即時-待機なし、すぐに戻ります。 入力の現在の値はパラメータ＃5399 に保存されます
  - モード 1：RISE-選択した入力が立ち上がりイベントを実行するのを待ちます。
  - モード 2：FALL-選択した入力がフォールイベントを実行するのを待ちます。
  - モード 3：HIGH-選択した入力が HIGH 状態になるのを待ちます。
  - モード 4：LOW-選択した入力が LOW 状態になるのを待ちます。
- Q--待機のタイムアウトを秒単位で指定します。 タイムアウトを超えると、待機は割り込みになり、変数＃5399 は値-1 を保持します。 L ワードがゼロ（IMMEDIATE）の場合、Q 値は無視されます。 L ワードがゼロ以外の場合、Q 値がゼロの場合はエラーになります。
- モード 0 は、アナログ入力に許可されている唯一のモードです。

M66 の例の行

```
M66 P0 L3 Q5 (wait up to 5 seconds for digital input 0 to turn on)
```

入力を待機する M66 は、選択されたイベント（またはプログラムされたタイムアウト）が発生するまで、プログラムのそれ以上の実行を停止します。

M66 を P ワードと E ワードの両方でプログラムするのはエラーです（したがって、アナログ入力とデジタル入力の両方を選択します）。 LinuxCNC では、これらの入力はリアルタイムで監視されないため、タイミングが重要なアプリケーションには使用しないでください。

モーションコントローラをロードするときに num_dio または num_aio パラメータを使用すると、I / O の数を増やすことができます。 詳細については、モーションセクションを参照してください。

Note

適切な motion.digital-in-nn ピンまたは motion.analog-in-nn ピンが hal ファイルで入力に接続されていない限り、M66 は機能しません。

HAL 接続の例

```
net signal-name motion.digital-in-00 <= parport.0.pin10-in
```

4.4.17 M67 アナログ出力、同期

M67 E- Q-

- M67-モーションに同期したアナログ出力を設定します。
- E --- 0 から 3 の範囲の出力番号。
- Q--設定する値です（オフにするには 0 に設定します）。

指定された出力の実際の変更は、次のモーションコマンドの開始時に行われます。 後続のモーションコマンドがない場合、キューに入れられた出力の変更は行われません。 M67 の直後にモーション G コード（G0、G1 など）を常にプログラムすることをお勧めします。 M67 は M62-63 と同じように機能します。

モーションコントローラをロードするときに num_dio または num_aio パラメータを使用すると、I / O の数を増やすことができます。 詳細については、モーションセクションを参照してください。

Note

適切な motion.analog-out-nn ピンが hal ファイルで出力に接続されていない限り、M67 は機能しません。

4.4.18 M68 アナログ出力、即時

M68 E- Q-

- M68-すぐにアナログ出力を設定します。
- E --- 0 から 3 の範囲の出力番号。
- Q--設定する値です（オフにするには 0 に設定します）。

M68 出力は、モーションコントローラによって受信されるとすぐに発生します。 それらは動きと同期しておらず、ブレンドを壊します。 M68 は M64-65 と同じように機能します。

モーションコントローラをロードするときに num_dio または num_aio パラメータを使用すると、I / O の数を増やすことができます。 詳細については、モーションセクションを参照してください。

Note

適切な motion.analog-out-nn ピンが hal ファイルで出力に接続されていない限り、M68 は機能しません。

4.4.19 M70 モーダル状態を保存

現在のコールレベルでモーダル状態を明示的に保存するには、M70 をプログラムします。 モーダル状態が M70 で保存されたら、M72 を実行することで正確にその状態に復元できます。

M70 命令と M72 命令のペアは、通常、サブルーチン内の不注意なモーダル変更からプログラムを保護するために使用されます。

保存された状態は次のもので構成されます。

- 現在の G20 / G21 設定（インペリアル/メートル法）
- 選択した平面（G17 / G18 / G19 G17.1、G18.1、G19.1）
- カッター補正の状態（G40、G41、G42、G41.1、G42、1）
- 距離モード-相対/絶対（G90 / G91）
- 送りモード（G93 / G94、G95）
- 現在の座標系（G54-G59.3）
- 工具長補正状態（G43、G43.1、G49）
- リトラクトモード（G98、G99）
- スピンドルモード（G96-css または G97-RPM）
- アーク距離モード（G90.1、G91.1）
- 旋盤半径/直径モード（G7、G8）
- パス制御モード（G61、G61.1、G64）
- 現在の送りと速度（F と S の値）
- スピンドルステータス（M3、M4、M5）-オン/オフと方向
- ミスト（M7）と洪水（M8）のステータス
- 速度オーバーライド（M51）およびフィードオーバーライド（M50）設定
- アダプティブフィード設定（M52）
- フィードホールド設定（M53）

特に、モーションモード（G1 など）は復元されないことに注意してください。

現在の通話レベルは、次のいずれかを意味します。

- メインプログラムで実行します。 メインプログラムレベルでの状態の単一の保存場所があります。 複数の M70 命令が順番に実行された場合、M72 が実行されると、最後に保存された状態のみが復元されます。
- G コードサブルーチン内で実行します。 サブルーチン内で M70 を使用して保存された状態は、ローカルの名前付きパラメーターとまったく同じように動作します。M72 を使用したこのサブルーチン呼び出し内でのみ参照でき、サブルーチンが終了するとパラメーターは消えます。

サブルーチンを再帰的に呼び出すと、新しい呼び出しレベルが導入されます。

4.4.20 M71 保存されたモーダル状態を無効にする

現在の通話レベルで M70 または M73 によって保存されたモーダル状態は無効になります（これ以上復元できません）。

同じコールレベルの後続の M72 は失敗します。

M73 によってモーダル状態を保護するサブルーチンで実行された場合、後続の return または endsub はモーダル状態を復元しません。 この機能の有用性は疑わしいです。 なくなる可能性があるため、信頼しないでください。

4.4.21 M72 モーダル状態の復元

M70 コードで保存されたモーダル状態は、M72 を実行することで復元できます。

G20 / G21 の処理は、フィードが G20 / G21 に応じて異なる方法で解釈されるため、特別に扱われます。復元操作によって長さの単位（mm / in）が変更されようとしている場合、「M72」は最初に距離モードを復元し、次に フィード値が正しい単位設定で解釈されることを確認するためのフィードを含む他のすべての状態。

そのレベルで以前の M70 保存操作なしで M72 を実行するとエラーになります。

次の例は、M70 および M72 を使用して、サブルーチン呼び出しの周囲のモーダル状態を保存し、明示的に復元する方法を示しています。 imperialsub サブルーチンは、M7x 機能を「認識」しておらず、変更せずに使用できることに注意してください。

```
O<showstate> sub
(DEBUG, imperial=#<_imperial> absolute=#<_absolute> feed=#<_feed> rpm=#<_rpm>)
O<showstate> endsub
O<imperialsub> sub
g20 (imperial)
g91 (relative mode)
F5 (low feed)
S300 (low rpm)
(debug, in subroutine, state now:)
o<showstate> call
O<imperialsub> endsub
; main program
g21 (metric)
g90 (absolute)
f200 (fast speed)
S2500 (high rpm)
(debug, in main, state now:)
o<showstate> call
M70 (save caller state in at global level)
O<imperialsub> call
M72 (explicitely restore state)
(debug, back in main, state now:)
o<showstate> call
m2
```

4.4.22 M73 モーダル状態の保存と自動復元

サブルーチン内でモーダル状態を保存し、サブルーチン endsub または任意のリターンパスで状態を復元するには、M73 をプログラムします。 M73 操作のあるサブルーチンで実行中のプログラムを中止しても、状態は復元されません。

また、M73 を含むメインプログラムのノーマルエンド（M2）は状態を復元しません。

推奨される使用法は、次の例のように O-word サブルーチンの先頭にあります。 このように M73 を使用すると、モーダル状態を変更する必要があるが、不注意によるモーダル変更から呼び出し側プログラムを保護するサブルーチンの設計が可能になります。 showstate サブルーチンで事前定義された名前付きパラメーターを使用していることに注意してください。

```
O<showstate> sub
(DEBUG, imperial=#<_imperial> absolute=#<_absolute> feed=#<_feed> rpm=#<_rpm>)
O<showstate> endsub
O<imperialsub> sub
M73 (save caller state in current call context, restore on return or endsub)
g20 (imperial)
g91 (relative mode)
F5 (low feed)
S300 (low rpm)
(debug, in subroutine, state now:)
o<showstate> call
; note - no M72 is needed here - the following endsub or an
; explicit 'return' will restore caller state
O<imperialsub> endsub
; main program
g21 (metric)
g90 (absolute)
f200 (fast speed)
S2500 (high rpm)
(debug, in main, state now:)
o<showstate> call
o<imperialsub> call
(debug, back in main, state now:)
o<showstate> call
m2
```

4.4.23 M98 および M99

インタプリタは、M98 および M99M コードでファナックスタイルのメインプログラムとサブプログラムをサポートします。 ファナックスタイルのプログラムを参照してください。

1.1.1.1 モーダル状態を選択的に復元する

M72 を実行するか、M73 を含むサブルーチンから戻ると、保存されているすべてのモーダル状態が復元されます。

モーダル状態の一部の側面のみを保持する必要がある場合は、事前定義された名前付きパラメーター、ローカルパラメーター、および条件ステートメントを使用することもできます。 アイデアは、サブルーチンの開始時に復元されるモードを記憶し、終了する前にこれらを復元することです。 nc_files /tool-length-probe.ngc のスニペットに基づく例を次に示します。

```
#<absolute> = #<_absolute> (remember in local variable if G90 was set)
;
g30 (above switch)
g38.2 z0 f15 (measure)
g91 g0z.2 (off the switch)
#1000=#5063 (save reference tool length)
(print,reference length is #1000)
;
O<restore_abs> if [#<absolute>]
g90 (restore G90 only if it was set on entry:)
O<restore_abs> endif
;
O<measure> endsub
```

4.4.24 M100-M199 ユーザー定義コマンド

```
M1-- <P- Q->
```

- M1- --- 100～199 の範囲の整数。
- P ---最初のパラメータとしてファイルに渡される数値。
- Q --- 2 番目のパラメーターとしてファイルに渡される数値。

Note

新しい M1nn ファイルを作成した後、GUI を再起動して、新しいファイルを認識できるようにする必要があります。そうしないと、不明な m コードエラーが発生します。

M100 から M199（拡張子なしおよびキャピトル M）という名前の外部プログラムは、オプションの P 値と Q 値を 2 つの引数として実行されます。 G コードファイルの実行は、外部プログラムが終了するまで一時停止します。 任意の有効な実行可能ファイルを使用できます。 ファイルは、ini ファイル構成で指定された検索パスに配置する必要があります。 検索パスの詳細については、表示セクションを参照してください。

警告

ファイルの作成または編集にワードプロセッサを使用しないでください。 ワードプロセッサは、問題を引き起こし、bash または python ファイルが機能しなくなる可能性のある目に見えないコードを残します。 Ubuntu の Gedit や他のオペレーティングシステムの Notepad ++などのテキストエディタを使用して、ファイルを作成または編集します。

使用されたエラー不明の M コードは、次のいずれかを示します

- 指定されたユーザー定義コマンドは存在しません
- ファイルは実行可能ファイルではありません
- ファイル名の拡張子は
- ファイル名は、この形式 M1nn に従いません。ここで、nn = 00 から 99 です。
- ファイル名は小文字の M を使用しました

たとえば、M101 および M102 を使用する bash スクリプトファイルを使用して、パラレルポートピンによって制御されるコレットクローザーを開閉します。 M101 および M102 という名前の 2 つのファイルを作成します。 LinuxCNC を実行する前に、それらを実行可能ファイル（通常は右クリック/プロパティ/権限）として設定します。 パラレルポートピンが HAL ファイル内の何にも接続されていないことを確認してください。

M101 サンプルファイル

```
#!/bin/bash
# file to turn on parport pin 14 to open the collet closer
halcmd setp parport.0.pin-14-out True
exit 0
```

M102 サンプルファイル

```
#!/bin/bash
# file to turn off parport pin 14 to open the collet closer
halcmd setp parport.0.pin-14-out False
exit 0
```

変数を M1nn ファイルに渡すには、次のように P および Q オプションを使用します。

```
M100 P123.456 Q321.654
```

M100 サンプルファイル

```
#!/bin/bash
voltage=$1
feedrate=$2
halcmd setp thc.voltage $voltage
halcmd setp thc.feedrate $feedrate
exit 0
```

グラフィックメッセージを表示し、メッセージウィンドウが閉じるまで停止するには、Eye ofGnome などのグラフィック表示プログラムを使用してグラフィックファイルを表示します。 それを閉じると、プログラムが再開されます。

M110 サンプルファイル

```
#!/bin/bash
eog /home/john/linuxcnc/nc_files/message.png
exit 0
```

グラフィックメッセージを表示し、G コードファイルのサフィックスの処理を続行するには、コマンドのアンパサンドを付けます。

M110 例の表示と続行

```
#!/bin/bash
eog /home/john/linuxcnc/nc_files/message.png &
exit 0
```

4.5 O コード

O コードは、NC プログラムのフロー制御を提供します。 各ブロックには、O の後に使用される番号である関連付けられた番号があります。O 番号と正しく一致するように注意する必要があります。 O コードは、O100 や o100 のように、数字の最初の文字として数字のゼロではなく文字 O を使用します。

4.5.1 ナンバリング

番号付き O コードには、サブルーチンごとに一意の番号が必要です。番号付けの例

```
(the start of o100)
o100 sub
```

```
(notice that the if-endif block uses a different number)
(the start of o110)
o110 if [#2 GT 5]
(some code here)
(the end of o110)
o110 endif
(some more code here)
(the end of o100)
o100 endsub
```

4.5.2 コメント

O ワードと同じ行のコメントは、動作が将来変更される可能性があるため、使用しないでください。

次の場合、動作は未定義です。

- 同じ番号が複数のブロックに使用されている
- 他の単語は O-単語のある行で使用されます
- コメントは O-word のある行で使用されます

> Note
>
> 小文字の o を使用すると、タイプミスした可能性のある 0 と区別しやすくなります。 たとえば、o100 は O100 よりも 0 ではないことがわかりやすいです。

4.5.3 サブルーチン

サブルーチンは Onnnsub で始まり、Onnnendsub で終わります。 Onnnsub と Onnnendsub の間の行は、サブルーチンが Onnn 呼び出しで呼び出されるまで実行されません。 各サブルーチンは一意の番号を使用する必要があります。

サブルーチンの例

```
o100 sub
G53 G0 X0 Y0 Z0 (rapid move to machine home)
o100 endsub
(the subroutine is called)
o100 call
M2
```

詳細については、G53＆G0＆M2 セクションを参照してください。

O- Return サブルーチン内で、O-return を実行できます。 これは、O- endsub が検出されたかのように、すぐに呼び出し元のコードに戻ります。

O-戻り例

```
o100 sub
(test if parameter #2 is greater than 5)
o110 if [#2 GT 5]
(return to top of subroutine if test is true)
o100 return
o110 endif
(this only gets executed if parameter #2 is not greater than 5)
(DEBUG, parameter 1 is [#1])
o100 endsub
```

詳細については、「二項演算子とパラメーター」セクションを参照してください。

O-呼び出し O-呼び出しは最大 30 個のオプションの引数を取り、それらは＃1、＃2、としてサブルーチンに渡されます。 。 。 、 ＃NS。 ＃N + 1 から＃30 までのパラメーターは、呼び出し元のコンテキストと同じ値になります。 サブルーチンから戻ると、パラメーター＃1 から＃30 の値（引数の数に関係なく）は、呼び出し前の値に復元されます。 パラメータ＃1-＃30 は、サブルーチンに対してローカルです。

1 2 3 は数値 123 として解析されるため、パラメーターは角括弧で囲む必要があります。 以下は、3 つの引数を持つサブルーチンを呼び出します。

O-呼び出しの例

```
o100 sub
(test if parameter #2 is greater than 5)
o110 if [#2 GT 5]
(return to top of subroutine if test is true)
o100 return
o110 endif
(this only gets executed if parameter #2 is not greater than 5)
(DEBUG, parameter 1 is [#1])
(DEBUG, parameter 3 is [#3])
o100 endsub
o100 call [100] [2] [325]
```

サブルーチン本体はネストできません。 それらは、定義された後にのみ呼び出すことができます。 それらは他の関数から呼び出される場合があり、そうすることが理にかなっている場合は再帰的に呼び出すことができます。 サブルーチンの最大ネストレベルは 10 です。

サブルーチンは＃30 を超えるパラメーターの値を変更でき、それらの変更は呼び出し元のコードに表示されます。 サブルーチンは、グローバルな名前付きパラメーターの値を変更する場合もあります。

1.1.1.1 ファナックスタイルの番号付きプログラム

番号付きプログラム（メインプログラムとサブプログラムの両方）、M98 呼び出しおよび M99 は M コードを返し、それぞれのセマンティックの違いは、ファナックおよび他のマシンコントローラーとの互換性のために提供される上記の rs274ngc サブルーチンの代替です。

番号付きプログラムはデフォルトで有効になっており、.ini ファイルの[RS274NGC]セクションに DISABLE_FANUC_STYLE_SUB = 1 を配置することで無効にできます。

Note

番号付きのメインおよびサブプログラムの定義と呼び出しは、構文と実行の両方で従来の rs274ngc とは異なります。 混乱の可能性を減らすために、あるスタイルの定義が別のスタイルの呼び出しと混合されている場合、インタープリターはエラーを発生させます。

番号付きサブプログラムの簡単な例

```
o1 (Example 1) ; Main program 1, "Example 1"
M98 P100 ; Call subprogram 100
M30 ; End main program
o100 ; Beginning of subprogram 100
G53 G0 X0 Y0 Z0 ; Rapid move to machine home
M99 ; Return from subprogram 100
```

o1（タイトル）オプションのメインプログラム開始ブロックは、メインプログラムに番号 1 を与えます。一部のコントローラーは、オプションの次の括弧で囲まれたコメントをプログラムタイトルとして扱います。t の例 1 は例ですが、これは rs274ngc インタープリターでは特別な意味はありません。

M98 P- <L \->番号付きサブプログラムを呼び出します。 ブロック M98P100 は、従来の o100 呼び出し構文に類似していますが、o100 で定義された次の番号付きサブプログラムを呼び出すためにのみ使用できます。 。 .M99。 オプションの L ワードは、ループカウントを指定します。

M30 メインプログラムは、M02 または M30（または M99、以下を参照）で終了する必要があります。

O-サブプログラム定義の開始番号付きサブプログラム定義の開始をマークします。 ブロック O100 は o100sub に似ていますが、M98P100 呼び出しブロックよりもファイルの後半に配置する必要がある点が異なります。

番号付きサブルーチンからの M99 リターンブロック M99 は、従来の o100 endsub 構文に類似していますが、番号付きプログラム（この例では o100）のみを終了でき、o100sub 構文で始まるサブルーチンを終了することはできません。

M98 サブプログラム呼び出しは、次の点で rs274ngcO 呼び出しとは異なります。

- 番号付きサブプログラムは、プログラムファイルの M98 呼び出しに続く必要があります。 サブプログラムが呼び出しブロックの前にある場合、インタプリタはエラーをスローします。

- パラメータ＃1、＃2、。。。。、＃30 はグローバルであり、番号付きサブプログラムでアクセスできます。これは、従来のスタイルの呼び出しで番号が大きいパラメーターと同様です。 サブプログラム内のこれらのパラメーターへの変更はグローバルな変更であり、サブプログラムが戻った後も保持されます。
- M98 サブプログラム呼び出しには戻り値がありません。
- M98 サブプログラム呼び出しブロックには、ループの繰り返し回数を指定するオプションの L ワードを含めることができます。 L ワードがない場合、サブプログラムは 1 回だけ実行されます（M98 L1 と同等）。 M98L0 ブロックはサブプログラムを実行しません。

まれに、M99 M コードを使用してメインプログラムを終了することがあります。これは、エンドレスプログラムを示します。 インタプリタがメインプログラムの M99 に到達すると、ファイルの先頭にスキップして戻り、最初の行から実行を再開します。 エンドレスプログラムの使用例は、マシンのウォームアップサイクルです。 ブロック削除プログラム終了/ M30 ブロックは、オペレーターの準備ができたときの整頓された時点でサイクルを停止するために使用される場合があります。

番号付きサブプログラムの完全な例

```
O1 ; Main program 1
#1 = 0
(PRINT,X MAIN BEGIN: 1=#1)
M98 P100 L5 ; Call subprogram 100
(PRINT,X MAIN END: 1=#1)
M30 ; End main program
O100 ; Subprogram 100
#1 = [#1 + 1]
M98 P200 L5 ; Call subprogram 200
(PRINT,>> O100: #1)
M99 ; Return from Subprogram 100

O200 ; Subprogram 200
#1 = [#1 + 0.01]
(PRINT,>>>> O200: #1)
M99 ; Return from Subprogram 200
```

この例では、パラメーター＃1 が 0 に初期化されています。サブプログラム O100 は、ループ内で 5 回呼び出されます。 O100 への各呼び出し内にネストされたサブプログラム O200 は、ループ内で 5 回呼び出され、合計で 25 回呼び出されます。

パラメータ＃1 はグローバルであることに注意してください。 メインプログラムの終了時に、O100 および O200 内で更新した後、その値は 5.25 になります。

4.5.4 ループ

while ループには、while / endwhile と do / while の 2 つの構造があります。 いずれの場合も、while 条件が false と評価されると、ループは終了します。 違いは、テスト条件が実行されるときです。 do / while ループは、ループ内のコードを実行してから、テスト条件をチェックします。 while / endwhile ループが最初にテストを実行します。

While Endwhile の例

```
(draw a sawtooth shape)
G0 X1 Y0 (move to start position)
#1 = 0 (assign parameter #1 the value of 0)
F25 (set a feed rate)
o101 while [#1 LT 10]
G1 X0
G1 Y[#1/10] X1
#1 = [#1+1] (increment the test counter)
o101 endwhile
M2 (end program)
```

DoWhile の例

```
#1 = 0 (assign parameter #1 the value of 0)
o100 do
(debug, parameter 1 = #1)
o110 if [#1 EQ 2]
#1 = 3 (assign the value of 3 to parameter #1)
(msg, #1 has been assigned the value of 3)
o100 continue (skip to start of loop)
o110 endif
(some code here)
#1 = [#1 + 1] (increment the test counter)
o100 while [#1 LT 3]
(msg, Loop Done!)
M2
```

while ループ内では、O-break はすぐにループを終了し、O-continue はすぐに while 条件の次の評価にスキップします。 それでも当てはまる場合は、ループが再び先頭から始まります。 false の場合、ループを終了します。

4.5.5 条件付き

if 条件は、if で始まり endif で終わる同じ o 番号のステートメントのグループで構成されます。 オプションの elseif および else 条件は、開始 if と終了 endif の間にある場合があります。

if 条件が true と評価された場合、if から次の条件行までのステートメントのグループが実行されます。

if 条件が false と評価された場合、elseif 条件は true と評価されるまで順番に評価されます。 elseif 条件が真の場合、次の条件行までの elseif に続くステートメントが実行されます。 if または elseif 条件のいずれも true と評価されない場合、else に続くステートメントが実行されます。 条件が true と評価されると、グループ内でそれ以上の条件は評価されません。

If Endif の例の場合

```
(if parameter #31 is equal to 3 set S2000)
o101 if [#31 EQ 3]
S2000
o101 endif
```

If ElseIf ElseEndIf の例

```
(if parameter #2 is greater than 5 set F100)
o102 if [#2 GT 5]
F100
o102 elseif [#2 LT 2]
(else if parameter #2 is less than 2 set F200)
F200
(else if parameter #2 is 2 through 5 set F150)
o102 else
F150
o102 endif
```

上記のすべての条件が false の場合、else パスが最終的に実行されるまで、elseif ステートメントによっていくつかの条件がテストされる場合があります。

If Elseif Else Endiff の例

```
(if parameter #2 is greater than 5 set F100)
O102 if [#2 GT 5]
F100
(else if parameter #2 less than 2 set F200)
O102 elseif [#2 LT 2]
F20
```

```
(parameter #2 is between 2 and 5)
O102 else
F200
O102 endif
```

4.5.6 繰り返す

リピートは、指定された回数、リピート/エンドリピート内のステートメントを実行します。 この例は、現在の位置から開始して斜めの一連の形状をミリングする方法を示しています。

繰り返すの例

```
(Mill 5 diagonal shapes)
G91 (Incremental mode)
o103 repeat [5]
... (insert milling code here)
G0 X1 Y1 (diagonal move to next position)
o103 endrepeat
G90 (Absolute mode)
```

4.5.7 間接参照

O 番号は、パラメータおよび/または計算によって与えることができます。

間接参照の例

```
o[#101+2] call
```

O ワードでの値の計算値の計算の詳細については、次のセクションを参照してください。

- パラメーター
- 式
- 二項演算子
- 関数

4.5.8 ファイルの呼び出し

サブルーチン名を使用して別のファイルを呼び出すには、ファイルに呼び出しと同じ名前を付け、ファイルに sub と endsub を含めます。 ファイルは、ini ファイルの PROGRAM_PREFIX または SUBROUTINE_PATH が指すディレクトリにある必要があります。 ファイル名には、小文字、数字、ダッシュ、およびアンダースコアのみを含めることができます。 名前付きサブルーチンファイルには、単一のサブルーチン定義のみを含めることができます。

名前付きファイルの例

```
o<myfile> call
```

番号付きファイルの例

```
o123 call
```

呼び出されたファイルには、oxxx sub と endsub を含める必要があり、ファイルは有効なファイルである必要があります。

呼び出されたファイルの例

```
(filename myfile.ngc)
o<myfile> sub
(code here)
o<myfile> endsub
M2
```

Note

ファイル名は小文字のみであるため、o <MyFile>はインタプリタによって o <myfile>に変換されます。 検索パスと検索パスのオプションの詳細については、INI 構成セクションを参照してください。

4.5.9 サブルーチンの戻り値

サブルーチンは、オプションで、endsub または return ステートメントのオプションの式によって値を返すことができます。

戻り値の例

```
o123 return [#2 *5]
...
o123 endsub [3 * 4]
```

サブルーチンの戻り値は、<_ value>の事前定義された名前付きパラメーターに格納され、<_ value_returned>の事前定義されたパラメーターは 1 に設定され、値が返されたことを示します。 両方のパラメーターはグローバルであり、次のサブルーチン呼び出しの直前にクリアされます。

4.5.10 エラー

次のステートメントはエラーメッセージを表示し、インタープリターを中止します。

- サブ定義内にない return または endsub
- 他の場所で定義されている繰り返しのラベル
- 他の場所で定義されており、do を参照していない while のラベル
- 他の場所で定義されている場合のラベル

- else または elseif の未定義のラベル
- else のラベル、elseif または endif が一致する if を指していない
- 一致する while または do を指さない break または continue のラベル
- endrepeat または endwhile のラベルは、対応する while または repeat を参照していません

これらのエラーを stderr で致命的でない警告にするには、[RS274NGC] FEATURE = maskini オプションでビット 0x20 を設定します。

4.6 その他のコード

4.6.1 F：送り速度を設定する

Fx-送り速度を x に設定します。 x は通常、1 分あたりのマシン単位（インチまたはミリメートル）です。

逆時間送り速度モードまたは 1 回転あたりの送りモードが有効である場合を除き、送り速度の適用は送り速度セクションで説明されているとおりです。有効な場合、送り速度は G93 G94G95 セクションで説明されているとおりです。

4.6.2 S：スピンドル速度を設定します

Sx [$ n]-スピンドルの速度を x 回転/分（RPM）に設定し、オプションの$は特定のスピンドルのスピンドル速度を設定します。 $がないと、コマンドはデフォルトでスピンドルになります。0

M3 または M4 が有効な場合、スピンドルまたは選択したスピンドルはその速度で回転します。 主軸が回転しているかどうかに関係なく、S ワードをプログラムしても問題ありません。 速度オーバーライドスイッチが有効で、100％に設定されていない場合、速度はプログラムされたものとは異なります。

S0 をプログラムしても問題ありません。プログラムしてもスピンドルは回転しません。

次の場合はエラーになります。

- S 数は負です。

4.6.3 T：ツールを選択

Tx-ツール x に変更する準備をします。

M6 がプログラムされるまで、ツールは変更されません（セクション M6 を参照）。 T ワードは、M6 と同じ行または前の行に表示される場合があります。 ツールを変更せずに T ワードが 2 行以上表示されていても問題ありません。 次の工具交換では、最新の T ワードのみが有効になります。

Note

LinuxCNC が非ランダムツールチェンジャー用に構成されている場合（EMCIO セクションの RANDOM_TOOLCHANGER のエントリを参照）、T0 は特別な処理を受けます。ツールは選択されません。 これは、工具交換後にスピンドルを空にしたい場合に役立ちます。

Note

LinuxCNC がランダムツールチェンジャー用に構成されている場合（EMCIO セクションの RANDOM_TOOLCHANGER のエントリを参照）、T0 は特別な扱いを受けません。T0 は他のツールと同様に有効なツールです。 ランダムなツールチェンジャーマシンで T0 を使用して空のポケットを追跡するのが通例です。これにより、ランダムでないツールチェンジャーマシンのように動作し、スピンドルをアンロードします。

次の場合はエラーになります。

- 負の T 数が使用され、
- ツールテーブルファイルに表示されない T 番号が使用されます（上記のように、ランダムでないツールチェンジャーの T0 が受け入れられることを除いて）。

一部のマシンでは、T ワードがプログラムされると、カルーセルが移動し、同時に機械加工が行われます。 このようなマシンでは、工具交換の数行前に T ワードをプログラミングすると、時間を節約できます。 このようなマシンの一般的なプログラミング手法は、工具交換後に使用する次の工具の T ワードをラインに配置することです。 これにより、カルーセルの移動に利用できる時間が最大化されます。

T <n>後の高速移動は、フィード移動後まで AXIS プレビューに表示されません。 これは、長距離を移動して旋盤のように工具を交換する機械用です。 これは最初は非常に混乱する可能性があります。 現在のツールプログラムでこの機能をオフにするには、T <n>の後に移動せずに G1 をプログラムします。

4.7 G コード例

LinuxCNC をインストールすると、いくつかのサンプルファイルが/ nc_files フォルダーに配置されます。 実行する前に、サンプルファイルがマシンに適していることを確認してください。

4.7.1 ミルの例

1.1.1.1 らせん穴フライス

ファイル名：役に立つ-subroutines.ngc

説明：パラメータを使用して穴をフライス盤するためのサブルーチン。

4.7.1.1 スロット

ファイル名：役に立つ-subroutines.ngc

説明：パラメータを使用してスロットをフライス盤するためのサブルーチン。

4.7.1.2 グリッドプローブ

ファイル名：gridprobe.ngc

説明：長方形プロービング

このプログラムは、通常の XY グリッドを繰り返しプローブし、プローブされた場所を.ini ファイルと同じディレクトリにある probe-results.txt ファイルに書き込みます。

4.7.1.3 スマートプローブ

ファイル名：smartprobe.ngc

説明：長方形プロービング

このプログラムは、通常の XY グリッドを繰り返しプローブし、プローブされた場所を.ini ファイルと同じディレクトリにある probe-results.txt ファイルに書き込みます。 これは、グリッドプローブファイルから改善されています。

4.7.1.4 工具長プローブ

ファイル名：tool-length-probe.ngc

説明：工具長プロービング

このプログラムは、プローブ入力に接続されたスイッチを使用して工具長を自動的に測定する方法の例を示しています。 これは、工具を挿入するたびに工具の長さが異なる工具ホルダーのない機械に役立ちます。

4.7.1.5 ホールプローブ

ファイル名：probe-hole.ngc

説明：穴の中心と直径を見つけます。

プログラムは、穴の中心を見つけ、穴の直径を測定し、結果を記録する方法を示します。

4.7.1.6 カッター補正

追加される

4.7.2 旋盤の例

1.1.1.1 Threading

ファイル名 lathe-g76.ngc

説明：フェーシング、スレッディング、およびパーティングオフ。

このファイルは、パラメータを使用した旋盤でのねじ切りの例を示しています。

4.8 RS274 / NGC の違い

4.8.1 RS274 / NGC からの変更

RS274 / NGC プログラムの意味を変える違い

工具交換後の位置

工具交換後の位置

LinuxCNC では、工具交換後に機械が元の位置に戻らない。 この変更は、新しいツールが古いツールよりも長くなる可能性があり、元のマシン位置に移動するとツールチップが低くなりすぎる可能性があるために行われました。

オフセットパラメータは ini ファイル単位です

LinuxCNC では、G28 および G30 のホームロケーションである P1 のパラメーターに格納されている値。 。 。 P9 座標系、および G92 オフセットは「ini ファイル単位」です。 この変更が行われたのは、G28、G30、G10 L2、または G92.3 がプログラムされているときに、G20 または G21 がアクティブであったかどうかによって、場所の意味が変わったためです。

ツールテーブルの長さ/直径は ini ファイル単位です

LinuxCNC では、ツールテーブルのツールの長さ（オフセット）と直径は、ini ファイル単位でのみ指定されます。 この変更が行われたのは、G43、G41、G42 モードを開始するときに、G20 または G21 のどちらがアクティブであったかに基づいて、ツールの長さと直径が変わるためです。 これにより、G コードが単純で整形式（G20 または G21 から始まり、プログラム全体で単位を変更しなかった場合）でも、ツールテーブルを変更せずに、マシンの非ネイティブ単位で G コードを実行することは不可能でした。 。

G84、G87 は実装されていません

G84 および G87 は現在実装されていませんが、LinuxCNC の将来のリリースに追加される可能性があります。

軸ワード付きの G28、G30

G28 または G30 が一部の軸ワードのみが存在するようにプログラムされている場合、LinuxCNC は指定された軸のみを移動します。 これは他のマシンコントロールでは一般的です。 一部の軸を中間点に移動してから、すべての軸を事前定義された点に移動するには、次の 2 行の G コードを記述します。

G0 X- Y-（中間点に移動する軸）G28（すべての軸を事前定義された点に移動）

4.8.2 RS274 / NGC への追加

RS274 / NGC プログラムの意味を変えない違い

G33、G76 スレッドコード

これらのコードは RS274 / NGC では定義されていません。

G38.2

G38.2 を動かした後、プローブの先端が引っ込められません。 この撤回の動きは、LinuxCNC の将来のリリースで追加される可能性があります。

G38.3. . . G38.5

これらのコードは RS274 / NGC では定義されていません

O コード

これらのコードは RS274 / NGC では定義されていません

M50。 。 .M53 オーバーライド

これらのコードは RS274 / NGC では定義されていません

M61..M66

これらのコードは RS274 / NGC では定義されていません

G43、G43.1

負の工具長

RS274 / NGC 仕様では、すべての工具長が正になると「予想されます」と記載されています。 ただし、G43 は負の工具長に対して機能します。

旋盤工具

G43 工具長補正は、X 寸法と Z 寸法の両方で工具をオフセットできます。 この機能は主に旋盤で役立ちます。

動的工具長

LinuxCNC では、G43.1 IK を介して計算された工具長を指定できます。

G41.1、G42.1

LinuxCNC では、工具の直径と、旋盤モードの場合は G コードでの方向を指定できます。 形式は G41.1 / G42.1 D L です。ここで、D は直径、L（指定されている場合）は旋盤工具の向きです。

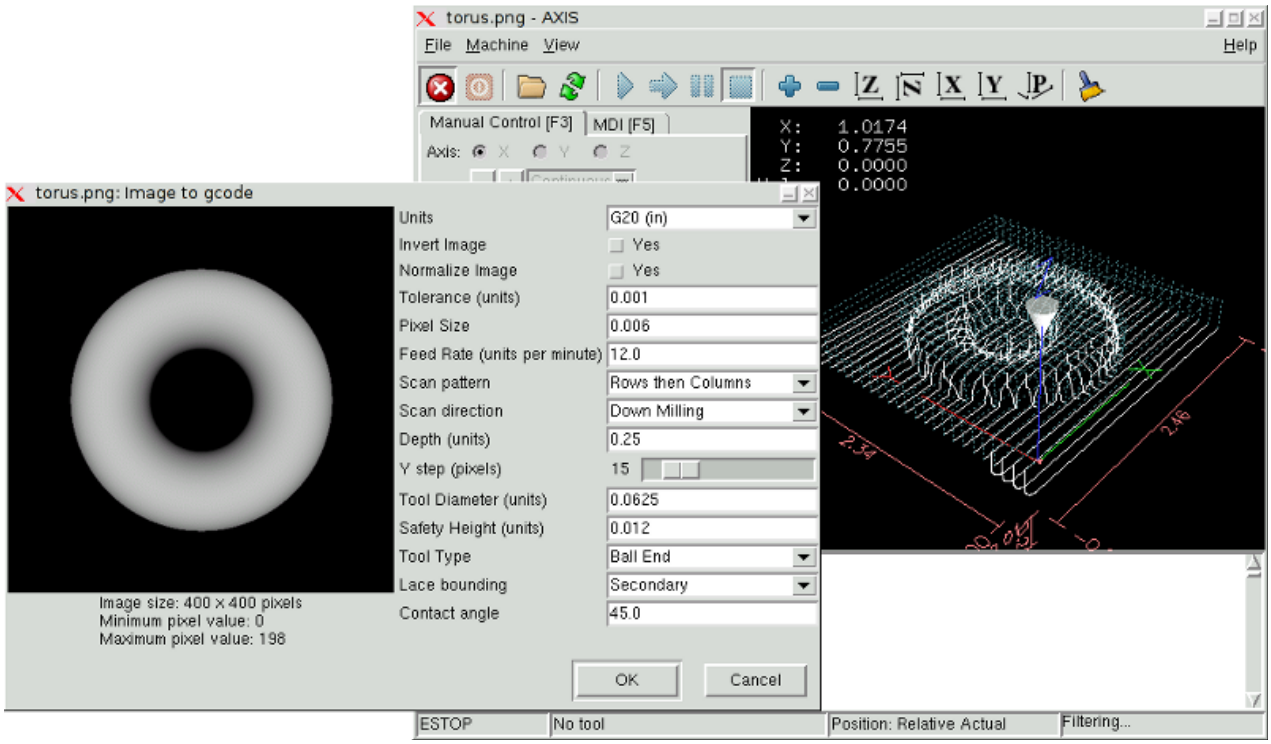
H ワードなしの G43

ngc では、これは許可されていません。 LinuxCNC では、現在ロードされているツールの長さオフセットを設定します。 現在ロードされているツールがない場合は、エラーです。 この変更は、ユーザーが工具交換ごとに 2 箇所で工具番号を指定する必要がないように、また D ワードが指定されていない場合の G41 / G42 の動作と一貫しているために行われました。

U、V、および W 軸

LinuxCNC では、U、V、W と呼ばれる 3 つの直線軸の追加セットを定義することにより、最大 9 軸のマシンを使用できます。

4.9 G コードへの画像



4.9.1 デプスマップとは何ですか？

デプスマップは、各ピクセルの明るさが各ポイントでのオブジェクトの深度（または高さ）に対応するグレースケール画像です。

4.9.2 image-to-gcode と AXIS ユーザーインターフェイスの統合

.ini ファイルの[FILTER]セクションに次の行を追加して、.png、.gif、または.jpg 画像を開いたときに AXIS が自動的に image-to-gcode を呼び出すようにします。

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

標準の sim / axis.ini 構成ファイルはすでにこの方法で構成されています。

4.9.3 image-to-gcode の使用

次のように、AXIS でイメージファイルを開くか、ターミナルから image-to-gcode を呼び出すことにより、image-to-gcode を開始します。

```
image-to-gcode torus.png > torus.ngc
```

右側の列のすべての設定を確認し、[OK]を押して gcode を作成します。 選択した画像サイズとオプションに応じて、これには数秒から数分かかる場合があります。 AXIS に画像を読み込んでいる場合、画像から gcode への変換が完了すると、gcode が自動的に読み込まれ、プレビューされます。 AXIS では、リロードを押すと、画像から gcode へのオプション画面が再び表示され、微調整が可能になります。

4.9.4 オプションリファレンス

1.1.1.1 単位

生成された g コードで G20（インチ）または G21（mm）のどちらを使用するか、およびラベル付けされた各オプションの単位（単位）として使用するかどうかを指定します。

4.9.4.1 画像を反転

「いいえ」の場合、黒いピクセルが最低点で、白いピクセルが最高点です。 「はい」の場合、黒いピクセルが最高点で、白いピクセルが最低点です。

4.9.4.2 画像の正規化

はいの場合、最も暗いピクセルが黒に再マップされ、最も明るいピクセルが白に再マップされます。

4.9.4.3 画像の境界線を展開

None の場合、入力画像はそのまま使用され、画像の端にある詳細が切り取られる可能性があります。 白または黒の場合、ツールの直径に等しいピクセルの境界線がすべての側面に追加され、画像の端にある詳細が切り取られることはありません。

4.9.4.4 許容値（単位）

一連の点が直線の許容範囲内にある場合、それらは直線として出力されます。 許容値を大きくすると、LinuxCNC の輪郭のパフォーマンスが向上する可能性がありますが、画像の細部を削除したりぼかしたりすることもできます。

4.9.4.5 ピクセルサイズ（単位）

入力画像の 1 ピクセルは、これだけの数の単位になります。通常、この数は 1.0 よりはるかに小さくなります。 たとえば、400x400 の画像ファイルから 2.5x2.5 インチのオブジェクトをミリングするには、2.5 / 400 = .00625 であるため、ピクセルサイズ.00625 を使用します。

4.9.4.6 プランジ送り速度（1 分あたりの単位）

プランジパッキング速度（1 分プレーヤーの単位）

4.9.4.7 送り速度（1 分あたりの単位）

パスの他の部分の送り速度。

4.9.4.8 スピンドル速度（RPM）

gcode ファイルに入れる必要のあるスピンドル速度 S コード。

4.9.4.9 スキャンパターン

可能なスキャンパターンは次のとおりです。

- 行
- 列
- 行、次に列
- 列、次に行

4.9.4.10 スキャン方向

可能なスキャン方向は次のとおりです。

- 正：低いX軸またはY軸の値でフライス盤を開始し、高いX軸またはY軸の値に向かって移動します
- 負：高いX軸またはY軸の値でフライス盤を開始し、低いX軸またはY軸の値に向かって移動します
- 交互：最後の移動が終了したX軸またはY軸の移動の同じ端から開始します。 これにより、トラバースの動きの量が減少します
- アップミリング：低いポイントからミリングを開始し、高いポイントに向かって移動します
- ダウンミリング：高いポイントからミリングを開始し、低いポイントに向かって移動します

4.9.4.11 深さ（単位）

材料の上部は常にZ = 0にあります。 材料への最も深い切り込みはZ =-深さです。

4.9.4.12 ステップオーバー（ピクセル）

隣接する行または列間の距離。 特定の単位距離のピクセル数を見つけるには、距離/ピクセルサイズを計算し、最も近い整数に丸めます。 たとえば、ピクセルサイズ= .006で、目的のステップオーバー距離= .015の場合、.015 / .006 = 2.5 ’であるため、2または3ピクセルのステップオーバーを使用します。

4.9.4.13 工具径

工具の切削部分の直径。

4.9.4.14 安全高さ

トラバース移動のために移動する高さ。 image-to-gcodeは、常にマテリアルの上部がZ = 0にあることを前提としています。

4.9.4.15 ツールタイプ

工具の切削部分の形状。 可能な工具形状は次のとおりです。

- ボールエンド
- フラットエンド
- 45度の「ビー」
- 60度の「ビー」

4.9.4.16 レースバウンディング

これは、行または列に沿って比較的平坦な領域をスキップするかどうかを制御します。 このオプションは、行と列の両方がミリングされている場合にのみ意味があります。 可能な境界オプションは次のとおりです。

- なし：行と列は両方とも完全にフライス盤で削られています。
- 二次：第 2 方向にフライス盤を使用する場合、その方向に大きく傾斜していない領域はスキップされます。
- フル：第 1 方向にフライス盤を使用する場合、第 2 方向に大きく傾斜する領域はスキップされます。 2 番目の方向にフライス盤を作成する場合、その方向に大きく傾斜していない領域はスキップされます。

4.9.4.17 接触角

レースの境界が[なし]でない場合、接触角よりも大きい勾配は強い勾配と見なされ、その角度よりも小さい勾配は弱い勾配と見なされます。

4.9.4.18 パスごとの粗いオフセットと深さ

Image-to-gcode は、オプションでルーティングパスを実行できます。 連続する荒削りパスの深さは、パスごとの荒削り深さによって与えられます。 たとえば、0.2 を入力すると、深さ 0.2 の最初の荒削りパス、深さ 0.4 の 2 番目の荒削りパスが実行され、画像の完全な深さに達するまで続きます。 荒削りパスのどの部分も、荒削りオフセットよりも最後の部分に近づくことはありません。 次の図は、フライス盤で作成されている背の高い垂直フィーチャーを示しています。 この画像では、パスあたりの荒削りの深さは 0.2 インチで、荒削りのオフセットは 0.1 インチです。

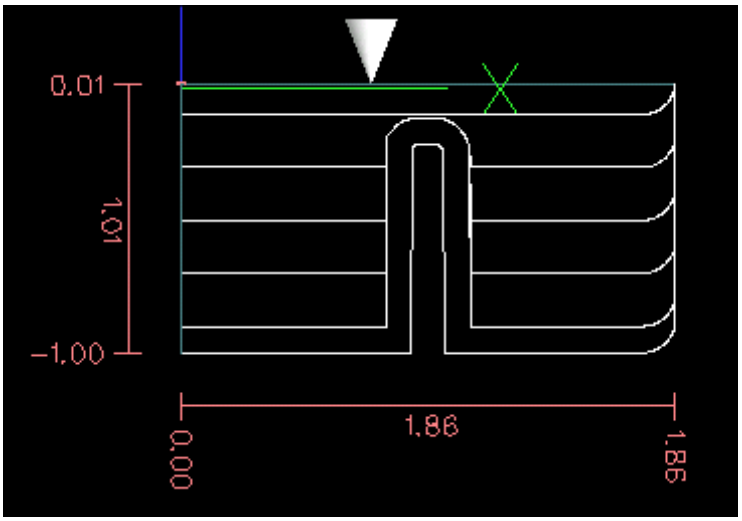


図 5-57

5章 工具補正

5.1 工具補正

5.1.1 工具長オフセット

1.1.1.1 タッチオフ

AXIS インターフェースのタッチオフ画面を使用すると、ツールテーブルを自動的に更新できます。

ツールテーブルを更新するための一般的な手順：

- ホーミング後、Tn M6 を使用して工具をロードします。ここで、n は工具番号です。
- ゲージを使用してツールを確立されたポイントに移動するか、テストカットを行って測定します。
- [手動制御]タブの[タッチオフ]ボタンをクリックします（またはキーボードの[終了]ボタンを押します）。
- [座標系]ドロップダウンボックスで[ツールテーブル]を選択します。
- ゲージまたは測定寸法を入力し、[OK]を選択します。

DRO が正しい Z 位置を表示するように、ツールテーブルが正しい Z 長に変更され、G43 コマンドが発行されて、新しいツール Z 長が有効になります。 ツールテーブルのタッチオフは、ツールに TnM6 がロードされている場合にのみ使用できます。

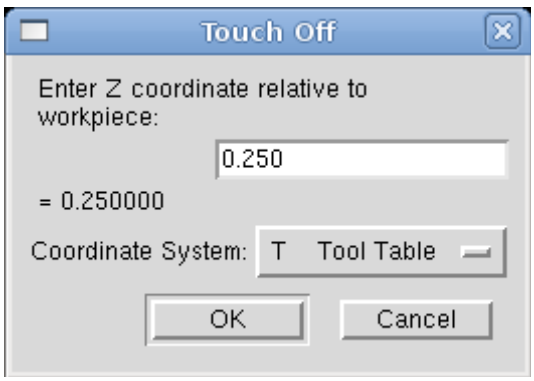


図 6-58

5.1.1.1 G10 L1 / L10 / L11 を使用する

G10 L1 / L10 / L11 コマンドを使用して、工具テーブルのオフセットを設定できます。

- G10 L1Pn-オフセットを値に設定します。 現在の位置は関係ありません。（詳細は G10 L1 を参照）
- G10 L10 Pn-フィクスチャ 1〜8 の現在の位置が値になるようにオフセットを設定します。（詳細は G10 L10 を参照）

- G10 L11Pn-フィクスチャ 9 の現在の位置が値になるようにオフセットを設定します。（詳細は G10 L11 を参照）

5.1.2 ツールテーブル

ツールテーブルは、各ツールに関する情報を含むテキストファイルです。 このファイルは、構成と同じディレクトリーにあり、tool.tbl と呼ばれます。 ツールはツールチェンジャーにあるか、手動で変更されている可能性があります。 ファイルはテキストエディタで編集するか、G10L1 を使用して更新できます。 旋盤工具テーブル形式の例については、旋盤工具テーブルのセクションを参照してください。 ツールテーブルのエントリの最大数は 1000 です。ツールとポケットの最大数は 99999 です。

ツールエディタまたはテキストエディタを使用して、ツールテーブルを編集できます。 テキストエディタを使用する場合は、GUI でツールテーブルをリロードしてください。

1.1.1.1 ツールテーブル形式

表 6.1：ツールテーブルの形式

T#	P#	X	Y	Z	A	B	C	U	V	W	Dia	FA	BA	Ori	Rem
;	（セミコロンを開いた後のデータはありません）														
T1	P17	X0	Y0	Z0	A0	B0	C0	U0	V0	W0	D0	I0	J0	Q0	;rem
T2	P5	X0	Y0	Z0	A0	B0	C0	U0	V0	W0	D0	I0	J0	Q0	;rem
T3	P12	X0	Y0	Z0	A0	B0	C0	U0	V0	W0	D0	I0	J0	Q0	;rem

一般に、新しいツールテーブルの行形式は次のとおりです。

- ; -セミコロンを開く、データなし
- T-工具番号、0-99999（工具番号は一意である必要があります）
- P-ポケット番号、1-99999（ポケット番号は一意である必要があります）
- X..W-指定された軸上の工具オフセット-浮動小数点
- D-工具径-浮動小数点、絶対値
- I-フロントアングル（旋盤のみ）-浮動小数点
- J-バックアングル（旋盤のみ）-浮動小数点
- Q-工具の向き（旋盤のみ）-整数、0-9
- ; -コメントまたはコメントの始まり-テキスト

このファイルは、最初の行の 1 つの開始セミコロンと、それに続く最大 1000 のツールエントリで構成されます。

Note

99999 までの工具番号が許可されていますが、現時点では、技術的な理由から、工具テーブルのエントリ数は最大 1000 個に制限されています。 LinuxCNC 開発者は、最終的にその制限を取り除くことを計画しています。 非常に大きなツールチェンジャーをお持ちの場合は、しばらくお待ちください。

LinuxCNC の以前のバージョンには、ミルと旋盤用に 2 つの異なるツールテーブル形式がありましたが、2.4.x リリース以降、すべてのマシンで 1 つのツールテーブル形式が使用されています。 マシンに関係のない、または使用する必要のないツールテーブルの部分は無視してください。

セミコロンを開いた後のツールテーブルファイルの各行には、1 つのツールのデータが含まれています。 1 行には最大 16 のエントリが含まれる場合がありますが、含まれるエントリははるかに少なくなる可能性があります。

長さ、直径などに使用される単位は機械単位です。

特にランダム化ツールチェンジャーを使用する場合は、ツールエントリを昇順で保持することをお勧めします。 ツールテーブルでは、任意の順序でツール番号を使用できますが。

各行には最大 16 のエントリを含めることができます。 最初の 2 つのエントリは必須です。 最後のエントリ（セミコロンが前に付いたコメントまたはコメント）はオプションです。 表に示すように、エントリが列に配置されていると読みやすくなりますが、唯一の形式要件は、行の各エントリの後に少なくとも 1 つのスペースまたはタブがあり、各エントリの最後に改行文字があることです。 エントリ。

エントリの意味とそれぞれに入力するデータの種類は次のとおりです。

ツール番号（必須）T 列には、ツールのコード番号を表す番号（符号なし整数）が含まれています。 コードが符号なし整数である限り、ユーザーは任意のツールに任意のコードを使用できます。

ポケット番号（必須）P 列には、ツールを見つけることができるツールチェンジャースロットのポケット番号（スロット番号）を表す番号（符号なし整数）が含まれます。 この列のエントリはすべて異なっている必要があります。

ポケット番号は通常 1 から始まり、ツールチェンジャーで利用可能な最も高いポケットまで上がります。 ただし、すべてのツールチェンジャーがこのパターンに従うわけではありません。 ポケット番号は、ツールチェンジャーがポケットを参照するために使用する番号によって決まります。 つまり、使用するポケット番号はツールチェンジャーで使用される番号付けスキームによって決定され、使用するポケット番号はマシン上で意味をなさなければならないということです。

データオフセット番号（オプション）データオフセット列（XYZABCUVW）には、各軸の工具オフセットを表す実数が含まれています。 この番号は、工具長オフセットが使用されていて、この工具が選択されている場合に使用されます。 これらの数値は、正、ゼロ、または負の場合があり、実際には完全にオプションです。 ここに少なくとも 1 つのエントリを作成することをお勧めしますが、そうでない場合は、最初にツールテーブルにエントリを作成してもほとんど意味がありません。

一般的なミルでは、おそらく Z（工具長オフセット）のエントリが必要です。 一般的な旋盤では、X（X 工具オフセット）と Z（Z 工具オフセット）のエントリが必要になる場合があります。 カッター直径補正（カッターコ

ンプ）を使用する一般的なミルでは、おそらく D（カッター直径）のエントリも追加する必要があります。 ツールノーズ径補正（ツールコンプ）を使用する一般的な旋盤では、おそらく D（ツールノーズ径）のエントリも追加する必要があります。

旋盤には、工具の形状と方向を説明するための追加情報も必要です。 したがって、I（ツールの前面角度）と J（ツールの背面角度）のエントリが必要になる場合があります。 おそらく、Q（ツールの向き）のエントリも必要です。

詳細については、旋盤ユーザー情報の章を参照してください。

[直径]列には実数が含まれています。 この数値は、このツールを使用してカッター補正がオンになっている場合にのみ使用されます。 補正中にプログラムされたパスが切削される材料のエッジである場合、これは工具の測定された直径を表す正の実数である必要があります。 補正中にプログラムされたパスが、直径が公称のツールのパスである場合、これは、ツールの測定された直径と公称直径の差のみを表す小さな数値（正または負、ただしゼロに近い）である必要があります。 工具でカッター補正を使用しない場合は、この列の番号は関係ありません。

コメント列は、オプションでツールを説明するために使用できます。 どんな種類の説明でも OK です。 このコラムは、人間の読者のみを対象としています。 コメントの前にはセミコロンを付ける必要があります。

5.1.2.1 ツールチェンジャー

LinuxCNC は、手動、ランダムロケーション、固定ロケーションの 3 種類のツールチェンジャーをサポートしています。 LinuxCNC ツールチェンジャーの構成に関する情報は、INI の章の EMCIO セクションにあります。

手動ツールチェンジャー手動ツールチェンジャー（手動でツールを変更する）は、固定位置ツールチェンジャーのように扱われ、P 番号は無視されます。 手動ツールチェンジャーの使用は、変更時にツールと一緒に残るツールホルダー（Cat、NMTB、Kwik Switch など）がある場合にのみ意味があり、したがって、スピンドルに対するツールの位置が保持されます。 R-8 またはルーターコレットタイプのツールホルダーを備えたマシンは、ツールの位置を保持しないため、手動のツールチェンジャーは使用しないでください。

固定位置ツールチェンジャー固定位置ツールチェンジャーは、常にツールをツールチェンジャーの固定位置に戻します。 これには、旋盤タレットなどの設計も含まれます。 LinuxCNC が固定ロケーションツールチェンジャー用に構成されている場合、P 番号は LinuxCNC によって無視されます（ただし、読み取り、保持、および書き換え）ので、任意の簿記番号に P を使用できます。

ランダムロケーションツールチェンジャーランダムロケーションツールチェンジャーは、スピンドルのツールをチェンジャーのツールと交換します。 このタイプのツールチェンジャーを使用すると、ツール交換後、ツールは常に別のポケットに入れられます。 ツールが変更されると、LinuxCNC はポケット番号を書き換えて、ツールがどこにあるかを追跡します。 T は任意の数にすることができますが、P はマシンにとって意味のある数でなければなりません。

5.1.3 カッター補正

カッター補正により、プログラマは正確な工具径を知らなくても工具パスをプログラムできます。 唯一の注意点は、プログラマーは、少なくとも使用される可能性のある最大の工具半径と同じ長さになるように、移動中のリードをプログラムする必要があることです。

カッターの後ろからカッターの動きの方向を向いているときに、カッター補正がラインの左側または右側にあるときに、カッターがたどることができる 2 つの可能なパスがあります。 これを視覚化するために、ツールがパーツを横切って進むときに、ツールの後ろを歩いているパーツの上に立っていると想像してください。 G41 は線の左側で、G42 は線の右側です。

各移動の終点は、次の移動によって異なります。 次の移動で外側のコーナーが作成された場合、移動は補正されたカットラインの終点になります。 次の動きが内側のコーナーで作成された場合、パーツを削らないように、動きは短く停止します。 次の図は、補正された移動が次の移動に応じてさまざまなポイントでどのように停止するかを示しています。

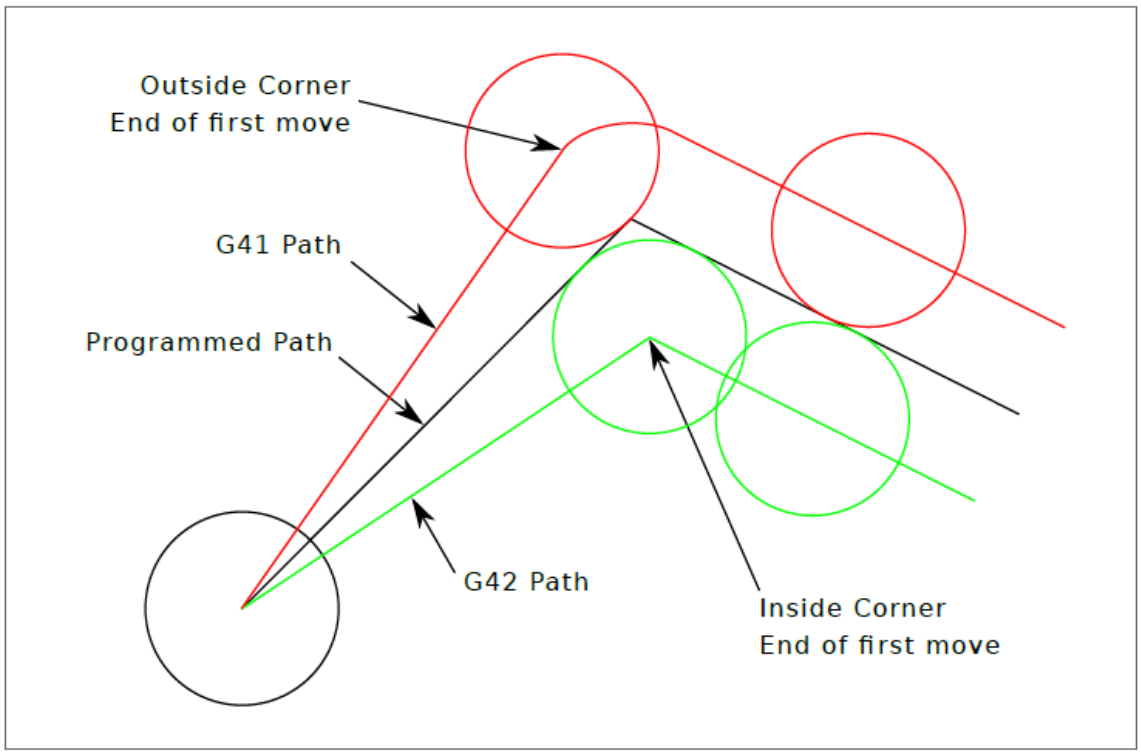


図 6-59

1.1.1.1 概要

工具テーブルカッター補正は、工具テーブルのデータを使用して、必要なオフセットを決定します。 データは、G10L1 を使用して実行時に設定できます。

エントリ移動のプログラミング補正を実行するのに十分な長さの移動は、エントリ移動として機能します。 最小の長さはカッターの半径です。 これは、ワークピースの上をすばやく移動する可能性があります。 G41 / 42 の後に複数の高速移動が発行された場合、最後の 1 つだけがツールを補正位置に移動します。

次の図では、エントリの移動が行の右側で補正されていることがわかります。 この場合、これによりツールの中心が X0 の右側に配置されます。 プロファイルをプログラムする場合、終了が X0 にある場合、結果のプロファイルは、エントリの移動のオフセットのためにバンプを残します。

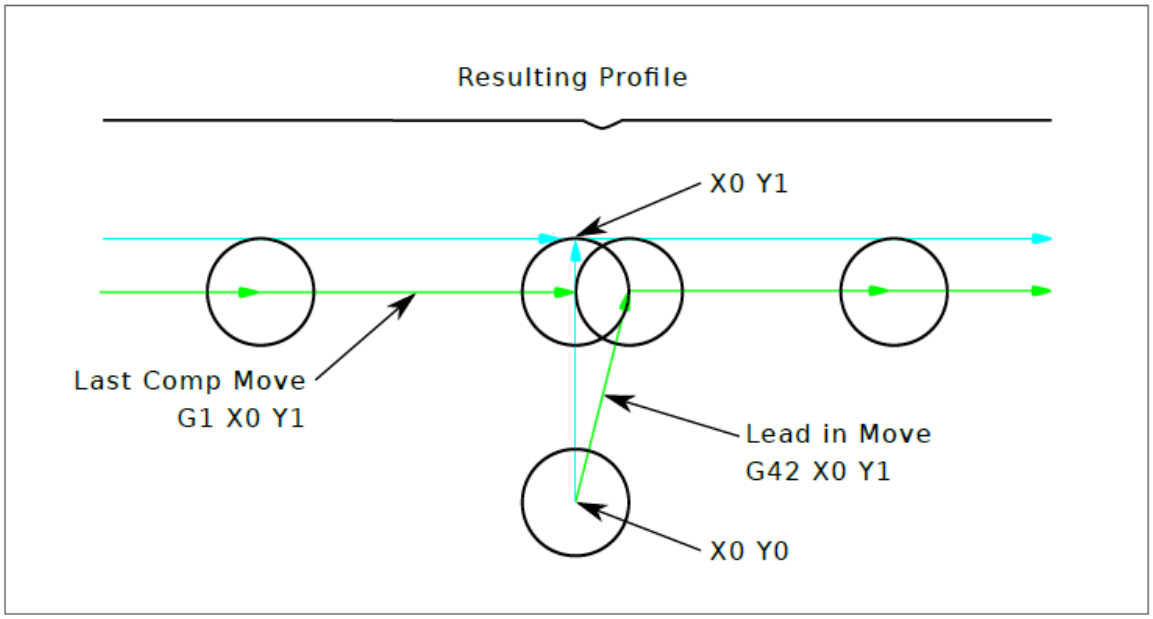


図 6-60

Z モーション Z 軸モーションは、XY 平面で輪郭を追跡しているときに発生する場合があります。 輪郭の一部は、パーツの上の Z 軸を後退させ、次の開始点で Z 軸を延長することによってスキップできます。

早送り早送りは、補正がオンになっているときにプログラムできます。

良い習慣

- G40 でプログラムを開始して、補正がオフになっていることを確認します。

5.1.3.1 例

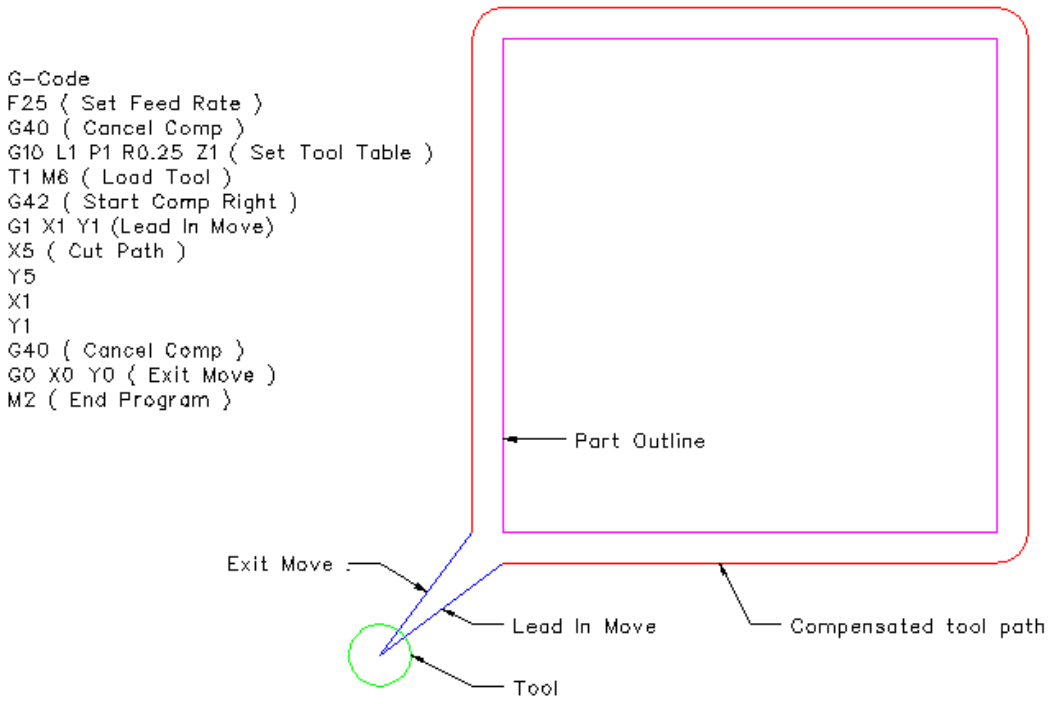


図 6-61

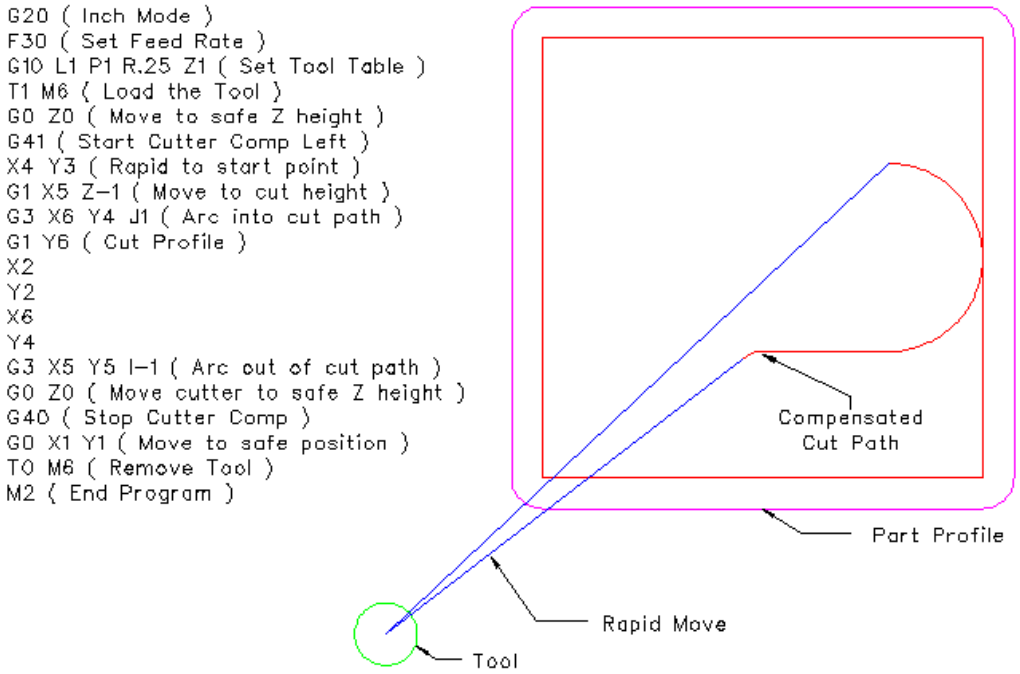
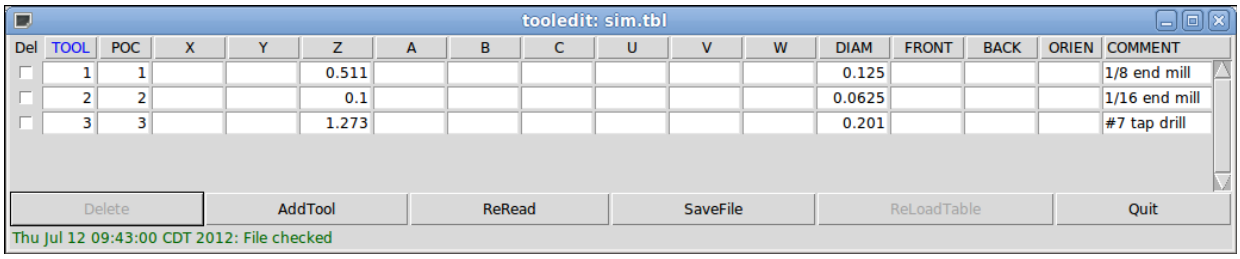


図 6-62

5.2 ツール編集 GUI

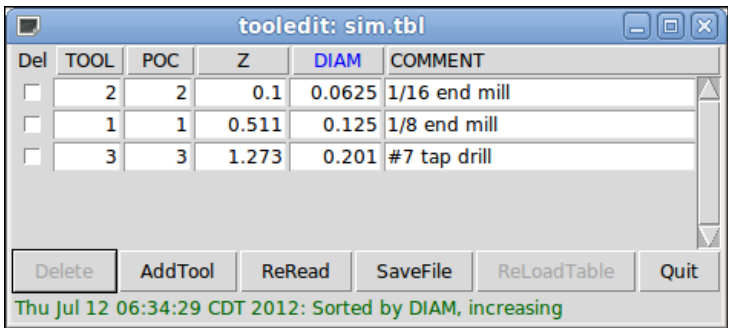
5.2.1 概要



ツール編集プログラムは、[ファイルの保存]ボタンを使用して、編集した変更でツールテーブルファイルを更新できます。 [ファイルの保存]ボタンはシステムファイルを更新しますが、実行中の LinuxCNC インスタンスで使用されるツールテーブルデータを更新するには、別のアクションが必要です。 軸 GUI を使用すると、LinuxCNC で使用されるファイルと現在のツールテーブルデータの両方を[再ロードテーブル]ボタンで更新できます。 このボタンは、マシンがオンでアイドル状態の場合にのみ有効になります。

5.2.2 列の並べ替え

ツールテーブルの表示は、列ヘッダーをクリックすることで、任意の列で昇順で並べ替えることができます。 2 回目のクリックは、降順で並べ替えられます。 列の並べ替えには、マシンがデフォルトの tcl バージョン> = 8.5 で構成されている必要があります。

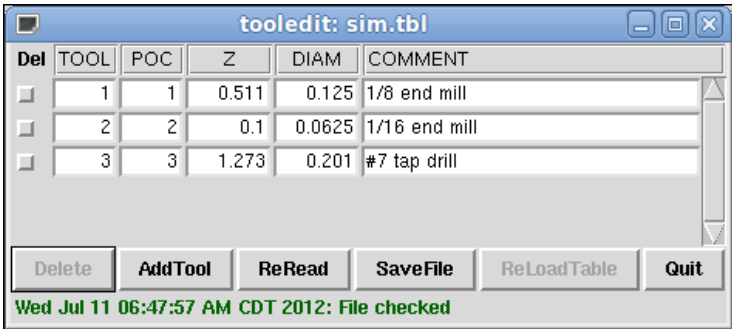


システムにインストールされている他のアプリケーションによっては、次のコマンドで tcl /tk8.5 を有効にする必要がある場合があります。

```
sudo update-alternatives --config tclsh ;# select the option for tclsh8.5
sudo update-alternatives --config wish ;# select the option for wish8.5
```

5.2.3 列の選択

デフォルトでは、tooledit プログラムはすべての可能なツールテーブルパラメータ列を表示します。 すべてのパラメータを使用するマシンはほとんどないため、表示される列は次の ini ファイル設定で制限できます。



INI ファイルの構文

```
[DISPLAY]
TOOL_EDITOR = tooledit column_name column_name ...
```

Z 列と DIAM 列の例

```
[DISPLAY]
TOOL_EDITOR = tooledit Z DIAM
```

5.2.4 スタンドアロン使用

ツール編集プログラムは、スタンドアロンプログラムとして呼び出すこともできます。 たとえば、プログラムがユーザー PATH にある場合、tooledit と入力すると、使用構文が表示されます。

スタンドアロン

```
tooledit
Usage:
tooledit filename
tooledit [column_1 ... column_n] filename
```

スタンドアロンのツールエディットを実行中の LinuxCNC アプリケーションと同期するには、ファイル名を LinuxCNCini ファイルで指定されているものと同じ[EMCIO] TOOL_TABLE ファイル名に解決する必要があります。

LinuxCNC の実行中にプログラム tooledit を使用すると、gcode コマンドの実行または他のプログラムがツールテーブルデータおよびツールテーブルファイルを変更する場合があります。 ファイルの変更は tooledit によって検出され、メッセージが表示されます。

```
Warning: File changed by another process
```

ツール編集ツールテーブルの表示を更新して、[再読み取り]ボタンで変更したファイルを読み取ることができます。

ツールテーブルは、ini ファイルで次のエントリで指定されます。

```
[EMCIO]TOOL_TABLE = tool_table_filename
```

ツールテーブルファイルは、（ワードプロセッサではなく）任意の単純なテキストエディタで編集できます。

Axis GUI は、オプションで ini ファイル設定を使用してツールエディタプログラムを指定できます。

```
[DISPLAY]TOOL_EDITOR = path_to_editor_program
```

デフォルトでは、tooledit という名前のプログラムが使用されます。 このエディタは、すべてのツールテーブルパラメータをサポートし、ツールエントリの追加と削除を可能にし、パラメータ値に対していくつかの有効性チェックを実行します。

6章 一般的な情報

6.1 インテグレーターの概念

6.1.1 ファイルの場所

LINUXCNC は、特定の場所で構成ファイルと G コードファイルを探します。 場所は、LINUXCNC の実行方法によって異なります。

1.1.1.1 インストール済み

LIVECD から実行中の LINUXCNC を実行している場合、または DEB を介してインストールし、メニューから構成ピッカー LINUXCNC を使用する場合、LINUXCNC は次のディレクトリを検索します。

- LINUXCNC ディレクトリは/ *HOME* / *USER-NAME* / *LINUXCNC* にあります。
- 構成ディレクトリは/ *HOME* / *USER-NAME* / *LINUXCNC* / *CONFIGS* にあります。
  - CONFIGURATION FILES ARE LOCATED AT */HOME/USER-NAME/LINUXCNC/CONFIGS/NAME-OF-CONFIG*.
- G コードファイルは/ *HOME* / *USER-NAME* / *LINUXCNC* / *NC_FILES* 'にあります。

  たとえば、MILL という構成とユーザー名 FRED の場合、ディレクトリとファイルの構造は次のようになります。

- */HOME/FRED/LINUXCNC*
- */HOME/FRED/LINUXCNC/NC_FILES*
- */HOME/FRED/LINUXCNC/CONFIGS/MILL*
  - */HOME/FRED/LINUXCNC/CONFIGS/MILL/MILL.INI*
  - */HOME/FRED/LINUXCNC/CONFIGS/MILL/MILL.HAL*
  - */HOME/FRED/LINUXCNC/CONFIGS/MILL/MILL.VAR*
  - */HOME/FRED/LINUXCNC/CONFIGS/MILL/TOOL.TBL*

NOTE

一部のファイルのオプションの場所は、INI ファイルで構成できます。 DISPLAY セクションと RS274NGC セクションを参照してくださ

6.1.1.1 コマンドライン

コマンドラインから LINUXCNC を実行し、INI ファイルの名前と場所を指定すると、ファイルの場所を別の場所に置くことができます。 コマンドラインから LINUXCNC を実行するためのオプションを表示するには、LINUXCNC-H を実行します。

6.1.2 ファイル

各構成ディレクトリには、少なくとも次のファイルが必要です。

- INI ファイル.INI
- INI ファイルの HAL セクションで指定された HAL ファイル.HAL または HALTCL ファイル.TCL。

[注]一部の GUI では、他のファイルが必要になる場合があります。

オプションで、次のものも使用できます。

- 変数ファイル.VAR
  - ディレクトリ内の.VAR ファイルを省略し、[RS274NGC] PARAMETER_FILE = SOMEFILENAME.VAR を含めると、LINUXCNC の起動時にファイルが作成されます。
  - .VAR ファイルを省略し、項目[RS274NGC] PARAMETER_FILE を省略した場合、LINUXCNC の起動時に RS274NGC.VAR という名前の VAR ファイルが作成されます。 [RS274NGC] PARAMETER_FILE を省略すると、紛らわしいメッセージが表示される場合があります。
- INI ファイルで[EMCMOT] TOOL_TABLE が指定されている場合は、ツールテーブルファイル.TBL。一部の構成では、ツールテーブルは必要ありません。

6.1.3 ステッパーシステム

6.1.3.1 基準期間

BASE_PERIOD は、LINUXCNC コンピューターのハートビートです。[1)] 周期ごとに、ソフトウェアステップジェネレーターは、次のステップパルスの時間であるかどうかを判断します。 周期を短くすると、制限内で 1 秒あたりのパルス数を増やすことができます。 ただし、短すぎると、コンピューターがステップパルスの生成に多くの時間を費やすため、他のすべての処理が遅くなったり、ロックしたりする可能性があります。 レイテンシーとステッピングドライブの要件は、使用できる最短期間に影響します。

最悪の場合の待ち時間は 1 分間に数回しか発生しない可能性があり、モーターが方向を変えているときに発生する悪い待ち時間の確率は低くなります。 そのため、非常にまれなエラーが発生して、パーツがときどき台無しになり、トラブルシューティングが不可能になる可能性があります。

この問題を回避する最も簡単な方法は、ドライブの最長のタイミング要件とコンピューターの最悪の場合の待ち時間の合計である BASE_PERIOD を選択することです。 これが常に最良の選択であるとは限りません。 たとえば、20 US の方向信号保持時間要件のあるドライブを実行していて、レイテンシテストで最大レイテンシが 11 US であると示された場合、BASE_PERIOD を 20 + 11 = 31US に設定すると -あるモードでは毎秒 32,258 ステップ、別のモードでは毎秒 16,129 ステップです。

問題は、20US のホールドタイム要件にあります。 これに 11US の遅延を加えると、31US の遅い期間を使用せざるを得なくなります。 ただし、LINUXCNC ソフトウェアステップジェネレーターに

は、さまざまな時間を 1 つの期間から複数の期間に増やすことができるパラメーターがいくつかあります。 たとえば、STEPLEN [2]が 1 から 2 に変更された場合、ステップパルスの開始と終了の間に 2 つの期間があります。 同様に、DIRHOLD [3]が 1 から 3 に変更された場合、ステップパルスと方向ピンの変更の間に少なくとも 3 つの期間があります。

DIRHOLD を使用して 20US のホールド時間要件を満たすことができる場合、次に長い時間は 4.5US のハイタイムです。 11US のレイテンシーを 4.5US のハイタイムに追加すると、最小期間は 15.5US になります。 15.5 US を試してみると、コンピューターの動作が遅いことがわかったので、16US に落ち着きました。 DIRHOLD を 1（デフォルト）のままにすると、ステップと方向の間の最小時間は、16US の期間から 11US のレイテンシー= 5 US を引いたものになり、これでは不十分です。 さらに 15 人必要です。 期間は 16US なので、もう 1 つ必要です。 したがって、DIRHOLD を 1 から 2 に変更します。これで、ステップパルスの終了から方向転換ピンまでの最小時間は 5 + 16 = 21 US であり、ドライブが間違った方向にステップすることを心配する必要はありません。 待ち時間。

STEPGEN の詳細については、STEPGEN のセクションを参照してください。

NOTE

1) このセクションでは、LINUXCNC の組み込みステップジェネレータである STEPGEN の使用について説明します。 一部のハードウェアデバイスには独自のステップジェネレーターがあり、LINUXCNC の組み込みのものを使用していません。 その場合は、ハードウェアのマニュアルを参照してください。

2) STEPLEN は、HAL コンポーネントである LINUXCNC の組み込みステップジェネレーターである STEPGEN のパフォーマンスを調整するパラメーターを指します。 このパラメータは、ステップパルス自体の長さを調整します。 読み続けてください、すべては最終的に説明されます。

3) DIRHOLD は、方向保持時間の長さを調整するパラメーターを指します。

6.1.3.2 ステップタイミング

一部のドライブのステップタイミングとステップスペースは異なります。 この場合、ステップポイントが重要になります。 ドライブが立ち下がりエッジを踏む場合は、出力ピンを反転させる必要があります。

6.1.4 サーボシステム

6.1.4.1 基本操作

サーボシステムは、同等のステッパーシステムよりも高速で正確ですが、コストが高く複雑です。 ステッパーシステムとは異なり、サーボシステムにはある種の位置フィードバックデバイスが必要であり、ステッパーシステムのように箱から出してすぐに機能するわけではないため、調整または調整する必要があります。 これらの違いは、一般に開ループで動作するステッピングモーターとは異なり、サーボが閉ループシステムであるために存在します。 閉ループとはどういう意味ですか？ サーボモーターシステムの接続方法の簡略図を見てみましょう。

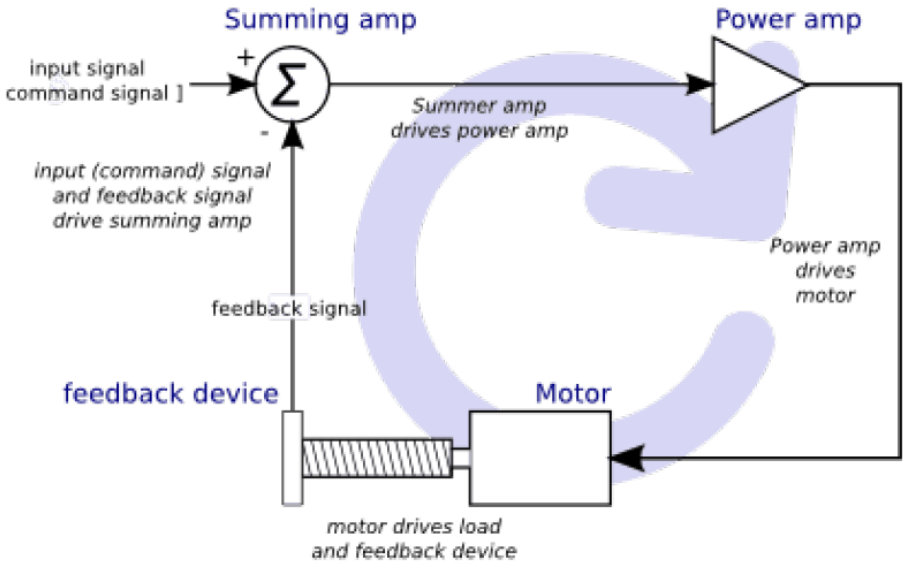


図 7-63

この図は、入力信号（およびフィードバック信号）が加算増幅器を駆動し、加算増幅器が電力増幅器を駆動し、電力増幅器がモーターを駆動し、モーターが負荷（およびフィードバック装置）を駆動し、フィードバック装置（ および入力信号）モーターを駆動します。 これは、A が B を制御し、B が C を制御し、C が D を制御し、D が A を制御する円（閉ループ）に非常によく似ています。

これまでサーボシステムを使用したことがない場合、特に入力がスムーズに出力に進み、二度と戻らない通常の電子回路と比較すると、これは最初は非常に奇妙な考えに思えることは間違いありません。[4] すべてが他のすべてを制御している場合、誰が責任を持って、それがどのように機能するのでしょうか。 答えは、LinuxCNC がこのシステムを制御できるということですが、いくつかの制御方法の 1 つを選択することによってそれを行う必要があります。 LinuxCNC が使用する制御方法は、最も単純で最良の 1 つであり、PID と呼ばれます。

PID は、比例、積分、および微分の略です。 比例値は現在のエラーに対する反応を決定し、積分値は最近のエラーの合計に基づいて反応を決定し、微分値はエラーが変化している速度に基づいて反応を決定します。 これらは、仕掛品を設定値に従わせるタスクに適用される 3 つの一般的な数学的手法です。 LinuxCNC の場合、制御したいプロセスは実際の軸位置であり、設定点はコマンドされた軸位置です。

NOTE

4) それが役立つ場合、デジタルの世界でこれに最も近いのはステートマシン、シーケンシャルマシンなどです。ここで、出力が現在実行していることは、入力（および出力）が以前に実行していたことによって異なります。 それが役に立たない場合は、気にしないでください。

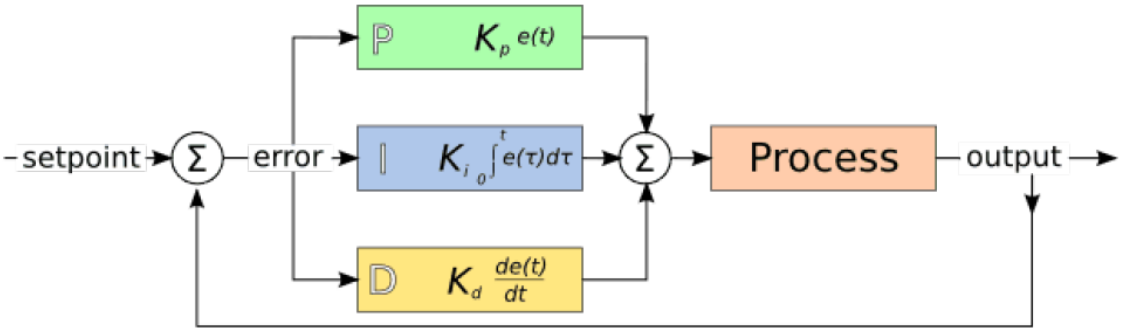


図 7-64

PID コントローラーアルゴリズムの 3 つの定数を調整することにより、コントローラーは特定のプロセス要件に合わせて設計された制御アクションを提供できます。 コントローラーの応答は、エラーに対するコントローラーの応答性、コントローラーが設定値をオーバーシュートする程度、およびシステムの振動の程度の観点から説明できます。

6.1.4.2 比例項

比例項（ゲインと呼ばれることもあります）は、現在のエラー値に比例する出力に変化をもたらします。 比例ゲインが高いと、エラーの特定の変化に対して出力が大きく変化します。 比例ゲインが高すぎると、システムが不安定になる可能性があります。 対照的に、ゲインが小さいと、大きな入力エラーに対する出力応答が小さくなります。 比例ゲインが低すぎると、システム障害に対応する際の制御動作が小さすぎる可能性があります。

外乱がない場合、純粋な比例制御はその目標値に落ち着きませんが、比例ゲインとプロセスゲインの関数である定常状態エラーを保持します。 定常状態のオフセットにもかかわらず、チューニング理論と産業慣行の両方が、出力変化の大部分に寄与するのは比例項であることを示しています。

6.1.4.3 積分項

積分項（リセットと呼ばれることもあります）からの寄与は、エラーの大きさとエラーの持続時間の両方に比例します。 時間の経過に伴う瞬間誤差を合計すると（誤差を積分）、以前に修正されているはずの累積オフセットが得られます。 次に、累積誤差に積分ゲインが乗算され、コントローラー出力に追加されます。

積分項（比例項に追加された場合）は、プロセスの設定点への移動を加速し、比例のみのコントローラーで発生する残留定常状態エラーを排除します。 ただし、積分項は過去からの累積誤差に対応しているため、現在値が設定値をオーバーシュートする可能性があります（設定値を超えて反対方向に偏差が生じる）。

6.1.4.4 微分項

プロセスエラーの変化率は、時間の経過に伴うエラーの傾き（つまり、時間に対するその1次導関数）を決定し、この変化率に導関数ゲインを掛けることによって計算されます。

微分項はコントローラー出力の変化率を遅くし、この効果はコントローラーの設定値の近くで最も顕著になります。 したがって、微分制御を使用して、積分コンポーネントによって生成されるオーバーシュートの大きさを減らし、コントローラーとプロセスの組み合わせの安定性を向上させます。

6.1.4.5 ループチューニング

PIDコントローラーパラメーター（比例、積分、微分項のゲイン）が正しく選択されていない場合、制御されたプロセス入力は不安定になる可能性があります。つまり、振動の有無にかかわらず、出力が発散し、飽和または機械的破損によってのみ制限されます。 制御ループの調整とは、その制御パラメーター（ゲイン/比例帯域、積分ゲイン/リセット、微分ゲイン/レート）を目的の制御応答の最適値に調整することです。

6.1.4.6 手動チューニング

簡単な調整方法は、最初にI値とD値をゼロに設定することです。 ループの出力が発振するまでPを増やします。その後、Pは、1/4振幅減衰タイプの応答の値の約半分に設定する必要があります。 次に、プロセスに十分な時間内にオフセットが正しくなるまでIを増やします。 ただし、多すぎると不安定になります。 最後に、必要に応じて、負荷の乱れの後でループが基準に到達するのに許容できる速さになるまで、Dを増やします。 ただし、Dが多すぎると、過剰な応答とオーバーシュートが発生します。 通常、高速PIDループ調整はわずかにオーバーシュートして、設定値にすばやく到達します。 ただし、一部のシステムはオーバーシュートを受け入れることができません。その場合、オーバーダンプされた閉ループシステムが必要になります。これには、発振を引き起こすP設定の半分よりも大幅に少ないP設定が必要になります。

6.1.5 RTAI

リアルタイムアプリケーションインターフェイス（RTAI）は、最高のリアルタイム（RT）パフォーマンスを提供するために使用されます。 RTAIパッチを適用したカーネルを使用すると、厳密なタイミング制約のあるアプリケーションを作成できます。 RTAIを使用すると、ソフトウェアのステップ生成など、正確なタイミングを必要とする機能を利用できます。

6.1.5.1 ACPI

ADVANCED CONFIGURATION AND POWER INTERFACE（ACPI）にはさまざまな機能があり、そのほとんどがRTパフォーマンスに干渉します（たとえば、電源管理、CPUパワーダウン、CPU周波数スケーリングなど）。 LINUXCNCカーネル（およびおそらくすべてのRTAIパッチが適用されたカー

ネル）では、ACPI が無効になっています。 ACPI は、シャットダウンの開始後にシステムの電源を切ることも行います。そのため、コンピューターの電源を完全に切るには、電源ボタンを押す必要がある場合があります。 RTAI グループは最近のリリースでこれを改善しているので、結局 LINUXCNC システムが自動的に停止する可能性があります。

6.1.6

6.2 レイテンシーテスト

このテストは、CNC マシンを駆動できるかどうかを確認するために PC で実行する必要がある最初のテストです。

レイテンシーとは、PC が実行中の処理を停止し、外部要求に応答するのにかかる時間です。 LINUXCNC の場合、要求は BASE_THREAD であり、ステップパルスのタイミング基準として機能する周期的なハートビートを作成します。 レイテンシーが低いほど、ハートビートをより速く実行でき、ステップパルスがより速くよりスムーズになります。

レイテンシーは CPU 速度よりもはるかに重要です。 毎回 10 マイクロ秒以内の割り込みに応答する低 PENTIUMII は、最新かつ最速の P4 ハイパースレッディングビーストよりも優れた結果をもたらす可能性があります。

レイテンシを決定する要因は CPU だけではありません。 マザーボード、ビデオカード、USB ポート、およびその他の多くのものが遅延を損なう可能性があります。 何を扱っているかを知る最良の方法は、RTAI レイテンシーテストを実行することです。

ソフトウェアでステップパルスを生成することには、非常に大きな利点が 1 つあります。それは無料です。 ほぼすべての PC には、ソフトウェアによって生成されたステップパルスを出力できるパラレルポートがあります。 ただし、ソフトウェアのステップパルスにはいくつかの欠点もあります。

- 制限された最大ステップレート
- 生成されたパルスのジッタ
- CPU をロードします

PC が LINUXCNC をどれだけうまく実行できるかを知る最良の方法は、HAL 遅延テストを実行することです。 テストを実行するには、ターミナルウィンドウを開き（UBUNTU では、アプリケーション→アクセサリ→ターミナルから）、次のコマンドを実行します。

```
latency-test
```

次のようなものが表示されます。

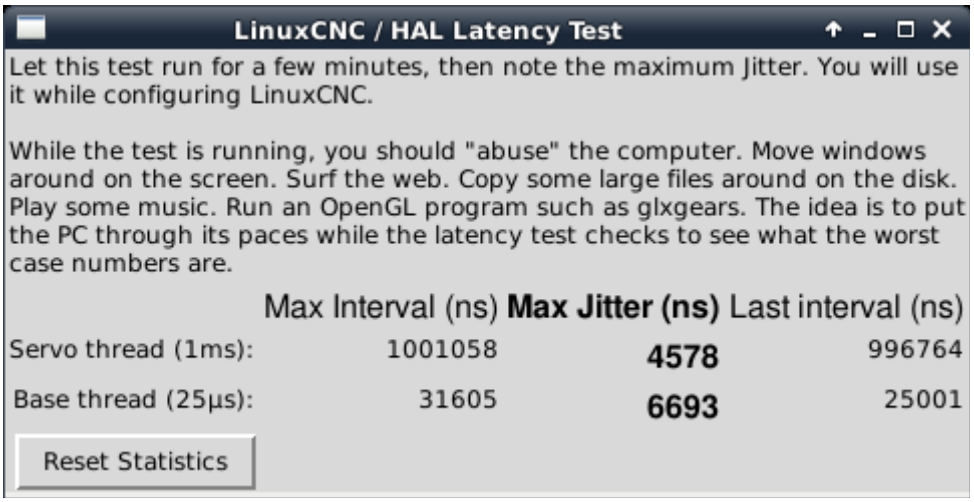


図 7-65

テストの実行中は、コンピューターを悪用する必要があります。 画面上でウィンドウを移動します。 ウェブをサーフィンします。 いくつかの大きなファイルをディスクにコピーします。 音楽を再生します。 GLXGEARS などの OPENGL プログラムを実行します。 アイデアは、レイテンシーテストが最悪のケースの数が何であるかを確認する間、PC をそのペースに乗せることです。

NOTE

遅延テストの実行中は、LINUXCNC または STEPCONF を実行しないでください。

重要な数値は最大ジッタです。 上記の例では、これは 9075 ナノ秒、つまり 9.075 マイクロ秒です。 この番号を記録し、要求されたときに STEPCONF に入力します。

上記の例では、レイテンシテストは数秒間しか実行されませんでした。 テストは少なくとも数分間実行する必要があります。 最悪の場合の遅延はあまり頻繁に発生しない場合や、特定のアクションを実行した場合にのみ発生する場合があります。 たとえば、1 つの INTEL マザーボードはほとんどの場合かなりうまく機能しましたが、64 秒ごとに 300US の非常に悪い遅延がありました。 幸い、それは修正可能でした。 HTTP：//WIKI.LINUXCNC.ORG/CGI-BIN/WIKI.PL？FIXINGSMIISSUES を参照してください。

では、結果はどういう意味ですか？ 最大ジッター数が約 15～20 マイクロ秒（15000～20000 ナノ秒）未満の場合、コンピューターはソフトウェアステッピングで非常に優れた結果をもたらすはずです。 最大遅延が 30～50 マイクロ秒のような場合でも、良好な結果を得ることができますが、特にマイクロステッピングを使用している場合や非常に細いピッチの親ねじがある場合は、最大ステップレートが少し期待外れになる可能性があります。 数値が 100US 以上（100,000 ナノ秒）の場合、PC はソフトウェアステッピングの候補として適していません。 1 ミリ秒（1,000,000 ナノ秒）を超える数値は、ソフトウェアステッピングを使用するかどうかに関係なく、PC が LINUXCNC の適切な候補ではないことを意味します。

あなたが高い数を得るならば、それらを改善する方法があるかもしれないことに注意してください。 別の PC は、オンボードビデオを使用するときに、非常に悪い遅延（数ミリ秒）を持っていました。 しかし、5 ドルの中古ビデオカードで問題は解決しました。

NOTE

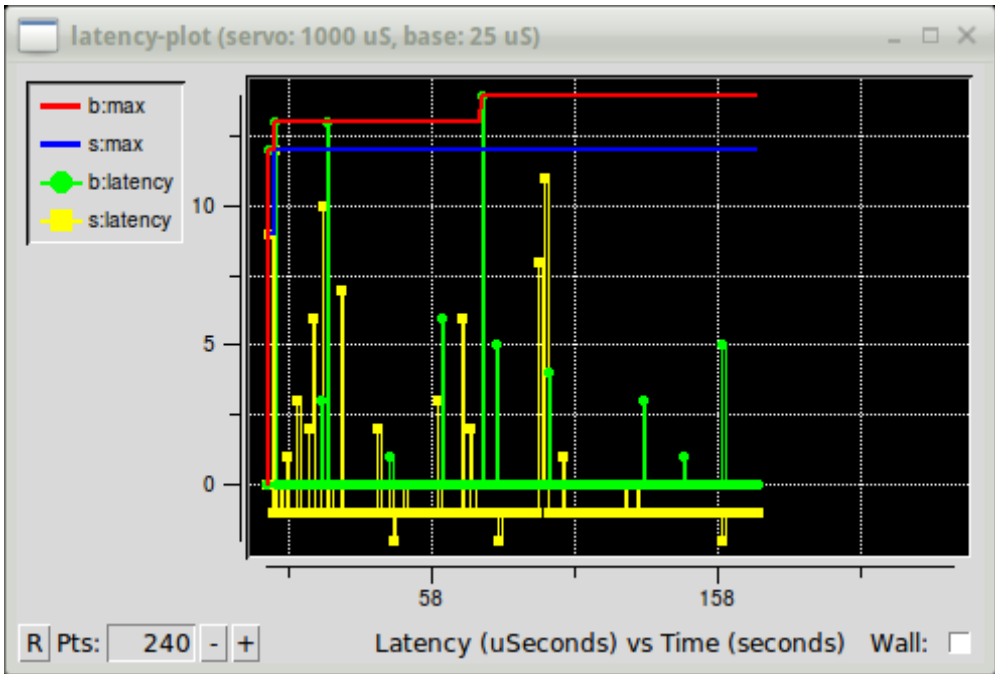
LINUXCNC は、最先端のハードウェアを必要としません。

ステッパーチューニングの詳細については、ステッパーチューニングの章を参照してください。

LINUXCNC が実行されていないときの遅延を調べるために、追加のコマンドラインツールを使用できます。

レイテンシープロットは、ベーススレッドとサーボスレッドのストリップチャート記録を作成します。 他のアプリケーションを起動または使用したときに、レイテンシーの急上昇を確認すると便利な場合があります。

使用法：



レイテンシーヒストグラムには、ベーススレッドとサーボスレッドのレイテンシー（ジッター）のヒストグラムが表示されます。

Usage:

latency-histogram --help | -?

or

latency-histogram [Options]

Options:

--base nS (base thread interval, default: 25000, min: 5000)

--servo nS (servo thread interval, default: 1000000, min: 25000)

--bbinsize nS (base bin size, default: 100

--sbinsize nS (servo bin size, default: 100

--bbins n (base bins, default: 200

--sbins n (servo bins, default: 200

--logscale 0|1 (y axis log scale, default: 1)

--text note (additional note, default: "")

--show (show count of undisplayed bins)

--nobase (servo thread only)

--verbose (progress and debug)

--nox (no gui, display elapsed,min,max,sdev for each thread)

Notes:

Linuxcnc and Hal should not be running, stop with halrun -U.

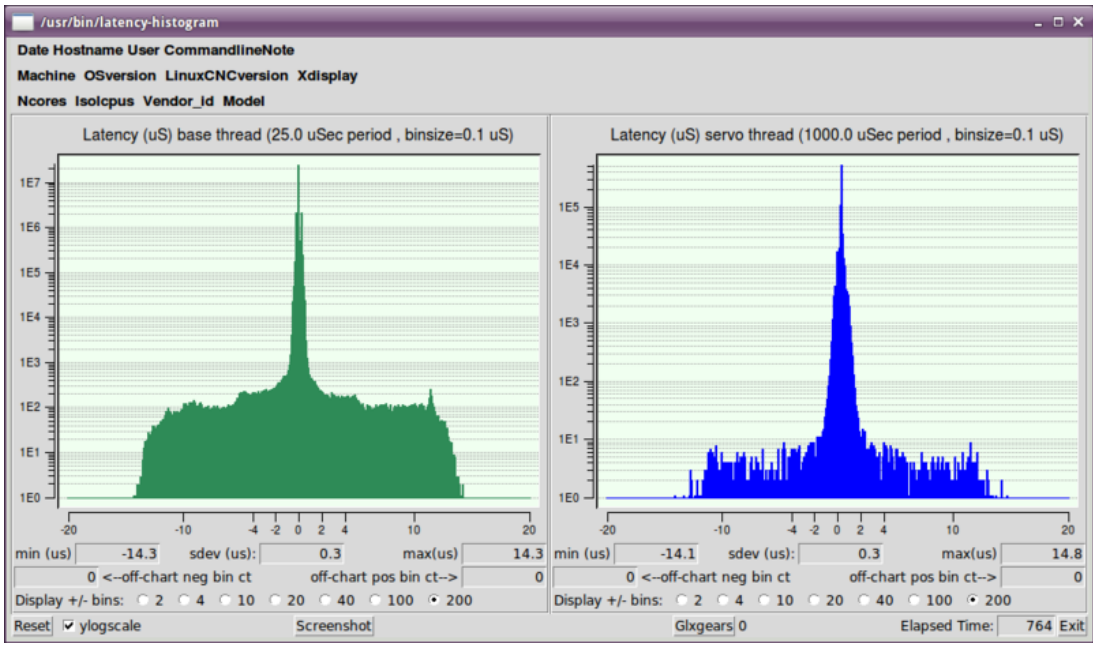
Large number of bins and/or small binsizes will slow updates.

For single thread, specify --nobase (and options for servo thread).

Measured latencies outside of the +/- bin range are reported

with special end bars. Use --show to show count for

the off-chart [pos|neg] bin



6.3 ステッパーチューニング

6.3.1 ソフトウェアステッピングを最大限に活用する

ソフトウェアでステップパルスを生成することには、非常に大きな利点が 1 つあります。それは無料です。 ほぼすべての PC には、ソフトウェアによって生成されたステップパルスを出力できるパラレルポートがあります。 ただし、ソフトウェアのステップパルスにはいくつかの欠点もあります。

- 制限された最大ステップレート
- 生成されたパルスのジッタ
- CPU をロードします

この章には、ソフトウェアで生成された手順から最良の結果を得るのに役立ついくつかの手順があります。

1.1.1.1 レイテンシーテストを実行する

レイテンシーテストの章の説明に従って、レイテンシーテストを実行します。

テストの実行中は、コンピューターを悪用する必要があります。 画面上でウィンドウを移動します。 ウェブをサーフィンします。 いくつかの大きなファイルをディスクにコピーします。 音楽を再生します。 GLXGEARS などの OPENGL プログラムを実行します。 アイデアは、レイテンシーテストが最悪のケースの数が何であるかを確認する間、PC をそのペースに乗せることです。

MAXJITTER というラベルの付いた列の最後の番号が最も重要です。 それを書き留めてください-後で必要になります。 これには、テストの実行全体での最悪の遅延測定値が含まれています。 上記の例では、これは 6693 ナノ秒、つまり 6,69 マイクロ秒であり、優れています。 ただし、この例は数秒間しか実行されませんでした（毎秒 1 行を出力します）。 テストは少なくとも数分間実行する必要があります。 最悪の場合の遅延はあまり頻繁に発生しない場合や、特定のアクションを実行した場合にのみ発生する場合があります。 私はほとんどの場合かなりうまく機能する INTEL マザーボードを 1 つ持っていましたが、64 秒ごとに 300US の非常に悪いレイテンシーがありました。 幸い、これは修正可能です。LINUXCNCWIKI の SMI の問題の修正を参照してください。

では、結果はどういう意味ですか？ 最大ジッター数が約 15〜20 マイクロ秒（15000〜20000 ナノ秒）未満の場合、コンピューターはソフトウェアステッピングで非常に優れた結果をもたらすはずです。 最大遅延が 30〜50 マイクロ秒のような場合でも、良好な結果を得ることができますが、特にマイクロステッピングを使用している場合や非常に細いピッチの親ねじがある場合は、最大ステップレートが少し期待外れになる可能性があります。 数値が 100US 以上（100,000 ナノ秒）の場合、PC はソフトウェアステッピングの候補として適していません。 1 ミリ秒（1,000,000 ナノ秒）を超える数値は、ソフトウェアステッピングを使用するかどうかに関係なく、PC が LINUXCNC の適切な候補ではないことを意味します。

あなたが高い数を得るならば、それらを改善する方法があるかもしれないことに注意してください。 たとえば、オンボードビデオを使用する場合、1 台の PC の遅延（数ミリ秒）が非常に悪かった。 しかし、5 ドルの中古ビデオカードで問題は解決しました。LINUXCNC は最先端のハードウェアを必要としません。

6.3.1.1 ドライブが何を期待しているかを把握する

ステッピングドライブのブランドが異なれば、ステップと方向の入力に異なるタイミング要件があります。 そのため、ドライブの仕様が記載されているデータシートを掘り下げる（または GOOGLE で検索する）必要があります。

GECKO G202 マニュアルから：

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)

GECKO G203V マニュアルから：

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 (0.2 us) before step pulse rising edge

200 ns (0.2 us) hold after step pulse rising edge

XYLOTEX データシートから：

Minimum DIR setup time before rising edge of STEP Pulse 200 ns Minimum

DIR hold time after rising edge of STEP pulse 200 ns

Minimum STEP pulse high time 2.0 us

Minimum STEP pulse low time 1.0 us

Step happens on rising edge

番号を見つけたら、それらも書き留めます。次のステップで必要になります。

6.3.1.2 BASE_PERIOD を選択してください

BASE_PERIOD は、LINUXCNC コンピューターのハートビートです。 周期ごとに、ソフトウェアステップジェネレータは、次のステップパルスの時間であるかどうかを判断します。 周期を短くすると、制限内で 1 秒あたりのパルス数を増やすことができます。 ただし、短すぎると、コンピューターがステップパルスの生成に多くの時間を費やすため、他のすべての処理が遅くなったり、ロックしたりする可能性があります。 レイテンシーとステッパードライブの要件は、1 分でわかるように、使用できる最短期間に影響します。

最初に GECKO の例を見てみましょう。 G202 は、0.5 US でロー、4.5 US でハイになるステップパルスを処理できます。方向ピンは、立ち下がりエッジの 1 US 前に安定し、立ち下がりエッジの後で 20US 安定している必要があります。 最長のタイミング要件は、20US のホールドタイムです。 簡単なアプローチは、期間を 20US に設定することです。 これは、STEP および DIR 行のすべての変更が 20US 離れていることを意味します。 すべてが良いですよね？

違う！ 待ち時間がゼロの場合、すべてのエッジが 20 US 離れており、すべてが正常です。 ただし、すべてのコンピューターにはある程度の待ち時間があります。 レイテンシーは遅延を意味します。 コンピューターのレイテンシーが 11US の場合、ソフトウェアが想定よりも 11US も遅く実行されることがあります。 ソフトウェアの 1 回の実行が 11US 遅れ、次の実行が時間どおりである場合、最初の実行から 2 番目の実行までの遅延はわずか 9US です。 最初のものがステップパルスを生成し、2 番目のものが方向ビットを変更した場合、20 USG202 ホールドタイム要件に違反しただけです。 これは、ドライブが間違った方向に進んだ可能性があり、パーツのサイズが間違っていることを意味します。

この問題の本当に厄介な部分は、それが非常にまれである可能性があるということです。 最悪の場合の待ち時間は 1 分間に数回しか発生しない可能性があり、モーターが方向を変えているときに発生する悪い待ち時間の確率は低くなります。 そのため、非常にまれなエラーが発生し、パーツがときどき破損し、トラブルシューティングが不可能になります。

この問題を回避する最も簡単な方法は、ドライブの最長のタイミング要件とコンピューターの最悪の場合の待ち時間の合計である BASE_PERIOD を選択することです。 20 US のホールドタイム要件で GECKO を実行していて、レイテンシテストで最大レイテンシが 11 US であることが示された場合、BASE_PERIOD を 20 + 11 = 31 US（INI ファイルで 31000 ナノ秒）に設定すると ）、ドライブのタイミング要件を満たすことが保証されています。

ただし、トレードオフがあります。 ステップパルスを作成するには、少なくとも 2 つの周期が必要です。 1 つはパルスを開始し、もう 1 つはパルスを終了します。 周期は 31US なので、ステップパルスを作成するには 2X31 = 62US かかります。 つまり、最大ステップレートは 1 秒あたりわずか 16,129 ステップです。 あまり良くない。（ただし、まだあきらめないでください。次のセクションでは、まだ調整が必要です。）

XYLOTEX の場合、セットアップ時間とホールド時間は非常に短く、それぞれ 200 NS（0.2 US）です。 最長時間は 2US ハイタイムです。 11 US のレイテンシーがある場合は、BASE_PERIOD を 11 + 2 = 13US まで低く設定できます。 長い 20 の私たちの保持時間を取り除くことは本当に役立ちます！ 13 US の期間で、完全なステップは 2X13 = 26 US を要し、最大ステップレートは毎秒 38,461 ステップです！

しかし、まだお祝いを始めることはできません。 13US は非常に短い期間であることに注意してください。 13 US ごとにステップジェネレーターを実行しようとすると、他の何かを実行するのに十分な時間が残っていない可能性があり、コンピューターがロックされます。 25 US 未満の期間を目標としている場合は、25 US 以上から開始し、LINUXCNC を実行して、状況がどのように応答するかを確認する必要があります。 すべてが順調であれば、期間を徐々に減らすことができます。 マウスポインタが遅くなり始め、PC 上の他のすべてが遅くなる場合は、期間が少し短すぎます。 コンピューターをスムーズに実行できるようにした前の値に戻ります。

この場合、25 US から始めて、13 US に到達しようとしましたが、約 16 US が制限であることがわかりました。それ以下であり、コンピューターの応答はあまり良くありません。 だからあなたは 16 私たちを使用します。 16US の周期と 11US のレイテンシーの場合、最短の出力時間は 16-11 = 5US になります。 ドライブに必要なのは 2US だけなので、ある程度の余裕があります。 マージンは良好です。タイミングを近づけすぎてステップを失いたくないのです。

最大ステップレートはどれくらいですか？ 一歩を踏み出すには 2 つの期間があることを忘れないでください。 あなたはその期間に 16US に落ち着いたので、ステップは 32US かかります。 これは、1 秒あたり 31,250 ステップも悪くありません。

6.3.1.3 STEPLEN、STEPSPACE、DIRSETUP、および/または DIRHOLD を使用します

前のセクションでは、XYLOTEX ドライブを 16 US の期間、31,250 ステップ/秒の最大速度にしました。 しかし、GECKO は 31 US でスタックし、1 秒あたり 16,129 ステップというあまり良くありませんでした。 XYLOTEX の例は、私たちが作成できる限り優れています。 しかし、GECKO は改善することができます。

G202 の問題は、20US のホールドタイム要件です。 これに 11US の遅延を加えると、31US の遅い期間を使用せざるを得なくなります。 ただし、LINUXCNC ソフトウェアステップジェネレーターには、さまざまな時間を 1 つの期間から複数の期間に増やすことができるパラメーターがいくつかあります。 たとえば、STEPLEN が 1 から 2 に変更された場合、ステップパルスの開始と終了の間に 2 つの期間があります。 同様に、DIRHOLD が 1 から 3 に変更された場合、ステップパルスと方向ピンの変更の間に少なくとも 3 つの期間があります。

DIRHOLD を使用して 20US のホールド時間要件を満たすことができる場合、次に長い時間は 4.5US のハイタイムです。 11US のレイテンシーを 4.5US のハイタイムに追加すると、最小期間は 15.5US になります。 15.5 US を試してみると、コンピューターの動作が遅いことがわかったので、16US に

落ち着きました。 DIRHOLD を 1（デフォルト）のままにすると、ステップと方向の間の最小時間は、16US の期間から 11US のレイテンシー= 5 US を引いたものになり、これでは不十分です。 さらに 15 人必要です。 期間は 16US なので、もう 1 つ必要です。 したがって、DIRHOLD を 1 から 2 に変更します。ステップパルスの終了から方向変更ピンまでの最小時間は 5 + 16 = 21 US であり、GECKO が間違った方向にステップすることを心配する必要はありません。 待ち時間。

コンピューターのレイテンシーが 11US の場合、16 US の基本期間と、DIRHOLD 値 2 の組み合わせにより、GECKO のタイミング要件を常に満たすことができます。 通常のステッピング（方向の変更なし）の場合、DIRHOLD 値を増やしても効果はありません。 各ステップを実行するのに合計 32US の 2 つの期間が必要であり、XYLOTEX で得たのと同じ 31,250 ステップ/秒のレートがあります。

この例で使用されている 11US のレイテンシー数は非常に優れています。 20 または 25US のように、より大きなレイテンシでこれらの例を実行すると、XYLOTEX と GECKO の両方のトップステップレートが低くなります。 ただし、最適な BASE_PERIOD の計算、および DIRHOLD またはその他のステップジェネレーターパラメーターの調整にも同じ式が適用されます。

6.3.1.4 推測しないでください！

高速で信頼性の高いソフトウェアベースのステッパーシステムの場合、期間やその他の構成パラメーターを推測することはできません。 コンピューターで測定を行い、計算を行って、ドライブが必要な信号を確実に受信できるようにする必要があります。

計算を簡単にするために、OPEN OFFICE スプレッドシートを作成しました。 HTTP：//WIKI.LINUXCNC.ORG/UPLOADS/STEPTIMINGCALCULATOR.ODS レイテンシーテストの結果とステッパードライブのタイミング要件を入力すると、スプレッドシートが最適な BASE_PERIOD を計算します。 次に、期間をテストして、PC の速度が低下したりロックされたりしないことを確認します。 最後に、実際の期間を入力すると、スプレッドシートに、ドライブのタイミング要件を満たすために必要な STEPGEN パラメータ設定が示されます。 また、生成できる最大ステップレートも計算します。

スプレッドシートにいくつかのものを追加して、最高速度とステッパーの電気計算を計算しました。

6.4 ステッパー診断

あなたが得るものがあなたが何度も期待するものではないならば、あなたはちょうどいくらかの経験を得ました。 経験から学ぶことで、全体の理解が深まります。 問題の診断は、分割統治法が最適です。 これは、問題が最も早く見つかるたびに、方程式から変数の 1/2 を削除できるかどうかを意味します。 現実の世界では、これが常に当てはまるとは限りませんが、通常は開始するのに適した場所です。

6.4.1 一般的な問題

1.1.1.1 ステッパーが 1 ステップ移動

ステッピングモーターが動かないという新しい設置の最も一般的な理由は、ステップ信号と方向信号が交換されることです。 ジョグフォワードキーとジョグバックキーを交互に押すと、ステッパーが毎回 1 ステップずつ同じ方向に移動し、手がかりが得られます。

6.4.1.1 ステッパーは動かない

多くのドライブにはイネーブルピンがあるか、出力を有効にするためにチャージポンプが必要です。

6.4.1.2 距離が正しくない

軸に特定の距離を移動するように命令し、その距離を移動しない場合は、スケール設定が間違っています。

6.4.2 エラーメッセージ

1.1.1.1 次のエラー

ステッピングモーターについて話すとき、次のエラーの概念は奇妙です。 これらは開ループシステムであるため、実際に範囲外にあるかどうかを通知する位置フィードバックはありません。 LINUXCNC は、要求されたモーションに対応できるかどうかを計算し、対応できない場合は、次のエラーを返します。 次のエラーは通常、ステッパーシステムで次のいずれかの結果です。

- FERROR が小さすぎます
- MIN_FERROR が小さすぎます
- MAX_VELOCITY が速すぎます
- MAX_ACCELERATION が速すぎます
- BASE_PERIOD の設定が長すぎます
- 軸にバックラッシュが追加されました

上記のいずれかにより、リアルタイムパルスが要求されたステップレートを維持できなくなる可能性があります。 これは、STEPCONF ウィザードにプラグインするのに十分な数を取得するのに十分な時間レイテンシテストを実行しなかった場合、または最大速度または最大加速度を高く設定しすぎた場合に発生する可能性があります。

バックラッシュを追加した場合は、バックラッシュを追加した軸ごとに、STI ファイルの AXIS セクションの MAX_ACCELERATION を最大 2 倍まで STEPGEN_MAXACCEL を増やす必要があります。 LINUXCNC は、反転時に「追加の加速」を使用してバックラッシュを吸収します。 バックラッシュ補正がない場合、ステップジェネレータの加速度はモーションプランナの加速度よりもわずか数パーセント高くなる可能性があります。

6.4.2.1 RTAPI エラー

このエラーが発生した場合：

```
RTAPI: ERROR: Unexpected realtime delay on task n
```

このエラーは、締め切りに間に合わなかったという RTAI からの指示に基づいて RTAPI によって生成されます。 これは通常、INI ファイルの[EMCMOT]セクションの BASE_PERIOD の設定が低すぎることを示しています。 この問題の原因となる遅延があるかどうかを確認するために、遅延テストを長期間実行する必要があります。 STEPCONF WIZARD を使用した場合は、もう一度実行して、基本期間のジッターを再度テストし、[基本的なマシン情報]ページで基本期間の最大ジッターを調整します。 一部のハードウェアが断続的な問題を引き起こすかどうかを確認するために、テストを長時間実行したままにする必要がある場合があります。

LINUXCNC は、リアルタイムスレッドの呼び出し間の CPU サイクル数を追跡します。 ハードウェアの一部の要素が遅延を引き起こしている場合、またはリアルタイムスレッドの設定が速すぎる場合は、このエラーが発生します。

NOTE

このエラーは、セッションごとに 1 回だけ表示されます。 BASE_PERIOD が低すぎる場合、複数のエラーメッセージが表示されると、1 秒あたり数十万のエラーメッセージが表示される可能性があります。

6.4.3 テスト

6.4.3.1 ステップタイミング

複数の移動で軸が間違った位置に配置されている場合は、ステッパードライバーの正しい方向保持時間またはステップタイミングがない可能性があります。 それぞれの方向転換は、1 ステップ以上を失う可能性があります。 モーターが停止している場合は、MAX_ACCELERATION または MAX_VELOCITY のいずれかをその軸に対して高すぎる値に設定している可能性もあります。

次のプログラムは、適切なセットアップのために Z 軸構成をテストします。 プログラムを～/EMC2 / NC_FILES ディレクトリにコピーし、TESTZ.NGC などの名前を付けます。 テーブルトップで Z = 0.000 でマシンをゼロにします。 プログラムをロードして実行します。 0.5 から 1 "まで 200 回前後に移動します。構成に問題がある場合は、軸ウィンドウに表示されている最終的な位置が 0.500" にならないことがわかります。 別の軸をテストするには、Z を G0 ラインの軸に置き換えるだけです。

```
( test program to see if Z axis loses position )
( msg, test 1 of Z axis configuration )
G20 #1000=100 ( loop 100 times )
```

```
( this loop has delays after moves )
( tests acc and velocity settings )
o100 while [#1000]
G0 Z1.000
G4 P0.250
G0 Z0.500
G4 P0.250
#1000 = [#1000 - 1]
o100 endwhile
( msg, test 2 of Z axis configuration S to continue)
M1 (stop here)
#1000=100 ( loop 100 times )
( the next loop has no delays after moves )
( tests direction hold times on driver config and also max accel setting )
o101 while [#1000]
G0 Z1.000
G0 Z0.500
#1000 = [#1000 - 1]
o101 endwhile
( msg, Done...Z should be exactly .5" above table )
M2
```

8章 構成

6.5 ステッパークイックスタート

このセクションでは、LIVECD から標準インストールを実行したことを前提としています。 インストール後、コンピューターをインターネットに接続し、更新マネージャーがポップアップして LINUXCNC と UBUNTU の最新の更新を取得するのを待ってから続行することをお勧めします。

6.5.1 レイテンシーテスト

遅延テストは、コンピュータプロセッサが要求に応答するのにどれだけ遅れているかを判断します。 一部のハードウェアは処理を中断する可能性があり、CNC マシンの実行時にステップの欠落を引き起こす可能性があります。 これはあなたがする必要がある最初のことです。 こちらの手順に従って、レイテンシテストを実行してください。

6.5.2 SHERLINE

SHERLINE を使用している場合は、いくつかの事前定義された構成が提供されます。 これはメインメニュー CNC / EMC にあり、自分に一致する SHERLINE 構成を選択してコピーを保存します

6.5.3 XYLOTEX

XYLOTEX をお持ちの場合は、次のセクションをスキップして、STEPPERCONFIGWIZARD に直接移動できます。 LINUXCNC は、XYLOTEX マシンのクイックセットアップを提供しています。

6.5.4 機械情報

マシンの各軸に関する情報を収集します。

ドライブのタイミングはナノ秒単位です。 タイミングがわからない場合は、多くの一般的なドライブがステッパー構成ウィザードに含まれています。 一部の新しい GECKO ドライブは、元のドライブとはタイミングが異なることに注意してください。 リストは、より多くのドライブのユーザーが管理する LINUXCNCWIKI サイトにもあります。

Axis	Drive Type	Step Time ns	Step Space ns	Dir. Hold ns	Dir. Setup ns
X					
Y					
Z					

6.5.5 ピン配置情報

マシンから PC のパラレルポートへの接続に関する情報を収集します。

Output Pin	Typ. Function	If Different	Input Pin	Typ. Function	If Different
1	E-Stop Out		10	X Limit/Home	
2	X Step		11	Y Limit/Home	
3	X Direction		12	Z Limit/Home	
4	Y Step		13	A Limit/Home	
5	Y Direction		15	Probe In	
6	Z Step				
7	Z Direction				
8	A Step				
9	A Direction				
14	Spindle CW				
16	Spindle PWM				
17	Amplifier Enable				

使用しないピンは、ドロップダウンボックスで[未使用]に設定する必要があることに注意してください。これらは、後で STEPCONF を再度実行することでいつでも変更できます。

6.5.6 機械的情報

ステップとギアリングに関する情報を収集します。 この結果は、.INI ファイルの SCALE に使用されるユーザー単位あたりのステップ数です。

Axis	Steps/Rev.	Micro Steps	Motor Teeth	Leadscrew Teeth	Leadscrew Pitch
X					
Y					
Z					

- 1 回転あたりのステップ数-ステッピングモーターを 1 回転させるのに必要なステッピングモーターのステップ数です。 通常は 200 です。
- マイクロステップ-ドライブがステッピングモーターを 1 ステップ移動するために必要なステップ数です。 マイクロステッピングを使用しない場合、この数値は 1 になります。マイクロステッピングを使用する場合、値はステッパードライブのハードウェアによって異なります。
- モーターの歯と親ねじの歯-モーターと親ねじの間に何らかの減少（ギア、チェーン、タイミングベルトなど）がある場合です。 そうでない場合は、これら両方を 1 に設定します。
- 親ねじピッチ-1 回の親ねじ回転で発生する移動量（ユーザー単位）です。 インチで設定している場合は、1 ターンあたりのインチです。 ミリメートルで設定している場合は、1 ターンあたりのミリメートルです。

探している最終的な結果は、1 つのユーザーユニット（インチまたは MM）を移動するのに必要な CNC 出力ステップの数です。

例 8.1 単位インチ

```
Stepper = 200 steps per revolution
Drive = 10 micro steps per step
```

Motor Teeth = 20

Leadscrew Teeth = 40

Leadscrew Pitch = 0.2000 inches per turn

上記の情報から、親ねじは 1 回転あたり 0.200 インチ移動します。 -モーターは、親ねじ 1 回転あたり 2.000 回回転します。 -ドライブは、ステッパーを 1 回ステップさせるために、10 マイクロステップの入力を取ります。 -ステッパーを 1 回転させるには、ドライブに 2000 ステップが必要です。 したがって、必要なスケールは次のとおりです。

$$\frac{200 motor steps}{1 motor rev} \times \frac{10 microsteps}{1 motor step} \times \frac{2 motor revs}{1 leadscrew rev} \times \frac{1 leadscrew revs}{0.2000 inch} = \frac{20{,}000 microsteps}{inch}$$

例 8.2 単位 MM

Stepper = 200 steps per revolution

Drive = 8 micro steps per step

Motor Teeth = 30

Leadscrew Teeth = 90

Leadscrew Pitch = 5.00 mm per turn

上記の情報から：-親ねじは 1 回転あたり 5.00MM 移動します。 -モーターは、親ねじの 1 回転あたり 3.000 回回転します。 -ドライブは、ステッパーを 1 回ステップさせるために、8 つのマイクロステップ入力を取ります。 -ステッパーを 1 回転させるには、ドライブに 1600 ステップが必要です。 したがって、必要なスケールは次のとおりです。

$$\frac{200\,full\,steps}{1\,rev} x \frac{8\,microsteps}{1\,step} x \frac{3\,revs}{1\,leadscrew\,rev} x \frac{1\,leadscrew\,rev}{5.00\text{mm}} = \frac{960\,steps}{1\,mm}$$

6.6 INI の構成

6.6.1 INI ファイルコンポーネント

典型的な INI ファイルは、以下を含むかなり単純なレイアウトに従います。

- コメント
- セクション
- 変数

これらの各要素は 1 行で区切られています。 行末または改行文字はそれぞれ、新しい要素を作成します。

1.1.1.1 コメント

コメント行は;で始まります。 または＃マーク。 INI リーダーが行の先頭にこれらのマークのいずれかを表示すると、行の残りの部分はソフトウェアによって無視されます。 コメントは、INI 要素が何をするかを説明するために使用できます。

```
; This is my mill configuration file.
# I set it up on January 12, 2012
```

コメントを使用して、変数をオフにすることもできます。 これにより、異なる変数を簡単に選択できます。

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

このリストでは、もう一方がコメントアウトされているため、DISPLAY 変数は AXIS に設定されます。 誰かがこのようなリストを不注意に編集し、2 行のコメントを外したままにすると、最初に検出された行が使用されます。

```
INCORRECT = value # and a comment
# Correct Comment
CORRECT = value
```

6.6.1.1 セクション

INI ファイルの関連部分はセクションに分かれています。 セクション名は次のように角かっこで囲まれています[THIS_SECTION]セクションの順序は重要ではありません。 セクションはセクション名で始まり、次のセクション名で終わります。

次のセクションは、LINUXCNC によって使用されます。

- [EMC]一般情報
- [DISPLAY]グラフィカルユーザーインターフェースに関連する設定
- [FILTER]設定入力フィルタープログラム
- [RS274NGC] G コードインタープリターが使用する設定
- [EMCMOT]リアルタイムモーションコントローラーで使用する設定
- [TASK]タスクコントローラーが使用する設定
- [HAL]は.HAL ファイルを指定します
- [HALUI] HALUI が使用する MDI コマンド
- [APPLICATIONS] LINUXCNC が起動するその他のアプリケーション
- [TRAJ]リアルタイムモーションコントローラーが使用する追加設定

- [JOINT_N]個々の関節変数
- [AXIS_N]個々の軸変数
- [KINS]キネマティクス変数
- [EMCIO] I / O コントローラーが使用する設定

6.6.1.2 変数

変数行は、変数名、等号（=）、および値で構成されます。 =の後の最初の空白以外の文字から行の終わりまでのすべてが値として渡されるため、必要に応じて文字列記号にスペースを埋め込むことができます。 変数名はしばしばキーワードと呼ばれます。

変数の例

```
MACHINE = My Machine
```

可変行は、終端の円記号（\）文字を使用して複数の行に拡張できます。 最大MAX_EXTEND_LINES（== 20）が許可されます。 末尾の円記号の後に空白があってはなりません。

セクション識別子を複数行に拡張することはできません。

LINEEXTENDS の変数の例

```
APP = sim_pin \
ini.0.max_acceleration \
ini.1.max_acceleration \
ini.2.max_acceleration \
ini.0.max_velocity \
ini.1.max_velocity \
ini.2.max_velocity
```

次のセクションでは、構成行のサンプル値を使用して、構成ファイルの各セクションについて詳しく説明します。

LINUXCNC で使用される変数は、示されているように常にセクション名と変数名を使用する必要があります。 次の例では、変数 MACHINE に値 MYMACHINE が割り当てられています。

6.6.1.3 カスタムセクションと変数

ほとんどのサンプル構成では、カスタムセクションと変数を使用して、便宜上すべての設定を 1 つの場所に配置しています。 既存の LINUXCNC セクションにカスタム変数を追加するには、そのセクションに変数を含めるだけです。

カスタム変数の例

```
[JOINT_0]
TYPE = LINEAR
...
SCALE = 16000
```

独自の変数を持つカスタムセクションを導入するには、セクションと変数を INI ファイルに追加します。

カスタムセクションの例

```
[PROBE]
Z_FEEDRATE = 50
Z_OFFSET = 12
Z_SAFE_DISTANCE = -10
```

HAL ファイルでカスタム変数を使用するには、値の代わりにセクションと変数名を入力します。

HAL の例

```
setp offset.1.offset [PROBE]Z_OFFSET
setp stepgen.0.position-scale [JOINT_0]SCALE
```

NOTE

変数に格納される値は、コンポーネントピンで指定されたタイプと一致する必要があります。

G コードでカスタム変数を使用するには、グローバル変数構文＃<_ INI [SECTION] VARIABLE>を使用します。 次の例は、プローブプレートを使用したルーターまたはミルの簡単な Z 軸タッチオフルーチンを示しています。

G コードの例

```
G91
G38.2 Z#<_ini[probe]z_safe_distance> F#<_ini[probe]z_feedrate>
G90
G1 Z#5063
G10 L20 P0 Z#<_ini[probe]z_offset>
```

6.6.1.4 ファイルを含める

INI ファイルには、＃INCLUDE ディレクティブを使用して別のファイルの内容を含めることができます。

#INCLUDE フォーマット

```
INCLUDE filename
```

ファイル名は次のように指定できます。

- INI ファイルと同じディレクトリにあるファイル
- 作業ディレクトリに関連して配置されたファイル
- 絶対ファイル名（/で始まる）

- ユーザーホーム相対ファイル名（～で始まる）

複数の#INCLUDE ディレクティブがサポートされています。

#INCLUDE の例

```
#INCLUDE joint_0.inc
#INCLUDE ../parallel/joint_1.inc
#INCLUDE below/joint_2.inc
#INCLUDE /home/myusername/myincludes/display.inc
#INCLUDE ~/linuxcnc/myincludes/rs274ngc.inc
```

#INCLUDE ディレクティブは、1 レベルの拡張でのみサポートされます。インクルードされたファイルには、追加のファイルが含まれない場合があります。 推奨されるファイル拡張子は.INC です。 インクルードファイルに.INI のファイル拡張子を使用しないでください。

6.6.2 INI ファイルセクション

1.1.1.1 [EMC]セクション

- VERSION = 1.1-構成のバージョン番号。 1.1 以外の値を指定すると、構成チェッカーが実行され、構成が新しいスタイルのジョイント軸タイプの構成に更新されます。
- MACHINE = MY CONTROLLER-これはコントローラーの名前であり、ほとんどのグラフィカルインターフェイスの上部に出力されます。 一行の長さにすれば、ここに好きなものを入れることができます。
- DEBUG = 0-デバッグレベル 0 は、LINUXCNC が端末から実行されたときにメッセージが出力されないことを意味します。 デバッグフラグは通常、開発者にのみ役立ちます。 その他の設定については、SRC / EMC / NML_INTF /DEBUGFLAGS.H を参照してください。

6.6.2.1 [DISPLAY]セクション

異なるユーザーインターフェイスプログラムは異なるオプションを使用し、すべてのオプションがすべてのユーザーインターフェイスでサポートされているわけではありません。
AXIS、GMOCCAPY、TOUCHY、QTVCP の QTDRAGON、GSCREEN などのいくつかのインターフェースがあります。 AXIS は通常のコンピューターとモニターで使用するためのインターフェースであり、TOUCHY はタッチスクリーンで使用するためのものです。 GMOCCAPY は両方の方法で使用でき、ハードウェア制御用の多くの接続も提供します。 インターフェイスの説明は、ユーザーマニュアルの「インターフェイス」セクションにあります。

- DISPLAY = AXIS-使用するユーザーインターフェイスの名前。 有効なオプションには、AXIS、TOUCHY、GMOCCAPY、GSCREEN、TKLINUXCNC、QTVCP などがあります。

- POSITION_OFFSET = RELATIVE-ユーザーインターフェイスの起動時に DRO に表示する座標系（RELATIVE または MACHINE）。 RELATIVE 座標系は、現在有効な G92 および G5x 座標オフセットを反映しています。
- POSITION_FEEDBACK = COMMANDED-ユーザーインターフェイスの起動時に DRO に表示される座標値（COMMANDED または ACTUAL）。 AXIS では、これは[表示]メニューから変更できます。 COMMANDED ポジションは、LINUXCNC によって要求されたポジションです。 実際の位置は、ほとんどのサーボシステムのようにモーターにフィードバックがある場合のモーターのフィードバック位置です。 通常、COMMANDED 値が使用されます。
- DRO_FORMAT_MM =%+ 08.6F-メトリックモードのデフォルトの DRO フォーマットを上書きします。 （通常、小数点以下 3 桁、左側に 6 桁のスペースが埋め込まれます）上記の例では、ゼロが埋め込まれ、小数点以下 6 桁が表示され、正の数には+記号が強制的に表示されます。 フォーマットは PYTHON の慣習に従います。 HTTPS://DOCS.PYTHON.ORG/2/LIBRARY/STRING.HTML#FORMAT-SPECIFICATIONMINI-LANGUAGE 形式が浮動小数点値を受け入れることができない場合、エラーが発生します。
- DRO_FORMAT_IN =%4.1F-インペリアルモードのデフォルトの DRO フォーマットを上書きします。 （通常は小数点以下 4 桁、左側に 6 桁のスペースが埋め込まれます）上記の例では、小数点以下 1 桁のみが表示されます。 フォーマットは PYTHON の慣習に従います。 HTTPS://DOCS.PYTHON.ORG/2/LIBRARY/STRING.HTML#FORMAT-SPECIFICATION-MINI-LANGUAGE フォーマットが浮動小数点値を受け入れることができない場合、エラーが発生します。
- CONE_BASESIZE = .25-グラフィックディスプレイのデフォルトのコーン/ツールベースサイズである.5 を上書きします
- MAX_FEED_OVERRIDE = 1.2-ユーザーが選択できる最大フィードオーバーライド。 1.2 は、プログラムされた送り速度の 120%を意味します。
- MIN_SPINDLE_OVERRIDE = 0.5-ユーザーが選択できる最小スピンドルオーバーライド。 0.5 は、プログラムされたスピンドル速度の 50%を意味します。 （これは、最小スピンドル速度を設定するために使用されます）。
- MIN_SPINDLE_0_OVERRIDE = 0.5-ユーザーが選択できる最小スピンドルオーバーライド。 0.5 は、プログラムされたスピンドル速度の 50%を意味します。 （これは、最小スピンドル速度を設定するために使用されます）。 マルチスピンドルマシンでは、各スピンドル番号のエントリがあります。 QTVCP のみ
- MAX_SPINDLE_OVERRIDE = 1.0-ユーザーが選択できる最大スピンドルオーバーライド。 1.0 は、プログラムされたスピンドル速度の 100%を意味します。

- MAX_SPINDLE_0_OVERRIDE = 1.0-ユーザーが選択できる最大送りオーバーライド。 1.2 は、プログラムされた送り速度の 120%を意味します。 マルチスピンドルマシンでは、各スピンドル番号のエントリがあります。 QTVCP のみ
- DEFAULT_SPINDLE_SPEED = 100-スピンドルが手動モードで起動されたときのデフォルトのスピンドル RPM。 この設定が存在しない場合、これはデフォルトで AXIS の場合は 1 RPM、GMOCCAPY の場合は 300RPM になります。
- DEFAULT_SPINDLE_0_SPEED = 100-スピンドルが手動モードで起動されたときのデフォルトのスピンドル RPM。 マルチスピンドルマシンでは、各スピンドル番号のエントリがあります。 QTVCP のみ
- SPINDLE_INCREMENT = 200-増加/減少ボタンをクリックしたときに使用される増分 QTVCP のみ
- MIN_SPINDLE_0_SPEED = 1000-手動で選択できる最小 RPM。 マルチスピンドルマシンでは、各スピンドル番号のエントリがあります。 QTVCP のみ
- MAX_SPINDLE_0_SPEED = 20000-手動で選択できる最大 RPM。 マルチスピンドルマシンでは、各スピンドル番号のエントリがあります。 QTVCP のみ
- PROGRAM_PREFIX =~/ LINUXCNC / NC_FILES-G コードファイルのデフォルトの場所とユーザー定義の M コードの場所。 [RS274NGC]セクションで指定されている場合、この場所でサブルーチンパスおよびユーザー M パスの前のファイル名が検索されます。
- INTRO_GRAPHIC = EMC2.GIF-スプラッシュ画面に表示される画像。
- INTRO_TIME = 5-スプラッシュ画面を表示する最大時間（秒単位）。
- CYCLE_TIME = 0.05-表示がポーリング間でスリープするサイクルタイム（秒単位）。

NOTE

次の[DISPLAY]アイテムは GLADEVCP によって使用されます。GLADEVCP の章のタブセクションの埋め込みを参照してください。

- EMBED_TAB_NAME = GLADEVCP デモ

NOTE

異なるユーザーインターフェイスプログラムは異なるオプションを使用し、すべてのオプションがすべてのユーザーインターフェイスでサポートされているわけではありません。 AXIS の詳細については、AXISGUI ドキュメントを参照してください。 GMOCCAPY の詳細については、GMOCCAPY ドキュメントを参照してください。

- DEFAULT_LINEAR_VELOCITY = .25-線形ジョグのデフォルト速度（1 秒あたりのマシン単位）。
- MIN_VELOCITY = .01-ジョグスライダーのおおよその最小値。
- MAX_LINEAR_VELOCITY = 1.0-リニアジョグの最大速度（マシン単位/秒）。
- MIN_LINEAR_VELOCITY = .01-ジョグスライダーのおおよその最小値。
- DEFAULT_ANGULAR_VELOCITY = .25-角ジョグのデフォルト速度（マシン単位/秒）。
- MIN_ANGULAR_VELOCITY = .01-角ジョグスライダーのおおよその最小値。
- MAX_ANGULAR_VELOCITY = 1.0-角ジョグの最大速度（マシン単位/秒）。
- 増分= 1 MM、.5 インチ、。 。 。 -インクリメンタルジョグに使用できるインクリメンタルを定義します。 INCREMENTS を使用して、デフォルトをオーバーライドできます。 値は、10 進数（例：0.1000）または小数（例：1/16）で、オプションで単位（CM、MM、UM、INCH、IN、または MIL）が続きます。 単位が指定されていない場合は、機械単位が想定されます。 メートル法とインペリアル距離は混在する場合があります。増分= 1 インチ、1 ミル、1 CM、1 MM、1UM が有効なエントリです。
- グリッド= 10 MM、1 インチ、。 。 。 -グリッド線のプリセット値を定義します。 値は INCREMENTS と同じように解釈されます。
- OPEN_FILE = / FULL / PATH / TO / FILE.NGC-AXIS の起動時にプレビュープロットに表示するファイル。 空白の文字列 ""を使用すると、起動時にファイルが読み込まれません。 GMOCCAPY は、設定ページに対応するエントリを提供するため、この設定を使用しません。
- EDITOR = GEDIT-AXIS メニューから G コードを編集するために[ファイル]> [編集]を選択するときに使用するエディター。 このメニュー項目を機能させるには、これを構成する必要があります。 もう 1 つの有効なエントリは GNOME-TERMINAL-EVIM です。 GMOCCAPY にはエディターが統合されているため、このエントリは GMOCCAPY には適用されません。
- TOOL_EDITOR = TOOLEDIT-ツールテーブルを編集するときに使用するエディタ（たとえば、AXIS で[ファイル]> [ツールテーブルの編集...]を選択する）。 その他の有効なエントリは、「GEDIT」、「GNOME-TERMINAL -E VIM」、および「GVIM」です。 GMOCCAPY にはエディターが統合されているため、このエントリは GMOCCAPY には適用されません。
- PYVCP = / FILENAME.XML-PYVCP パネル記述ファイル。 詳細については、PYVCP の章を参照してください。
- PYVCP_POSITION = BOTTOM-AXIS ユーザーインターフェイスでの PYVCP パネルの配置。 この変数を省略すると、パネルはデフォルトで右側になります。 唯一の有効な代替手段は BOTTOM です。 詳細については、PYVCP の章を参照してください。

- LATHE = 1-空でない値（「0」を含む）があると、軸は上面図と DRO の半径と直径で「旋盤モード」を使用します。
- BACK_TOOL_LATHE = 1-空でない値（「0」を含む）があると、軸は X 軸が反転した「バックツール旋盤モード」を使用します。
- FOAM = 1-空でない値（「0」を含む）があると、軸はフォームカッターモードの表示を変更します。
- GEOMETRY = XYZABCUVW-回転運動のプレビューとバックプロットを制御します。このアイテムは一連の軸文字で構成され、オプションで「-」記号が前に付きます。このシーケンスは、各軸の効果が適用される順序を指定し、「-」は回転の意味を反転させます。適切な GEOMETRY ストリングは、マシン構成とそれを制御するために使用される運動学によって異なります。例の文字列 GEOMETRY = XYZBCUVW は、キネマティクスによって UVW がツールの座標系で移動し、XYZ が材料の座標系で移動する 5 軸マシン用です。文字の順序は、さまざまな変換が適用される順序を表すため、重要です。たとえば、C を中心に B を回転させることは、B を中心に C を中心に回転させることとは異なります。ジオメトリは、回転軸がないと効果がありません。フォームカッティングマシン（FOAM = 1）は、「XY; UV」を指定するか、フォームカッターモードでこの値が現在無視されている場合でも、値を空白のままにする必要があります。将来のバージョンでは、「;」が定義される可能性があります。を意味しますが、「XY; UV」を実行すると、現在のフォームのデフォルトと同じ意味になります。
- ARCDIVISION = 64-アークのプレビューの品質を設定します。 円弧は、いくつかの直線に分割してプレビューされます。 半円は ARCDIVISION の部分に分割されます。 値を大きくするとプレビューがより正確になりますが、読み込みに時間がかかり、表示が遅くなります。 値を小さくするとプレビューの精度は低下しますが、読み込みにかかる時間が短くなり、表示が速くなる可能性があります。 デフォルト値の 64 は、最大 3 インチの円が 1 MIL（.03%）以内に表示されることを意味します。
- MDI_HISTORY_FILE =-ローカル MDI 履歴ファイルの名前。 これが指定されていない場合、AXIS は MDI 履歴をユーザーのホームディレクトリの.AXIS_MDI_HISTORY に保存します。 これは、1 台のコンピューターに複数の構成がある場合に役立ちます。
- JOG_AXES =-ジョグキーが軸文字に割り当てられる順序。 左矢印と右矢印は最初の軸文字に割り当てられ、上下は 2 番目に、PAGE UP / PAGE DOWN は 3 番目に、左右の括弧は 4 番目に割り当てられます。 指定しない場合、デフォルトは[TRAJ] COORDINATES、[DISPLAY] LATHE、および[DISPLAY] FOAM の値から決定されます。
- JOG_INVERT =-軸文字ごとに、ジョグ方向が反転します。 デフォルトは旋盤の場合は「X」で、それ以外の場合は空白です。

Note

JOG_AXES および JOG_INVERT の設定は、軸座標文字によるワールドモードジョギングに適用され、ホーミングが成功した後、ワールドモードで有効になります。 原点復帰前にジョイントモードで操作する場合、キーボードジョグキーは固定順序で割り当てられます：左/右：joint0、上/下：joint1、ページアップ/ページダウン：joint2、左/右ブラケット：joint3

- USER_COMMAND_FILE = MYCOMMANDS.PY-ユーザー固有のファイル〜/ .AXISRC ではなく、AXISGUI によって供給されるオプションの構成固有の PYTHON ファイルの名前。

NOTE

次の[DISPLAY]項目は、TKLINUXCNC インターフェースでのみ使用されます。

- HELP_FILE = TKLINUCNC.TXT-ヘルプファイルへのパス。

6.6.2.2 [FILTER] セクション

AXIS と GMOCCAPY には、ロードされたファイルをフィルタープログラムを介して送信する機能があります。 このフィルターは、ファイルが M2 で終わることを確認するような単純なものから、入力が深度画像であるかどうかを検出するような複雑なもの、定義する形状をミリングするための G コードの生成などの任意のタスクを実行できます。 INI ファイルの[FILTER]セクションは、フィルターの動作を制御します。 まず、ファイルの種類ごとに、PROGRAM_EXTENSION 行を記述します。 次に、ファイルの種類ごとに実行するプログラムを指定します。 このプログラムには、最初の引数として入力ファイルの名前が指定されており、RS274NGC コードを標準出力に書き込む必要があります。 この出力は、テキスト領域に表示され、表示領域でプレビューされ、実行時に LINUXCNC によって実行されるものです。

- PROGRAM_EXTENSION = .EXTENSION 説明

ポストプロセッサがすべて大文字でファイルを出力する場合は、次の行を追加することをお勧めします。

- PROGRAM_EXTENSION = .NGCXYZ ポストプロセッサ

次の行は、LINUXCNC に含まれている画像から G コードへのコンバーターのサポートを追加します。

- プログラム拡張子= .PNG、.GIF、.JPG グレースケール深度画像
  - PNG = IMAGE-TO-GCODE
  - GIF = IMAGE-TO-GCODE
  - JPG = IMAGE-TO-GCODE

LINUXCNC ディレクトリにあるカスタム G コードコンバーターの例。

- PROGRAM_EXTENSION = .GCODE3D プリンター
  - GCODE = /HOME/MILL/LINUXCNC/CONVERT.PY

NOTE

拡張機能に関連付けられたプログラムファイルは、プログラムへのフルパスを持っているか、システムパス上にあるディレクトリに配置されている必要があります。

インタプリタを指定することもできます。

- PROGRAM_EXTENSION = .PYPYTHON スクリプト
  - PY = PYTHON

このようにして、任意の PYTHON スクリプトを開くことができ、その出力は G コードとして扱われます。 そのようなサンプルスクリプトの 1 つは、NC_FILES /HOLECIRCLE.PY で入手できます。 このスクリプトは、円の円周に沿って一連の穴を開けるための G コードを作成します。 LINUXCNC WIKI サイト HTTP://WIKI.LINUXCNC.ORG/には、さらに多くの G コードジェネレーターがあります。

環境変数 AXIS_PROGRESS_BAR が設定されている場合、次の形式の STDERR に書き込まれる行

- FILTER_PROGRESS=%D

AXIS プログレスバーを指定されたパーセンテージに設定します。 この機能は、長時間実行されるすべてのフィルターで使用する必要があります。 PYTHON フィルターは、PRINT 関数を使用して結果を AXIS に出力する必要があります。

このサンプルプログラムは、ファイルをフィルタリングし、Z 軸に一致するように W 軸を追加します。 各軸の単語の間にスペースがあるかどうかによって異なります。

```
#!/usr/bin/env python
import sys
def main(argv):
openfile = open(argv[0], 'r')
file_in = openfile.readlines()
openfile.close()
file_out = []
for line in file_in:
# print line
if line.find('Z') != -1:
words = line.rstrip('\n')
words = words.split(' ')
newword = ''
for i in words:
if i[0] == 'Z':
```

```
newword = 'W'+ i[1:]
if len(newword) > 0:
words.append(newword)
newline = ' '.join(words)
file_out.append(newline)
else:
file_out.append(line)
for item in file_out:
print "%s" % item
if __name__ == "__main__":
main(sys.argv[1:])
```

6.6.2.3 [RS274NGC]セクション

- PARAMETER_FILE = MYFILE.VAR-インタプリタが使用するパラメータを含む INI ファイルと同じディレクトリにあるファイル（実行間で保存）。
- ORIENT_OFFSET = 0-M19 オリエントスピンドル操作の R ワードパラメータに追加された浮動小数点値。 エンコーダマウントの向きに関係なく、任意のゼロ位置を定義するために使用されます。
- RS274NGC_STARTUP_CODE = G17 G20 G40 G49 G64 P0.001 G80 G90 G92 G94 G97G98-インタプリタが初期化される NC コードの文字列。 マシンのモーダルコードは異なり、セッションの前半で解釈された G コードによって変更される可能性があるため、これは各 NGC ファイルの先頭にモーダル G コードを指定する代わりにはなりません。
- SUBROUTINE_PATH = NCSUBROUTINES：/ TMP / TESTSUBS：LATHESUBS：MILLSUBS-単一ファイルのサブルーチンが GCODE で指定されている場合に、検索される最大 10 個のディレクトリのコロン（:)区切りのリストを指定します。 これらのディレクトリは、[DISPLAY] PROGRAM_PREFIX（指定されている場合）を検索した後、[WIZARD] WIZARD_ROOT（指定されている場合）を検索する前に検索されます。 パスは、リストされている順序で検索されます。 検索で最初に一致したサブルーチンファイルが使用されます。 ディレクトリは、INI ファイルの現在のディレクトリを基準にして、または絶対パスとして指定されます。 リストには、間に空白が含まれていてはなりません。
- CENTER_ARC_RADIUS_TOLERANCE_INCH = N デフォルト 0.00005
- CENTER_ARC_RADIUS_TOLERANCE_MM = N デフォルト 0.00127
- USER_M_PATH = MYFUNCS：/ TMP / MCODES：EXPERIMENTALMCODES-ユーザー定義関数のコロン（:)で区切られたディレクトリのリストを指定します。 ディレクトリは、INI ファイルの現在のディレクトリを基準にして、または絶対パスとして指定されます。 リストには、間に空白が含まれていてはなりません。

通常、ユーザー定義関数（M100-M199）ごとに検索が行われます。 検索順序は次のとおりです。

1. [DISPLAY] PROGRAM_PREFIX（指定されている場合）
2. [DISPLAY] PROGRAM_PREFIX が指定されていない場合は、デフォルトの場所 NC_FILES を検索します。
3. 次に、リスト[RS274NGC] USER_M_PATH の各ディレクトリを検索します
   検索で最初に見つかった実行可能 M1XX は、各 M1XX に使用されます。

NOTE

USER_M_PATH ディレクトリの最大数は、コンパイル時に定義されます（TYP：USER_DEFINED_FUNCTION_MAX_DIRS == 5）。

- INI_VARS = 1 デフォルト 1G コードプログラムが#<_ INI [SECTION] NAME>の形式を使用して INI ファイルから値を読み取ることを許可します。 << GCODE：PARAMETERS、G-CODEPARAMETERS>を参照してください
- HAL_PIN_VARS = 1 デフォルト 1G コードプログラムが#<_ HAL [HAL ITEM]>の形式を使用して HAL ピンの値を読み取ることができるようにします。変数アクセスは読み取り専用です。詳細と重要な注意事項については、G コードパラメータを参照してください。
- RETAIN_G43 = 0 デフォルト 0 設定すると、最初のツールをロードした後に G43 をオンにでき、プログラムを介して心配する必要はありません。 最後の工具を最終的にアンロードすると、G43 モードはキャンセルされます。
- OWORD_NARGS = 0 デフォルト 0 この機能が有効になっている場合、呼び出されたサブルーチンは、#<N_ARGS>パラメーターを検査することによって渡された実際の位置パラメーターの数を判別できます。
- NO_DOWNCASE_OWORD = 0 デフォルト 0 設定されている場合、コメント内の O-WORD 名の大文字と小文字を保持し、（DEBUG、#<_ HAL [MIXEDCASEITEM]）のような構造化コメント内の大文字と小文字が混在する HAL アイテムの読み取りを有効にします。
- OWORD_WARNONLY = 0 デフォルト 0O-WORD サブルーチンでエラーが発生した場合、エラーではなく警告します。

[注]上記の 6 つのオプションは、2.8 より前のバージョンの LINUXCNC の FEATURES ビットマスクによって制御されていました。 この INI タグは機能しなくなります。

NOTE

[WIZARD] WIZARD_ROOT は有効な検索パスですが、ウィザードが完全に実装されておらず、使用した結果が予測できません。

- REMAP = M400 MODALGROUP = 10 ARGSPEC = PQ NGC = MYPROCEDURE 詳細については、「G コードの拡張の再マップ」の章を参照してください。
- ON_ABORT_COMMAND = O <ON_ABORT>呼び出し詳細については、「G コードの拡張の再マップ」の章を参照してください。

6.6.2.4 [EMCMOT]セクション

このセクションはカスタムセクションであり、LINUXCNC によって直接使用されることはありません。 ほとんどの構成では、このセクションの値を使用してモーションコントローラをロードします。 モーションコントローラの詳細については、モーションセクションを参照してください。

- EMCMOT = MOTMOD-モーションコントローラー名は通常ここで使用されます。
- BASE_PERIOD = 50000-ナノ秒単位の基本タスク期間。
- SERVO_PERIOD = 1000000-これは、ナノ秒単位の「サーボ」タスク期間です。
- TRAJ_PERIOD = 100000-これは、ナノ秒単位の TRAJECTORYPLANNER タスク期間です。
- COMM_TIMEOUT = 1.0-モーション（モーションコントローラーのリアルタイム部分）がタスク（モーションコントローラーの非リアルタイム部分）からのメッセージの受信を確認するのを待機する秒数。

6.6.2.5 [TASK] セクション

- TASK = MILLTASK-タスク実行可能ファイルの名前を指定します。 タスク実行可能ファイルは、NML を介した UI との通信、非 HAL 共有メモリを介したリアルタイムモーションプランナーとの通信、GCODE の解釈など、さまざまなことを行います。 現在、99.9%のユーザーにとって意味のあるタスク実行可能ファイルは MILLTASK だけです。
- CYCLE_TIME = 0.010-TASK が実行される期間（秒単位）。 このパラメーターは、モーションの完了を待機するとき、一時停止命令を実行するとき、およびユーザーインターフェイスからコマンドを受け入れるときのポーリング間隔に影響します。 通常、この番号を変更する必要はありません。

6.6.2.6 [HAL]セクション

- HALFILE = EXAMPLE.HAL-起動時にファイル EXAMPLE.HAL を実行します。 HALFILE が複数回指定されている場合、ファイルは INI ファイルに表示されている順序で実行されます。 ほとんどすべての構成には少なくとも 1 つの HALFILE があり、ステッパーシステムには通常 2 つのそのようなファイルがあります。1 つは一般的なステッパー構成（CORE_STEPPER.HAL）を指定し、もう 1 つはマシンのピン配置（XXX_PINOUT.HAL）を指定します。 HALFILES は、検索を使用し

て検索されます。 指定されたファイルが INI ファイルを含むディレクトリで見つかった場合は、それが使用されます。 指定されたファイルがこの INI ファイルディレクトリに見つからない場合は、システムライブラリを使用して検索が行われます。 ハーフイルの。 HALFILE は、絶対パスとして指定することもできます（名前が/文字で始まる場合）。 絶対パスを使用すると構成の再配置が制限される可能性があるため、絶対パスはお勧めしません。

- HALFILE = TEXAMPLE.TCL [ARG1 [ARG2]。 。 。]]-起動時に、ARG1、ARG2 などを:: ARGV リストとして TCL ファイル TEXAMPLE.TCL を実行します。 .TCL サフィックスが付いたファイルは上記のように処理されますが、処理には HALTCL を使用します。詳細については、HALTCL の章を参照してください。
- HALFILE = LIB：SYS_EXAMPLE.HAL-起動時にシステムライブラリファイル SYS_EXAMPLE.HAL を実行します。 LIB：プレフィックスを明示的に使用すると、INI ファイルディレクトリを検索せずにシステムライブラリ HALFILE が使用されます。
- HALFILE = LIB：SYS_TEXAMPLE.TCL [ARG1 [ARG2。 。 。]]-起動時にシステムライブラリファイル SYS_TEXAMPLE.TCL を実行します。 LIB：プレフィックスを明示的に使用すると、INI ファイルディレクトリを検索せずにシステムライブラリ HALFILE が使用されます。 HALFILE アイテムは、HAL コンポーネントをロードし、コンポーネントピン間で信号接続を行うファイルを指定します。 よくある間違いは、1）コンポーネントの関数をスレッドに追加するために必要な ADDF ステートメントの省略、2）不完全なシグナル（ネット）指定子です。 必要な ADDF ステートメントの省略は、ほとんどの場合エラーです。 信号には通常、1 つ以上の入力接続と 1 つの出力が含まれます（ただし、両方が厳密に必要なわけではありません）。 これらの状態をチェックし、STDOUT およびポップアップ GUI に報告するために、システムライブラリファイルが提供されています。

```
HALFILE = LIB:halcheck.tcl [ nopopup ]
```

NOTE

LIB：HALCHECK.TCL 行は最後の[HAL] HALFILE である必要があります。 NOPOPUP オプションを指定して、ポップアップメッセージを抑制し、すぐに開始できるようにします。 POSTGUI_HALFILE を使用して確立された接続はチェックされません。

- TWOPASS = ON-HAL コンポーネントのロードに 2 パス処理を使用します。 TWOPASS 処理では、[HAL] HALFILE =行は 2 つのパスで処理されます。 最初のパス（PASS0）では、すべての HALFILE が読み取られ、LOADRT および LOADUSR コマンドの複数の出現が蓄積されます。 これらの累積ロードコマンドは、PASS0 の最後に実行されます。 この累積により、特定のコンポーネントに対して負荷線を複数回指定できます（NAMES =使用される名前が使用ごとに一意

である場合）。 2 番目のパス（PASS1）では、HALFILES が再読み取りされ、以前に実行されたロードコマンドを除くすべてのコマンドが実行されます。

- TWOPASS = NODELETE VERBOSE-TWOPASS 機能は、キーワード VERBOSE および NODELETE を含む NULL 以外の任意の文字列でアクティブ化できます。 VERBOSE キーワードを使用すると、詳細が標準出力に出力されます。 NODELETE キーワードは、一時ファイルを/ TMP に保存します。

詳細については、HALTWOPASS の章を参照してください。

- HALCMD = COMMAND-コマンドを単一の HAL コマンドとして実行します。 HALCMD が複数回指定されている場合、コマンドは INI ファイルに表示されている順序で実行されます。 HALCMD 行は、すべての HALFILE 行の後に実行されます。
- SHUTDOWN = SHUTDOWN.HAL-LINUXCNC が終了しているときに、ファイル SHUTDOWN.HAL を実行します。 使用するハードウェアドライバによっては、LINUXCNC が正常に終了したときに、出力を定義された値に設定できる場合があります。 ただし、このファイルが実行される保証はないため（たとえば、コンピューターがクラッシュした場合）、適切な物理的な非常停止チェーンやその他のソフトウェア障害に対する保護に代わるものではありません。
- POSTGUI_HALFILE = EXAMPLE2.HAL-GUI が HAL ピンを作成した後、EXAMPLE2.HAL を実行します。 一部の GUI は、HAL ピンを作成し、それらを使用するための POSTGUIHALFILE の使用をサポートします。 POSTGUI ハーフイルをサポートする GUI には、TOUCHY、AXIS、GSCREEN、および GMOCCAPY が含まれます。

See section <<sec:pyvcp-with-axis,pyVCP with Axis>> Section for more information.

- HALUI = HALUI-HAL ユーザーインターフェイスピンを追加します。 詳細については、HAL ユーザーインターフェイスの章を参照してください。

6.6.2.7 [HALUI] セクション

- MDI_COMMAND = G53 G0 X0 Y0 Z0-MDI コマンドは、HALUI.MDI-COMMAND-00 を使用して実行できます。 [HALUI]セクションにリストされている各コマンドの番号をインクリメントします。

6.6.2.8 [APPLICATIONS] セクション

LINUXCNC は、指定された GUI が開始される前に他のアプリケーションを開始できます。 アプリケーションは、指定された遅延の後に起動して、GUI に依存するアクション（GUI 固有の HAL ピンの作成など）を可能にすることができます。

- DELAY = VALUE-他のアプリケーションを開始する前に待機する秒数。 アプリケーションが [HAL] POSTGUI_HALFILE アクションまたは GUI で作成された HAL ピン（デフォルトは DELAY = 0）に依存している場合は、遅延が必要になることがあります。

- APP = APPNAME [ARG1 [ARG2。 。 。]]-開始するアプリケーション。 この仕様は複数回含めることができます。 APPNAME は、絶対またはチルダで指定されたファイル名（最初の文字は/または〜）、相対ファイル名（FILENAME の最初の文字は./）、または INIFILE ディレクトリ内のファイルとして明示的に名前を付けることができます。 これらの名前を使用して実行可能ファイルが見つからない場合は、ユーザー検索 PATH を使用してアプリケーションが検索されます。

例：

テスト用に HAL ピンへの入力をシミュレートします（SIM_PIN-パラメーター、接続されていないピン、またはライターのない信号への入力を設定するための単純な GUI を使用）：

```
APP = sim_pin motion.probe-input halui.abort motion.analog-in-00
```

以前に保存したウォッチリストを使用して HALSHOW を呼び出します。 LINUXCNC は作業ディレクトリを INIFILE のディレクトリに設定するため、そのディレクトリ内のファイルを参照できます（例：MY.HALSHOW）。

```
APP = halshow my.halshow
```

または、フルパス名で識別されるウォッチリストファイルを指定することもできます。

```
APP = halshow ~/saved_shows/spindle.halshow
```

以前に保存した構成を使用して HALSCOPE を開きます。

```
APP = halscope -i my.halscope
```

6.6.2.9 [TRAJ]セクション

警告

新しい TRAJECTORYPLANNER（TP）はデフォルトでオンになっています。

[TRAJ]セクションに TP 設定がない場合、LINUXCNC のデフォルトは次のとおりです。

ARC_BLEND_ENABLE = 1

ARC_BLEND_FALLBACK_ENABLE = 0

ARC_BLEND_OPTIMIZATION_DEPTH = 50

ARC_BLEND_GAP_CYCLES = 4

ARC_BLEND_RAMP_FREQ = 100

[TRAJ]セクションには、動作中の軌道計画モジュールの一般的なパラメータが含まれています。

- ARC_BLEND_ENABLE = 1-新しい TP をオンにします。 0 に設定すると、TP は放物線ブレンディングを使用します（1 セグメント先読み）。デフォルト値は 1 です。

- ARC_BLEND_FALLBACK_ENABLE = 0-推定速度が速い場合は、オプションで放物線ブレンドにフォールバックします。 ただし、この見積もりは大まかなものであり、無効にするだけでパフォーマンスが向上するようです。 デフォルト値0。
- ARC_BLEND_OPTIMIZATION_DEPTH = 50-セグメント数の先読み深度。

これを少し拡張するために、この値をいくらか任意に選択できます。 特定の構成に必要な深さを見積もる式は次のとおりです。

＃N = V_MAX /（2.0 * A_MAX * T_C）＃ここで：＃N =最適化深度＃V_MAX =最大軸速度（UU /秒）＃A_MAX =最大軸加速度（UU /秒）＃T_C =サーボ周期（秒）

したがって、最大軸速度が10 IPS、最大加速度が100 IPSˆ2、サーボ周期が0.001秒のマシンには次のものが必要です。

10 /（2.0 * 100 * 0.001）= 50セグメントで、常に最速の軸に沿って最大速度に到達します。

実際には、非常に短いセグメントがたくさんない限り、先読みが完全な深さを必要とすることはめったにないため、この数値を調整することはそれほど重要ではありません。 テスト中に奇妙な速度低下に気づき、それらがどこから来ているのかわからない場合は、最初に上記の式を使用してこの深さを増やしてみてください。

それでも奇妙な速度低下が見られる場合は、プログラムに短いセグメントがあることが原因である可能性があります。 この場合は、ナイーブCAM検出に小さな許容値を追加してみてください。 経験則は次のとおりです。

＃MIN_LENGTH〜 = V_REQ * T_C＃ここで：＃V_REQ = UU /秒での希望の速度＃T_C =サーボ周期（秒）

1 IPS = 60 IPMのパスに沿って移動する場合、サーボ周期が0.001秒の場合、MIN_LENGTHより短いセグメントはパスの速度を低下させます。 ナイーブCAMの許容範囲をこの最小の長さに設定すると、過度に短いセグメントが組み合わされて、このボトルネックが解消されます。 もちろん、許容値を高く設定しすぎるとパスの偏差が大きくなるため、適切な値を見つけるには少し試してみる必要があります。 MIN_LENGTHの1/2から始めて、必要に応じて処理します。

- ARC_BLEND_GAP_CYCLES = 4軌道プランナーが消費する前に前のセグメントがどれだけ短くなければならないか。

多くの場合、円弧ブレンドでは、ブレンド間に短い線分が残ります。 ジオメトリは円形である必要があるため、次のラインが少し短い場合、すべてのラインをブレンドすることはできません。 軌道プランナーは各セグメントに少なくとも1回は触れる必要があるため、非常に小さなセグメントでは処理速度が大幅に低下します。 短いセグメントをブレンドアークの一部にすることで「消費」するこの方法に対する私の修正。 ライン+ブレンドは1つのセグメントであるため、非常に短いセグメントに到達するために速度を落とす必要はありません。 おそらく、この設定に触れる必要はありません。

- ARC_BLEND_RAMP_FREQ = 20-これは、傾斜速度を使用するためのカットオフ周波数です。

この場合の傾斜速度は、セグメント全体で一定の加速を意味します。 加速度が最大化されていないため、これは台形の速度プロファイルよりも最適ではありません。 ただし、セグメントが十分に短い場合は、次のセグメントに到達する前に加速するのに十分な時間がありません。 前の例の短い線分を思い出してください。 ラインなのでコーナリング加速がないので、ご希望の速度まで自由に加速できます。 ただし、この線が 2 つの円弧の間にある場合は、次のセグメントの最大速度内に収まるように、再びすばやく減速する必要があります。 これは、加速のスパイク、次に減速のスパイクがあり、パフォーマンスの向上がほとんどないために大きなジャークが発生することを意味します。 この設定は、短いセグメントのこのジャークを排除する方法です。

基本的に、セグメントが 1 / ARC_BLEND_RAMP_FREQ 未満の時間で完了する場合、そのセグメントの台形の速度プロファイルを気にせず、一定の加速度を使用します。
（ARC_BLEND_RAMP_FREQ = 1000 に設定することは、サーボループが 1kHz の場合、常に台形加速度を使用することと同じです）。

台形プロファイルが到達する速度とランプを比較することにより、パフォーマンスの最悪の場合の損失を特徴付けることができます。

＃V_RIPPLE = A_MAX /（4.0 * F）＃ここで：＃V_RIPPLE =ランピングによる平均速度「損失」＃A_MAX =最大軸加速度

＃F = INI からのカットオフ周波数

前述のマシンの場合、20Hz のカットオフ周波数のリップルは 100 /（4 * 20）= 1.25IPS です。 これは高いように思われますが、これは最悪の場合の見積もりにすぎないことに注意してください。 実際には、台形のモーションプロファイルは、通常の加速度や要求された速度などの他の要因によって制限されるため、実際のパフォーマンスの低下ははるかに小さいはずです。 カットオフ周波数を上げると、パフォーマンスが低下する可能性がありますが、加速の不連続性のためにモーションが粗くなります。 20Hz から 200Hz の範囲の値から始めるのが妥当です。

最後に、コーナリングの加速によって制限されるため、微調整を行っても、小さくてタイトなコーナーがたくさんあるツールパスの速度は上がりません。

- SPINDLES = 3-サポートするスピンドルの数。 この番号は、モーションモジュールに渡される「NUM_SPINDLES」パラメータと一致する必要があります。
- COORDINATES = X YZ-制御されている軸の名前。 X、Y、Z、A、B、C、U、V、WARE のみが有効です。 COORDINATES で指定された軸のみが G コードで受け入れられます。 軸名を複数回書き込むことができます（例：ガントリーマシンの場合は X Y Y Z）。 一般的なトリブキンの運動学では、ジョイント番号はトリブキンのパラメータ COORDINATES =に従って順番に割り当てられます。 したがって、TRIVKINS 座標= XZ の場合、JOINT0 は X に対応し、JOINT1 は Z

に対応します。TRIVKINS およびその他のキネマティクスモジュールについては、キネマティクスのマニュアルページ（$ MAN KINS）を参照してください。

- LINEAR_UNITS = <UNITS>-直線軸の機械単位を指定します。 可能な選択肢は MM またはインチです。 これは NC コードの線形単位には影響しません（G20 および G21 ワードがこれを行います）。
- ANGULAR_UNITS = <UNITS>-回転軸の機械単位を指定します。 可能な選択肢は、度、度（円あたり 360）、ラジアン、ラジアン（円あたり 2PI）、グラデーション、またはゴン（円あたり 400）です。 これは、NC コードの角度単位には影響しません。 RS274NGC では、A-、B-、および C-の単語は常に度で表されます。
- DEFAULT_LINEAR_VELOCITY = 0.0167-直線軸のジョグの初期速度（マシン単位/秒）。 軸に表示される値は、1 分あたりのマシン単位に等しくなります。
- DEFAULT_LINEAR_ACCELERATION = 2.0-自明でない運動学を備えたマシンでは、「TELEOP」（デカルト空間）ジョグに使用される加速度（マシン単位/秒/秒）。
- MAX_LINEAR_VELOCITY = 5.0-任意の軸または協調移動の最大速度（マシン単位/秒）。 表示される値は、1 分あたり 300 ユニットに相当します。
- MAX_LINEAR_ACCELERATION = 20.0-任意の軸または調整された軸の移動の最大加速度（マシン単位/秒/秒）。
- POSITION_FILE = POSITION.TXT-空でない値に設定されている場合、ジョイント位置はこのファイルの実行間で保存されます。 これにより、マシンはシャットダウン時と同じ座標で起動できます。 これは、電源がオフのときにマシンが動かなかったことを前提としています。 設定されていない場合、ジョイント位置は保存されず、LINUXCNC が開始されるたびに 0 から始まります。 これは、ホームスイッチのない小型のマシンで役立ちます。 MESA リゾルバーインターフェースを使用する場合、このファイルを使用してアブソリュートエンコーダーをエミュレートし、ホーミングの必要性を排除できます（精度を損なうことなく）。 詳細については、HOSTMOT2 のマンページを参照してください。
- NO_FORCE_HOMING = 1-デフォルトの動作では、LINUXCNC は、MDI コマンドまたはプログラムが実行される前に、ユーザーにマシンのホームを強制します。 通常、ホーミングの前にジョギングのみが許可されます。 ID キネマティクスを使用する構成の場合、NO_FORCE_HOMING = 1 に設定すると、ユーザーは最初にマシンをホーミングせずに MDI を移動し、プログラムを実行できます。 ホーミング機能のないアイデンティティキネマティクスを使用するインターフェイスでは、このオプションを 1 に設定する必要があります。
- HOME = 0 0 0 0 0 0 0 0 0-ジョイントモードからテレオプモードに切り替えるときに KINEMATICSFORWARD（）を使用してワールド座標を計算するキネマティクスモジュールに必要なワールドホーム位置。 最大 9 つの座標値（X Y Z A B C U V W）を指定でき、未使用の末尾

項目は省略できます。 この値は、重要なキネマティクスを備えたマシンにのみ使用されます。 些細な運動学（ミル、旋盤、ガントリータイプ）を備えたマシンでは、この値は無視されます。 注：SIM ヘキサポッド構成では、Z 座標にゼロ以外の値が必要です。

警告

LINUXCNC は、NO_FORCE_HOMING = 1 を使用する場合、共同移動制限を認識しません。

6.6.2.10 [KINS]セクション

- JOINTS = 3-システム内のジョイント（モーター）の数を指定します。 たとえば、各軸に 1 つのモーターを備えた TRIVKINS XYZ マシンには、3 つのジョイントがあります。 2 つの軸のそれぞれに 1 つのモーターがあり、3 番目の軸に 2 つのモーターがあるガントリーマシンには、4 つのジョイントがあります。 （この構成変数は、モーションモジュール（MOTMOD）に指定されたジョイントの数（NUM_JOINTS）を設定するために GUI によって使用される場合があります）。 AXIS GUI、PNCCONF、および STEPCONF はこのアイテムを使用します。
- KINEMATICS = TRIVKINS-モーションモジュールのキネマティクスモジュールを指定します。 GUI はこの変数を使用して、MOTMOD モジュールの HAL ファイルの LOADRT 行を指定できます。 キネマティクスモジュールの詳細については、マンページを参照してください：$ MAN KINS

6.6.2.11 [AXIS_ <LETTER>]セクション

<LETTER>は、次のいずれかを指定します。XY Z A B C U V W

- MAX_VELOCITY = 1.2-この軸の最大速度（マシン単位/秒）。
- MAX_ACCELERATION = 20.0-この軸の最大加速度（機械単位/秒の 2 乗）。
- MIN_LIMIT = -1000-機械単位での軸モーションの最小制限（ソフト制限）。 この制限を超えると、コントローラーは軸の動きを中止します。 MIN_LIMIT を有効にする前に、軸をホームにする必要があります。 [AXIS_ <LETTER>]セクションのその軸に MIN_LIMIT がない無制限の回転を持つ回転軸（A、B、C TYP）の場合、-1E99 の値が使用されます。
- MAX_LIMIT = 1000-機械単位での軸モーションの最大制限（ソフト制限）。 この制限を超えると、コントローラーは軸の動きを中止します。 MAX_LIMIT を有効にする前に、軸をホームにする必要があります。 [AXIS_ <LETTER>]セクションでその軸の MAX_LIMIT がない無制限の回転を持つ回転軸（A、B、C TYP）の場合、1E99 の値が使用されます。
- WRAPPED_ROTARY = 1-これが ANGULAR 軸に対して 1 に設定されている場合、軸は 0～359.999 度移動します。 正の数は軸を正の方向に移動し、負の数は軸を負の方向に移動します。
- LOCKING_INDEXER_JOINT = 4-この値は、指定された軸<LETTER>のロッキングインデクサーに使用するジョイントを選択します。 この例では、ジョイントは 4 であり、これは、トリブキ

ン（アイデンティティ）キネマティクスを備えたXYZABシステムのB軸に対応します。 設定すると、この軸のG0移動により、ジョイントでロック解除が開始されます。4。ロック解除ピンは、ジョイントを待ちます。4。ロック解除ピンは、そのジョイントの高速でジョイントを移動します。 移動後、JOINT.4.UNLOCKはFALSEになり、モーションはJOINT.4.IS-UNLOCKEDがFALSEになるのを待ちます。 ロックされたロータリージョイントを移動する場合、他のジョイントと一緒に移動することはできません。 ロック解除ピンを作成するには、MOTMODパラメーターを使用します。

```
unlock_joints_mask=jointmask
The jointmask bits are: (LSB)0:joint0, 1:joint1, 2:joint2, ...
Example: loadrt motmod ... unlock_joints_mask=0x38
creates unlock pins for joints 3,4,5
```

- OFFSET_AV_RATIO = 0.1-ゼロ以外の場合、このアイテムは外部軸オフセットにHAL入力ピンの使用を有効にします。

'axis.<letter>.eoffset-enable'

'axis.<letter>.eoffset-counts'

'axis.<letter>.eoffset-scale'

使用法については、「外部軸オフセット」の章を参照してください。

6.6.2.12 [JOINT_ <NUM>]セクション

<NUM>は、ジョイント番号0を指定します。。。（NUM_JOINTS-1）NUM_JOINTSの値は、[KINS] JOINTS =によって設定されます。

[JOINT_0]、[JOINT_1]などのセクションには、ジョイント制御モジュールの個々のコンポーネントの一般的なパラメータが含まれています。 ジョイントセクション名は0から始まり、[KINS] JOINTSエントリで指定されたジョイントの数から1を引いた数まで続きます。

通常（TRIVKINSキネマティクスを使用するシステムの場合、ジョイントと軸座標文字の間には1：1の対応があります）：

- JOINT_0 = X
- JOINT_1 = Y
- JOINT_2 = Z
- JOINT_3 = A
- JOINT_4 = B
- JOINT_5 = C
- JOINT_6 = U

- JOINT_7 = V
- JOINT_8 = W

アイデンティティキネマティクスを備えた他のキネマティクスモジュールは、軸の半セットを使用した構成をサポートするために使用できます。 たとえば、COORDINATES = XZ の TRIVKINS を使用すると、ジョイント軸の関係は次のようになります。

- JOINT_0 = X
- JOINT_1 = Z

キネマティクスモジュールの詳細については、マンページを参照してください：$ MAN KINS

- TYPE = LINEAR-LINEAR または ANGULAR のいずれかのジョイントのタイプ。
- UNITS = INCH-指定されている場合、この設定は関連する[TRAJ] UNITS 設定を上書きします。（例：このジョイントのタイプが LINEAR の場合は[TRAJ] LINEAR_UNITS、このジョイントのタイプが ANGULAR の場合は[TRAJ] ANGULAR_UNITS）
- MAX_VELOCITY = 1.2-このジョイントの最大速度（マシン単位/秒）。
- MAX_ACCELERATION = 20.0-このジョイントの最大加速度（機械単位/秒の 2 乗）。
- BACKLASH = 0.0000-マシンユニットのバックラッシュ。 バックラッシュ補正値は、ジョイントの駆動に使用されるハードウェアの小さな欠陥を補うために使用できます。 ジョイントにバックラッシュが追加され、ステッパーを使用している場合は、STEPGEN_MAXACCEL をジョイントの MAX_ACCELERATION の 1.5〜2 倍に増やす必要があります。 過度のバックラッシュ補正により、方向が変わるときにジョイントがジャークする可能性があります。 COMP_FILE がジョイントに指定されている場合、BACKLASH は使用されません。
- COMP_FILE = FILE.EXTENSION-補正ファイルは、関節の位置情報のマップで構成されます。 補正ファイルの値はマシン単位です。 値の各セットは、スペースで区切られた 1 行にあります。 最初の値は公称値（指令位置）です。 2 番目と 3 番目の値は、COMP_FILE_TYPE の設定によって異なります。 公称値の間のポイントは、2 つの公称値の間で補間されます。 補正ファイルは、最小の名義で始まり、名義の最大値に昇順である必要があります。 ファイル名では大文字と小文字が区別され、文字や数字を含めることができます。 現在、LINUXCNC 内の制限は、ジョイントごとに 256 トリプレットです。

ジョイントに COMP_FILE が指定されている場合、BACKLASH は使用されません。
COMP_FILE_TYPE は、COMP_FILE ごとに指定する必要があります。

- COMP_FILE_TYPE = 0 または 1-補正ファイルのタイプを指定します。 最初の値は、両方のタイプの公称（コマンド）位置です。

- タイプ 0：2 番目の値は、ジョイントが正の方向に移動するときの実際の位置（増加値）を指定し、3 番目の値は、ジョイントが負の方向に移動するときの実際の位置（減少値）を指定します。

タイプ 0 の例

```
-1.000 -1.005 -0.995
0.000 0.002 -0.003
1.000 1.003 0.998
```

- タイプ 1：2 番目の値は、正の方向に移動するときの公称値からの正のオフセットを指定します。 3 番目の値は、負の方向に移動するときの公称値からの負のオフセットを指定します。

タイプ 1 の例

```
-1.000 0.005 -0.005
0.000 0.002 -0.003
1.000 0.003 -0.004
```

- MIN_LIMIT = -1000-機械単位での、関節運動の最小制限。 この制限に達すると、コントローラーは関節の動きを中止します。 [JOINT_N]セクションのジョイントに MIN_LIMIT がない、無制限の回転を持つロータリージョイントの場合、値-1E99 が使用されます。
- MAX_LIMIT = 1000-機械単位での、関節運動の最大制限。 この制限に達すると、コントローラーは関節の動きを中止します。 [JOINT_N]セクションのジョイントに MAX_LIMIT がない、無制限の回転を持つロータリージョイントの場合、値 1E99 が使用されます。

NOTE

ID キネマティクスの場合、[JOINT_N] MIN_LIMIT、MAX_LIMIT 設定は、対応する（1 対 1 の ID）[AXIS_L] 制限以上である必要があります。 これらの設定は、TRIVKINS キネマティクスモジュールが指定されている場合、起動時に検証されます。

NOTE

[JOINT_N] MIN_LIMIT、MAX_LIMIT 設定は、ホーミング前のジョイントモードでのジョギング中に適用されます。 原点復帰後、[AXIS_L] MIN_LIMIT、MAX_LIMIT 座標制限は、軸（座標文字）ジョギングの制約として、および GCODE 移動（プログラムおよび MDI コマンド）に使用される軌道計画によって使用されます。 軌道プランナーはデカルト空間（XYZABCUVW）で機能し、キネマティクスモジュールによって実装された関節の動きに関する情報はありません。 非アイデンティティキネマティクスが使用されている場合、軌道計画の位置制限に従う GCODE でジョイント制限違反が発生する可能性があります。 モーション

モジュールは、GCODE コマンドの実行中に発生した場合、常に関節位置制限違反と障害を検出します。 関連する GITHUB の問題＃97 も参照してください。

- MIN_FERROR = 0.010-これは、ジョイントが非常に低速でコマンド位置から逸脱することを許可されるマシン単位の値です。 MIN_FERROR が FERROR より小さい場合、2 つはエラートリップポイントのランプを生成します。 これは、一方の次元が速度で、もう一方の次元がエラーの後に許可されるグラフと考えることができます。 速度が上がると、後続のエラーの量も FERROR 値に向かって増加します。
- FERROR = 1.0-FERROR は、マシン単位での最大許容後続エラーです。 指令位置と検出位置の差がこの量を超えると、コントローラはサーボ計算を無効にし、すべての出力を 0.0 に設定し、増幅器を無効にします。 MIN_FERROR が.INI ファイルに存在する場合、速度に比例した次のエラーが使用されます。 ここで、最大許容追従誤差は速度に比例し、FERROR は[TRAJ] MAX_VELOCITY で設定された高速レートに適用され、低速の場合は比例して小さい追従誤差になります。 許容される最大の後続エラーは、常に MIN_FERROR より大きくなります。 これにより、静止軸の小さな追従エラーが誤ってモーションを中断するのを防ぎます。 振動などにより、常に小さな追従誤差が発生します。
- LOCKING_INDEXER = 1-ジョイントがロッキングインデクサーとして使用されていることを示します。

ホーミングこれらのパラメータはホーミングに関連しています。より良い説明については、ホーミング構成の章をお読みください。

- HOME = 0.0-ホーミングシーケンスの完了時にジョイントが移動する位置。
- HOME_OFFSET = 0.0-ホームスイッチまたはインデックスパルスのジョイント位置（機械単位）。 ホーミングプロセス中にホームポイントが見つかった場合、これはそのポイントに割り当てられた位置です。 ホームスイッチとリミットスイッチを共有し、ホーム/リミットスイッチをトグル状態のままにするホームシーケンスを使用する場合、ホームオフセットを使用して、ホーム位置を 0 にする場合は、ホームスイッチ位置を 0 以外に定義できます。
- HOME_SEARCH_VEL = 0.0-1 秒あたりの機械単位での初期原点復帰速度。 記号は進行方向を示します。 ゼロの値は、現在の場所がマシンのホームポジションであると想定することを意味します。 マシンにホームスイッチがない場合は、この値をゼロのままにしておきます。
- HOME_LATCH_VEL = 0.0-ホームスイッチのラッチ位置への 1 秒あたりの機械単位でのホーミング速度。 記号は進行方向を示します。
- HOME_FINAL_VEL = 0.0-ホームラッチ位置からホーム位置までの 1 秒あたりの機械単位の速度。 0 のままにするか、ジョイントに含まれない場合は、高速が使用されます。 正の数である必要があります。

- HOME_USE_INDEX = NO-このジョイントに使用されるエンコーダにインデックスパルスがあり、モーションカードにこの信号が用意されている場合は、YES に設定できます。 はいの場合、使用するホームパターンの種類に影響します。 現在、ベロシティモードと PID で STEPGEN を使用していない限り、ステッパーでインデックスを作成することはできません。

- HOME_IGNORE_LIMITS = NO-リミットスイッチをホームスイッチとして使用し、リミットスイッチを使用する場合は、これを YES に設定する必要があります。 YES に設定すると、このジョイントのリミットスイッチはホーミング時に無視されます。 ホームムーブの最後にホーム/リミットスイッチが切り替えられた状態にならないようにホーミングを構成する必要があります。ホームムーブ後にリミットスイッチエラーが発生します。

- HOME_IS_SHARED = <N>-ホーム入力が複数のジョイントで共有されている場合<N>を 1 に設定して、共有スイッチの 1 つがすでに閉じている場合にホーミングが開始されないようにします。 <N>を 0 に設定すると、スイッチが閉じている場合にホーミングが可能になります。

- HOME_ABSOLUTE_ENCODER = 0 | 1 | 2-ジョイントがアブソリュートエンコーダを使用していることを示すために使用されます。 ホーミングの要求時に、現在のジョイント値が HOME_OFFSET 値に設定されます。 HOME_ABSOLUTE_ENCODER 設定が 1 の場合、マシンは通常の最終移動を HOME 値に行います。 HOME_ABSOLUTE_ENCODER 設定が 2 の場合、最終的な移動は行われません。

- HOME_SEQUENCE = <N>-「HOMEALL」シーケンスを定義するために使用されます。 <N>は、0 または 1 または-1 から開始する必要があります。 追加のシーケンスは、（絶対値で）1 ずつ増加する数値で指定できます。 シーケンス番号のスキップは許可されていません。 HOME_SEQUENCE を省略すると、ジョイントは「すべてホーム」機能によってホームになりません。 複数のジョイントに同じシーケンス番号を指定することにより、複数のジョイントを同時にホームすることができます。 負のシーケンス番号は、その（負または正の）シーケンス番号を持つすべての関節の最終的な動きを延期するために使用されます。 詳細については、「ホームシーケンス」を参照してください。

- VOLATILE_HOME = 0-有効（1 に設定）の場合、マシンの電源がオフの場合、または非常停止がオンの場合、このジョイントはホームから外されます。 これは、マシンにホームスイッチがあり、ステップおよび方向駆動マシンなどの位置フィードバックがない場合に役立ちます。

サーボこれらのパラメータは、サーボによって制御されるジョイントに関連しています。

警告

以下は、サンプル INI ファイルまたはウィザードで生成されたファイルにあるカスタム INI ファイルエントリです。 これらは LINUXCNC ソフトウェアでは使用されません。 それらは、すべての設定を 1 か所にまとめるためだけにあります。 カスタム INI ファイルエントリの詳細については、「カスタムセクションと変数」サブセクションを参照してください。

次の項目は PID コンポーネントによって使用される可能性があり、出力はボルトであると想定されています。

- DEADBAND = 0.000015-機械単位で、モーターが所定の位置にあると見なすのに十分な距離。これは多くの場合、1、1.5、2、または 3 エンコーダカウントに相当する距離に設定されますが、厳密な規則はありません。 設定を緩く（大きく）すると、精度が低下しますが、サーボハンティングが少なくなります。 よりタイトな（より小さな）設定は、より多くのサーボハンティングを犠牲にしてより高い精度を試みます。 それがより不確実であるならば、それは本当により正確ですか？ 原則として、可能であればサーボハンティングを回避するか、少なくとも制限することをお勧めします。

サーボが満足できる場所がない状態を作り出す可能性があるため、1 エンコーダカウントを下回ることには注意してください。 これは、狩猟（遅い）から神経質（速い）、さらには不適切なチューニングによって引き起こされる振動と混同されやすい鳴き声にまで及ぶ可能性があります。 少なくともグロスチューニングを終えるまでは、最初はここで 1、2 カウント緩めたほうがよいでしょう。

DEADBAND 値の決定に使用するエンコーダパルスごとのマシンユニットの計算例：

$$\frac{1\,revolution}{1000\,lines} X \frac{1\,line}{4\,pulse/line} X \frac{0.2\,units}{1\,revolution} = \frac{0.200\,units}{4000\,pulses} = \frac{0.00005\,units}{1\,pulse}$$

- BIAS = 0.000-これは HM2-SERVO などで使用されます。 バイアスは、出力に追加される一定量です。 ほとんどの場合、ゼロのままにしておく必要があります。 ただし、サーボアンプのオフセットを補正したり、垂直方向に移動するオブジェクトの重量のバランスをとったりすると便利な場合があります。 PID ループが無効になると、他のすべてのコンポーネントと同様に、バイアスがオフになります。 出力。

- P = 50-ジョイントサーボの比例ゲイン。 この値は、機械単位での指令位置と実際の位置の間の誤差を乗算し、モーターアンプの計算された電圧に寄与します。 P ゲインの単位は、マシン単位あたりのボルトです。例：$\frac{volts}{unit}$

- I = 0-ジョイントサーボの積分ゲイン。 この値は、機械ユニットのコマンド位置と実際の位置の間の累積誤差を乗算し、モーターアンプの計算された電圧に寄与します。 I ゲインの単位は、マシン単位秒あたりのボルトです。例：$\frac{volts}{unit\,second}$

- D = 0-ジョイントサーボの微分ゲイン。 この値は、現在のエラーと以前のエラーの差を乗算し、モーターアンプの計算された電圧に寄与します。 D ゲインの単位は、1 秒あたりのマシン単位あたりのボルトです。$\frac{volts}{unit\,second}$

- FF2 = 0-2 次フィードフォワードゲイン。 この数値に 1 秒あたり 1 秒あたりの指令位置の変化を掛けると、モーターアンプの計算電圧に寄与します。 FF2 ゲインの単位は、マシン単位/秒/秒あたりのボルトです。例：$\frac{volts}{unit\ second^2}$

- OUTPUT_SCALE = 1.000-

- OUTPUT_OFFSET = 0.000-これらの 2 つの値は、モーターアンプへのジョイント出力のスケール係数とオフセット係数です。 2 番目の値（オフセット）は、計算された出力（ボルト単位）から減算され、最初の値（スケール係数）で除算されてから、D / A コンバーターに書き込まれます。 スケール値の単位は、DAC 出力ボルトあたりの真のボルトです。 オフセット値の単位はボルトです。 これらは、DAC を線形化するために使用できます。 具体的には、出力を書き込むとき、LINUXCNC は最初に、準 SI 単位の目的の出力を、アンプ DAC のボルトなどの生のアクチュエータ値に変換します。 このスケーリングは次のようになります。

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

スケールの値は、単位分析を行うことで分析的に取得できます。つまり、単位は[出力 SI 単位] / [アクチュエータ単位]です。 たとえば、1 ボルトが 250 MM /秒の速度になるような速度モード増幅器を備えたマシンでは。

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

オフセットの単位は、MM / SEC などの機械単位であり、センサーの読み取り値から事前に差し引かれていることに注意してください。 このオフセットの値は、アクチュエータ出力に対して 0.0 を生成する出力の値を見つけることによって取得されます。 DAC が線形化されている場合、このオフセットは通常 0.0 です。

スケールとオフセットを使用して DAC を線形化することもでき、アンプゲイン、DAC 非線形性、DAC ユニットなどの複合効果を反映する値が得られます。これを行うには、次の手順に従います。

1. 出力のキャリブレーションテーブルを作成し、DAC を目的の電圧で駆動して、結果を測定します。

2. 最小二乗線形フィットを実行して、次のような係数 A、B を取得します。
   $measured = a * raw + b$

3. 測定結果がコマンド出力と同じになるように生の出力が必要であることに注意してください。 これの意味は

   (ア) $command = a * raw + b$

   (イ) $raw = (command - b) / a$

4. その結果、線形フィットからの A 係数と B 係数を、コントローラーのスケールとオフセットとして直接使用できます。

電圧測定の例については、次の表を参照してください。

Table 8.1: Output Voltage Measurements

Raw	Measured
-10	-9.93
-9	-8.83
0	-0.03
1	0.96
9	9.87
10	10.87

- MAX_OUTPUT = 10-モーターアンプに書き込まれる PID 補正の出力の最大値（ボルト単位）。 計算された出力値は、この制限に固定されます。 制限は、生の出力単位にスケーリングする前に適用されます。 値はプラス側とマイナス側の両方に対称的に適用されます。
- INPUT_SCALE = 20000-サンプル構成内
- ENCODER_SCALE = 20000-PNCCONF で構築された構成[TRAJ]セクションで設定された 1 台のマシンユニットの移動に対応するパルス数を指定します。 リニアジョイントの場合、1 つのマシンユニットは LINEAR_UNITS の設定と同じになります。 角度ジョイントの場合、1 単位は ANGULAR_UNITS の設定と同じです。 2 番目の数値は、指定されている場合は無視されます。 たとえば、1 回転エンコーダあたり 2000 カウント、10 回転/インチのギアリング、および必要なインチ単位の場合、次のようになります。

$$input\,scale = 2000\frac{counts}{rev} * 10\frac{rev}{inch} = 20000\frac{counts}{inch}$$

ステッパーこれらのパラメーターは、ステッパーによって制御されるジョイントに関連しています。

警告

以下は、サンプル INI ファイルまたはウィザードで生成されたファイルにあるカスタム INI ファイルエントリです。 これらは LINUXCNC ソフトウェアでは使用されません。 それらは、すべての設定を 1 か所にまとめるためだけにあります。 カスタム INI ファイルエントリの詳細については、「カスタムセクションと変数」サブセクションを参照してください。

次のアイテムは、STEPGEN コンポーネントによって使用される可能性があります。

- SCALE = 4000 - IN SAMPLE CONFIGS
- STEP_SCALE = 4000-PNCCONF で構築された構成[TRAJ]セクションで設定された 1 台のマシンユニットの移動に対応するパルス数を指定します。 ステッパーシステムの場合、これはマシンユニットごとに発行されるステップパルスの数です。 リニアジョイントの場合、1 つのマシ

ンユニットは LINEAR_UNITS の設定と同じになります。 角度ジョイントの場合、1 単位は ANGULAR_UNITS の設定と同じです。 サーボシステムの場合、これはマシンユニットあたりのフィードバックパルスの数です。 2 番目の数値は、指定されている場合は無視されます。

たとえば、ハーフステッピング、10 回転/インチのギアリング、およびインチの必要な機械単位を備えた 1.8 度のステッピングモーターでは、次のようになります。

$$input\,scale = \frac{2\,steps}{1.8\,degrees} * 360\,\frac{degree}{rev} * 10\,\frac{rev}{inch} = 4000\,\frac{steps}{inch}$$

- ENCODER_SCALE = 20000（オプションで PNCCONF ビルド構成で使用）-[TRAJ]セクションで設定された 1 台のマシンユニットの移動に対応するパルス数を指定します。 リニアジョイントの場合、1 つのマシンユニットは LINEAR_UNITS の設定と同じになります。 角度ジョイントの場合、1 単位は ANGULAR_UNITS の設定と同じです。 2 番目の数値は、指定されている場合は無視されます。 たとえば、1 回転エンコーダあたり 2000 カウント、10 回転/インチのギアリング、および必要なインチ単位の場合、次のようになります。

$$input\,scale = 2000\,\frac{counts}{rev} * 10\,\frac{rev}{inch} = 20000\,\frac{counts}{inch}$$

- STEPGEN_MAXACCEL = 21.0-ステップジェネレータの加速制限。 これは、ジョイントの MAX_ACCELERATION よりも 1%から 10%大きくする必要があります。 この値は、STEPGEN の「位置ループ」の調整を改善します。 ジョイントにバックラッシュ補正を追加した場合、これは MAX_ACCELERATION の 1.5〜2 倍になるはずです。
- STEPGEN_MAXVEL = 1.4-古い構成ファイルには、ステップジェネレーターの速度制限もあります。 指定する場合は、ジョイントの MAX_VELOCITY よりも 1%から 10%大きくする必要があります。 その後のテストでは、STEPGEN_MAXVEL を使用しても STEPGEN の位置ループの調整は改善されないことが示されています。

6.6.2.13 [EMCIO]セクション

- EMCIO = IO-IO コントローラープログラムの名前
- CYCLE_TIME = 0.100-EMCIO が実行される期間（秒単位）。 0.0 または負の数にすると、EMCIO はまったくスリープしないようになります。 通常、この番号を変更する必要はありません。
- TOOL_TABLE = TOOL.TBL-ユーザーマニュアルで説明されているツール情報を含むファイル。
- TOOL_CHANGE_POSITION = 0 0 2- 3 桁が使用されている場合に、工具交換を実行するときに移動する XYZ 位置を指定します。 6 桁を使用する場合の XYZABC の場所を指定します。 9 桁を使用する場合の XYZABCUVW の場所を指定します。 ツールの変更は組み合わせることが

できます。 たとえば、羽ペンと位置の変更を組み合わせると、最初に Z を移動し、次に X と Y を移動できます。

- TOOL_CHANGE_WITH_SPINDLE_ON = 1-値が 1 の場合、工具交換中、スピンドルはオンのままになります。材料が工具ではなくスピンドルにある旋盤または機械に役立ちます。
- TOOL_CHANGE_QUILL_UP = 1-値が 1 の場合、工具交換の前に Z 軸がマシン 0 に移動します。これは、G0 G53Z0 を発行するのと同じです。
- TOOL_CHANGE_AT_G30 = 1-値が 1 の場合、マシンは G30 のパラメータ 5181-5186 で定義された参照ポイントに移動します。詳細については、パラメータセクションおよび<< GCODE：G30-G30.1 を参照してください。
- RANDOM_TOOLCHANGER = 1-これは、ツールを元のポケットに戻すことができないマシン用です。 たとえば、アクティブポケットの工具をスピンドルの工具と交換する機械。

6.7 原点復帰構成

6.7.1 概要

ホーミングは、G53 マシン座標のゼロ原点を設定します。

ソフト制限は、マシンの原点を基準にして定義されます。

ソフト制限は、制限スイッチに達する前に軸を自動的に減速および停止します。適切にセットアップおよび機能しているマシンは、ソフト（ウェア）制限を超えて移動せず、ホームスイッチ/インデックスメカニズムと同じようにマシンの原点を繰り返し設定できます。

LINUXCNC は、目（アライメントマーク）、スイッチ、スイッチとエンコーダインデックス、またはアブソリュートエンコーダを使用してホームに設定できます。 ホーミングは非常に簡単に思えます。各ジョイントを既知の場所に移動し、それに応じて LINUXCNC の内部変数を設定するだけです。 ただし、マシンごとに要件が異なり、ホーミングは実際には非常に複雑です。

NOTE

ホーミングスイッチ/ホームプロシージャまたはリミットスイッチなしで LINUXCNC を使用することは可能ですが、ソフトリミットの追加のセキュリティを無効にします。

6.7.2 前提条件

ホーミングは、いくつかの基本的なマシンの仮定に依存しています。

- 負の方向と正の方向は工具の動きに基づいており、実際の機械の動きとは異なる場合があります。つまり、ミルでは通常、ツールではなくテーブルが移動します。

- すべてが G53 マシンのゼロ原点から参照され、原点はどこにでも配置できます（移動できる場所の外でも）
- G53 マシンのゼロ原点は通常、ソフト制限領域内にありますが、必ずしもそうとは限りません。
- 原点復帰スイッチのオフセットは原点がどこにあるかを設定しますが、それでも原点から参照されます。
- エンコーダインデックスホーミングを使用する場合、ホームスイッチのオフセットは、ホームスイッチが作動した後のエンコーダ基準位置から計算されます。
- 負のソフトウェア（ウェア）制限は、ホーミング後に負の方向に移動できる最大の制限です。（しかし、絶対的な意味で否定的ではないかもしれません）
- 正のソフトウェア（ウェア）制限は、ホーミング後に正の方向に移動できる最大の制限です。（ただし、通常は正の数として設定されますが、絶対的な意味では正ではない場合があります）
- ソフトウェア）リミットはリミットスイッチエリア内にあります。
- （最終）ホームポジションがソフトリミットエリア内にある
- （スイッチベースのホーミングを使用する場合）ホーミングスイッチは、リミットスイッチ（共有ホーム/リミットスイッチ）を使用するか、別のホームスイッチを使用する場合はリミットスイッチエリア内にあります。
- 別のホーミングスイッチを使用している場合、ホームスイッチの反対側でホーミングを開始する可能性があります。これを HOME_IGNORE_LIMITS オプションと組み合わせると、ハードクラッシュが発生します。 これを回避するには、ホームスイッチを、トリップドッグが特定の側にあるときの状態を、トリップポイントを再び通過するまで切り替えるようにします。

別の言い方をすれば、ホームスイッチの状態は、スイッチに対するドックの位置を表す必要があります（つまり、スイッチの前または後）。

ドックが同じ方向にスイッチを越えて惰性走行したとしても、そのようにとどまらなければなりません。

NOTE

ソフトマシンの制限外の G53 マシンオリジンで LINUXCNC を使用することは可能ですが、パラメーターを設定せずに G28 または G30 を使用すると、デフォルトでオリジンに移動します。 これにより、位置に到達する前にリミットスイッチが作動します。

6.7.3 個別のホームスイッチのレイアウト例

この例は、個別のホームスイッチを備えた最小および最大リミットスイッチを示しています。

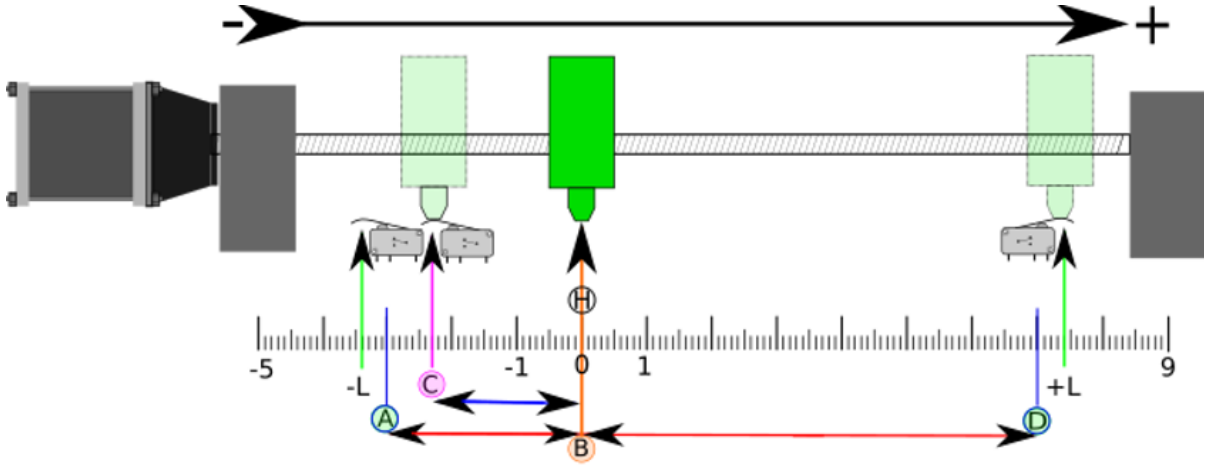


図 8-66

- A は負のソフト制限です
- B は G53 マシンの座標原点です
- C はホームスイッチのトリップポイントです
- D は正のソフトリミットです
- H は最終的なホームポジション（HOME）= 0 ユニットです
- -L と+ L は、リミットスイッチのトリップポイントです。
- A <-> B は負のソフト制限（MIN_LIMITS）=-3 単位です
- B <-> C は HOME_OFFSET（HOME_OFFSET）=-2.3 単位です
- B <-> D は正のソフト制限（MAX_LIMITS）= 7 単位です
- A <-> D は総移動量= 10 ユニットです
- この例では、リミットスイッチとソフトリミット（-L <-> A および D <-+ L）の間の距離が拡大されています。
- アンプが無効になった後の惰行のために、リミットスイッチと実際の物理的なハードコンタクトとの間に距離があることに注意してください。

NOTE

ホーミングは G53 座標系を設定しますが、機械の原点（ゼロ点）はどこでもかまいませんが、ゼロ点を負のソフト制限に設定すると、すべての G53 座標が正になります。これはおそらく覚えやすいでしょう。これを行うには、MIN_LIMIT = 0 を設定し、MAX_LIMIT が正であることを確認します。

6.7.4 共有制限/ホームスイッチのレイアウト例

この例は、最大リミットスイッチと組み合わせた最小リミット/ホームスイッチを示しています。

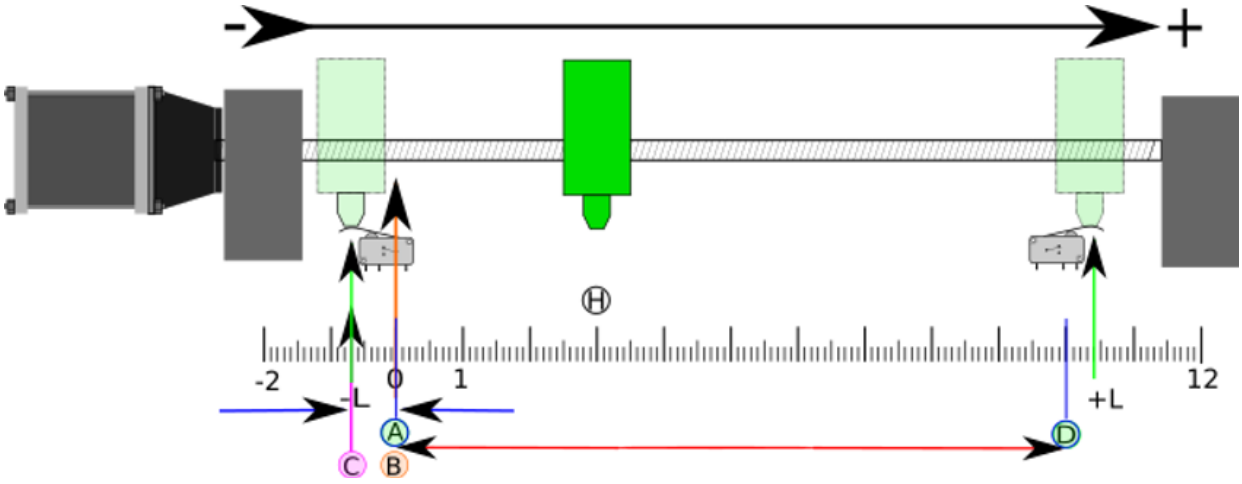


図 8-67

- A は負のソフト制限です
- B は G53 マシンの座標原点です
- C は、（-L）最小制限トリップと共有されるホームスイッチトリップポイントです。
- D は正のソフトリミットです
- H は最終的なホームポジション（HOME）= 3 ユニット
- -L と+ L はリミットスイッチのトリップポイントです
- A <-> B は負のソフト制限（MIN_LIMITS）= 0 単位です
- B <-> C は HOME_OFFSET（HOME_OFFSET）=-0.7 単位です
- B <-> D は正のソフト制限（MAX_LIMITS）10 単位です
- A <-> D は総移動量= 10 ユニットです
- この例では、制限スイッチとソフト制限（-L <-> A および D <-> + L）の間の距離が拡大されています。
- アンプが無効になった後の惰行のために、リミットスイッチと実際の物理的なハードコンタクトとの間に距離があることに注意してください。

6.7.5 ホーミングシーケンス

次の表に示すように、HOME_SEARCH_VEL および HOME_LATCH_VEL の符号によって定義される、関連する構成パラメーターとともに、4 つの可能なホーミングシーケンスがあります。 2 つの基本的な条件が存在します。HOME_SEARCH_VEL と HOME_LATCH_VEL は同じ符号であるか、反対の符号です。各構成パラメーターの機能の詳細については、次のセクションを参照してください。

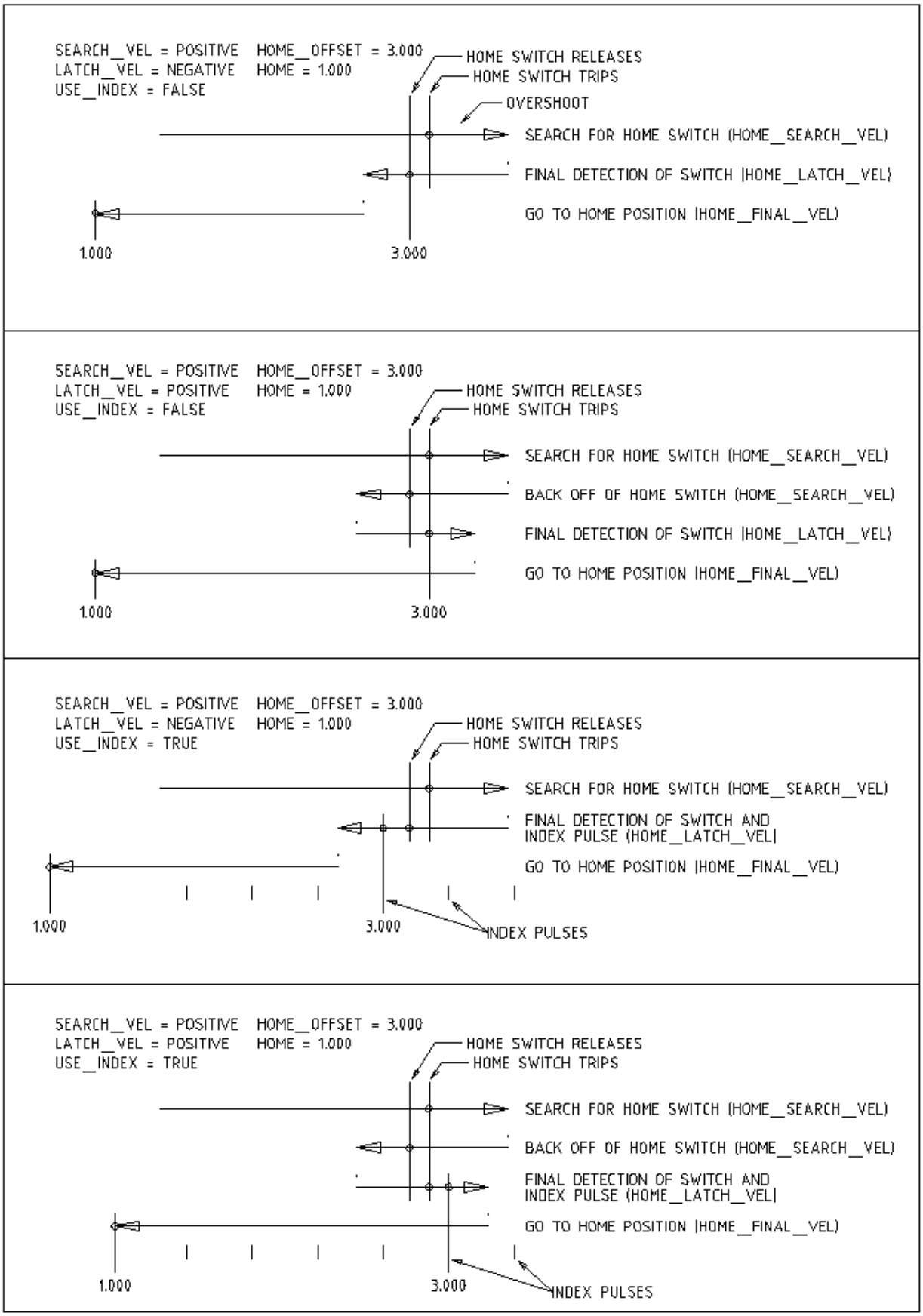

SEARCH_VEL = POSITIVE
LATCH_VEL = NEGATIVE
USE_INDEX = FALSE
HOME_OFFSET = 3.000
HOME = 1.000
HOME SWITCH RELEASES
HOME SWITCH TRIPS
OVERSHOOT
SEARCH FOR HOME SWITCH (HOME_SEARCH_VEL)
FINAL DETECTION OF SWITCH (HOME_LATCH_VEL)
GO TO HOME POSITION (HOME_FINAL_VEL)
1.000
3.000
SEARCH_VEL = POSITIVE
LATCH_VEL = POSITIVE
USE_INDEX = FALSE
HOME_OFFSET = 3.000
HOME = 1.000
HOME SWITCH RELEASES
HOME SWITCH TRIPS
SEARCH FOR HOME SWITCH (HOME_SEARCH_VEL)
BACK OFF OF HOME SWITCH (HOME_SEARCH_VEL)
FINAL DETECTION OF SWITCH (HOME_LATCH_VEL)
GO TO HOME POSITION (HOME_FINAL_VEL)
1.000
3.000
SEARCH_VEL = POSITIVE
LATCH_VEL = NEGATIVE
USE_INDEX = TRUE
HOME_OFFSET = 3.000
HOME = 1.000
HOME SWITCH RELEASES
HOME SWITCH TRIPS
SEARCH FOR HOME SWITCH (HOME_SEARCH_VEL)
FINAL DETECTION OF SWITCH AND
INDEX PULSE (HOME_LATCH_VEL)
GO TO HOME POSITION (HOME_FINAL_VEL)
1.000
3.000
INDEX PULSES
SEARCH_VEL = POSITIVE
LATCH_VEL = POSITIVE
USE_INDEX = TRUE
HOME_OFFSET = 3.000
HOME = 1.000
HOME SWITCH RELEASES
HOME SWITCH TRIPS
SEARCH FOR HOME SWITCH (HOME_SEARCH_VEL)
BACK OFF OF HOME SWITCH (HOME_SEARCH_VEL)
FINAL DETECTION OF SWITCH AND
INDEX PULSE (HOME_LATCH_VEL)
GO TO HOME POSITION (HOME_FINAL_VEL)
1.000
3.000
INDEX PULSES

図 8-68

6.7.6 構成

以下は、ホームシーケンスがどのように動作するかを正確に決定します。 これらは、INIFILE の[JOINT_N] セクションで定義されています。

Homing Type	HOME_SEARCH_VEL	HOME_LATCH_VEL	HOME_USE_INDEX
Immediate	0	0	NO
Index-only	0	nonzero	YES
Switch-only	nonzero	nonzero	NO
Switch and Index	nonzero	nonzero	YES

NOTE

他の組み合わせはエラーになる可能性があります。

1.1.1.1 HOME_SEARCH_VEL

この変数には、1 秒あたりのマシンユニットの単位があります。

デフォルト値はゼロです。 値がゼロの場合、LINUXCNC はホームスイッチがないと見なします。 ホーミングの検索段階はスキップされます。

HOME_SEARCH_VEL がゼロ以外の場合、LINUXCNC はホームスイッチがあると見なします。 まず、ホームスイッチがすでに作動しているかどうかを確認します。 トリップした場合は、HOME_SEARCH_VEL のスイッチをオフに戻します。 バックオフの方向は、HOME_SEARCH_VEL の符号と反対です。 次に、HOME_SEARCH_VEL の符号で指定された方向に、絶対値で決定された速度で移動して、ホームスイッチを検索します。 ホームスイッチが検出されると、ジョイントは可能な限り速く停止しますが、常にオーバーシュートが発生します。 オーバーシュートの量は速度によって異なります。 高すぎると、ジョイントがオーバーシュートしてリミットスイッチにぶつかったり、移動の終わりに衝突したりする可能性があります。 一方、HOME_SEARCH_VEL が低すぎると、ホーミングに時間がかかる場合があります。

6.7.6.1 HOME_LATCH_VEL

この変数には、1 秒あたりのマシンユニットの単位があります。

LINUXCNC がホームスイッチ（存在する場合）とインデックスパルス位置（存在する場合）を最終的に正確に決定するときに使用する速度と方向を指定します。 精度を最大化するために、通常は検索速度よりも遅くなります。 HOME_SEARCH_VEL と HOME_LATCH_VEL の符号が同じ場合、検索フェーズと同じ方向に移動しながらラッチフェーズが実行されます。 （その場合、LINUXCNC は最初にスイッチをバックオフしてから、ラッチ速度で再びスイッチに向かって移動します。）HOME_SEARCH_VEL と HOME_LATCH_VEL の符号が反対の場合、ラッチフェーズは検索フェーズから反対方向に移動しながら実行されます。 つまり、LINUXCNC は、スイッチをオフにした後、最初のパルスをラッチします。 HOME_SEARCH_VEL がゼロ（ホームスイッチがないことを意味する）で、このパラメーターがゼロ以外の場合、LINUXCNC はインデックスパルス検索に進

みます。 HOME_SEARCH_VEL がゼロ以外で、このパラメーターがゼロの場合、それはエラーであり、ホーミング操作は失敗します。 デフォルト値はゼロです。

6.7.6.2 HOME_FINAL_VEL

この変数には、1 秒あたりのマシンユニットの単位があります。

LINUXCNC が HOME_OFFSET から HOME 位置に移動するときに使用する速度を指定します。 HOME_FINAL_VEL が INI ファイルにない場合は、最大ジョイント速度を使用してこの移動を行います。 値は正の数でなければなりません。

6.7.6.3 HOME_IGNORE_LIMITS

YES / NO の値を保持できます。 このパラメーターのデフォルト値は NO です。 このフラグは、ホーミング中に LINUXCNC がこのジョイントのリミットスイッチ入力を無視するかどうかを決定します。 この設定は、他のジョイントの制限入力を無視しません。 別のホームスイッチがない場合は、これを YES に設定し、リミットスイッチ信号を HAL のジョイントホームスイッチ入力に接続します。 LINUXCNC は、ホーミング中にこのジョイントのリミットスイッチ入力を無視します。 すべてのホーミングと制限に 1 つの入力のみを使用するには、HAL でホーミングしていないジョイントの制限信号をブロックし、一度に 1 つのジョイントをホームにする必要があります。

6.7.6.4 HOME_USE_INDEX

インデックスパルスがあるかどうかを指定します。 フラグが TRUE（HOME_USE_INDEX = YES）の場合、LINUXCNC はインデックスパルスの立ち上がりエッジでラッチします。 FALSE の場合、LINUXCNC はホームスイッチの立ち上がりエッジまたは立ち下がりエッジのいずれかでラッチします（HOME_SEARCH_VEL および HOME_LATCH_VEL の符号に応じて）。 デフォルト値は NO です。

NOTE

HOME_USE_INDEX では、HAL ファイル内で ENCODER.N.INDEX-ENABLE から JOINT.N.INDEX-ENABLE への接続が必要です。

6.7.6.5 HOME_OFFSET

これは、G53 マシン座標系の原点ゼロ点の位置を定義します。

これは、機械の原点からホームスイッチのトリップポイントまたはインデックスパルスまでの、ジョイント単位での距離（オフセット）です。

LINUXCNC は、スイッチのトリップポイント/インデックスパルスを検出した後、ジョイント座標位置を HOME_OFFSET に設定し、ソフトリミットが参照する原点を定義します。

デフォルト値はゼロです。

NOTE

HOME_OFFSET 変数で示されるように、ホームスイッチの位置は、ソフト制限の内側または外側にすることができます。 それらは、ハードリミットスイッチと共有されるか、ハードリミットスイッチ内で共有されます。

6.7.6.6 HOME

ホーミングシーケンスの完了時にジョイントが移動する位置。 ホームスイッチまたはホームスイッチを検出し、パルスにインデックスを付け（構成に応じて）、そのポイントの座標を HOME_OFFSET に設定した後、LINUXCNC はホーミングプロセスの最終ステップとして HOME に移動します。 デフォルト値はゼロです。 このパラメータが HOME_OFFSET と同じであっても、ジョイントは停止時にラッチ位置をわずかにオーバーシュートすることに注意してください。 したがって、この時点では常に小さな動きがあります（HOME_SEARCH_VEL がゼロで、検索/ラッチステージ全体がスキップされた場合を除く）。 この最終的な移動は、HOME_FINAL_VEL が設定されていない限り、ジョイントの最大速度で行われます。

NOTE

HOME_OFFSET と HOME の違いは、HOME_OFFSET は、最初に HOME_OFFSET 値をホームが見つかった場所に適用することによってマシン上の原点の場所とスケールを確立し、次に HOME がそのスケールでジョイントを移動する場所を指定することです。

6.7.6.7 HOME_IS_SHARED

このジョイントに個別のホームスイッチ入力がなく、同じピンに多数のモーメンタリスイッチが配線されている場合は、この値を 1 に設定して、共有スイッチの 1 つがすでに閉じている場合にホーミングが開始されないようにします。 この値を 0 に設定すると、スイッチがすでに閉じている場合でもホーミングが可能になります。

6.7.6.8 HOME_ABSOLUTE_ENCODER

アブソリュートエンコーダに使用します。 ジョイントのホーム化が要求されると、現在のジョイント位置が[JOINT_N] HOME_OFFSET 値に設定されます。

[JOINT_N] HOME 位置への最後の移動は、HOME_ABSOLUTE_ENCODER 設定に従ってオプションです。

```
HOME_ABSOLUTE_ENCODER = 0 (Default) joint does not use an absolute encoder
HOME_ABSOLUTE_ENCODER = 1 Absolute encoder, final move to [JOINT_n]HOME
HOME_ABSOLUTE_ENCODER = 2 Absolute encoder, NO final move to [JOINT_n]HOME
```

NOTE

HOME_IS_SHARED 設定は黙って無視されます。

NOTE

ジョイントをホームに戻す要求は、黙って無視されます。

6.7.6.9 HOME_SEQUENCE

多関節ホーミングシーケンス HOMEALL を定義し、ホーミング順序を適用するために使用されます（たとえば、X がまだホーミングされていない場合、Z はホーミングされない可能性があります）。 低い（絶対値）HOME_SEQUENCE のすべてのジョイントがすでにホームに戻され、HOME_OFFSET にあると、ジョイントをホームに戻すことができます。 2 つのジョイントが同じ HOME_SEQUENCE を持っている場合、それらは同時にホームされる可能性があります。

NOTE

HOME_SEQUENCE が指定されていない場合、ジョイントは HOME ALL シーケンスによってホーミングされません（ただし、個々のジョイント固有のホーミングコマンドによってホーミングされる場合があります）。

最初の HOME_SEQUENCE 番号は、0、1（または-1）の場合があります。 シーケンス番号の絶対値は 1 ずつインクリメントする必要があります—シーケンス番号のスキップはサポートされていません。 シーケンス番号を省略した場合、最後の有効なシーケンス番号が完了すると、HOMEALL ホーミングが停止します。

負の HOME_SEQUENCE 値は、シーケンス内のすべてのジョイントの準備ができるまで待機することにより、シーケンス内のジョイントが[JOINT_N] HOME への最終移動を同期する必要があることを示します。 いずれかのジョイントの HOME_SEQUENCE 値が負の場合、HOME_SEQUENCE アイテム値の絶対値が同じ（正または負）のすべてのジョイントが最終移動を同期します。

負の HOME_SEQUENCE は、単一のジョイントをホームするコマンドにも適用されます。 HOME_SEQUENCE 値が負の場合、その HOME_SEQUENCE の絶対値が同じであるすべてのジョイントは、同期された最終移動とともにホームになります。 HOME_SEQUENCE 値がゼロまたは正の場合、ジョイントをホームするコマンドは、指定されたジョイントのみをホームします。

HOME_SEQUENCE が負のジョイントのジョイントモードジョギングは許可されていません。 一般的なガントリーアプリケーションでは、このようなジョギングはミスアライメント（ラッキン

グ）につながる可能性があります。 マシンがホームに戻ると、ワールド座標での従来のジョギングが常に利用可能になることに注意してください。

3 関節システムの例

2 つのシーケンス（0,1）、同期なし

```
[JOINT_0]HOME_SEQUENCE = 0
[JOINT_1]HOME_SEQUENCE = 1
[JOINT_2]HOME_SEQUENCE = 1
```

2 つのシーケンス、ジョイント 1 と 2 が同期

```
[JOINT_0]HOME_SEQUENCE = 0
[JOINT_1]HOME_SEQUENCE = -1
[JOINT_2]HOME_SEQUENCE = -1
```

正と負の値が混在している場合、ジョイント 1 と 2 は同期します

```
[JOINT_0]HOME_SEQUENCE = 0
[JOINT_1]HOME_SEQUENCE = -1
[JOINT_2]HOME_SEQUENCE = 1
```

1 つのシーケンス、同期なし

```
[JOINT_0]HOME_SEQUENCE = 0
[JOINT_1]HOME_SEQUENCE = 0
[JOINT_2]HOME_SEQUENCE = 0
```

1 つのシーケンス、すべての関節が同期

```
[JOINT_0]HOME_SEQUENCE = -1
[JOINT_1]HOME_SEQUENCE = -1
[JOINT_2]HOME_SEQUENCE = -1
```

6.7.6.10 VOLATILE_HOME

この設定が TRUE の場合、マシンがオフ状態に移行するたびに、このジョイントはホームから外れます。 これは、ジョイントドライブがオフのときに位置を維持しないジョイントに適しています。 一部のステッピングドライブ、特にマイクロステップドライブでは、これが必要になる場合があります。

6.7.6.11 LOCKING_INDEXER

このジョイントがロッキングロータリーインデクサーの場合、ホーミング前にロックが解除され、その後ロックされます。

6.7.6.12 即時ホーミング

ジョイントにホームスイッチがない場合、またはロータリージョイントのように論理的なホーム位置がなく、AXIS GUI で[すべてホーム]ボタンが押されたときにそのジョイントを現在の位置にホームにする場合は、次の INI エントリを使用します。 その関節のために必要です。

1. HOME_SEARCH_VEL = 0
2. HOME_LATCH_VEL = 0
3. HOME_USE_INDEX = NO
4. HOME EQUALS TO HOME_OFFSET
5. HOME_SEQUENCE = 0 (OR OTHER VALID SEQUENCE NUMBER)

NOTE

指定されていない HOME_SEARCH_VEL、HOME_LATCH_VEL、HOME_USE_INDEX、HOME、および HOME_OFFSET のデフォルト値はゼロであるため、即時ホーミングを要求する場合は省略できます。 HOME_SEQUENCE を省略すると、上記のように HOME ALL 動作からジョイントが削除されるため、通常は有効な HOME_SEQUENCE 番号を含める必要があります。

6.7.6.13 ホーミングの抑制

ハルピン（MOTION.HOMING-INHIBIT）は、「HOMEALL」と個々のジョイントホーミングの両方のホーミング開始を禁止するために提供されています。

一部のシステムは、負の[JOINT_N] HOME_SEQUENCE = INI ファイルアイテムによって制御される最終的なジョイントホーミング移動を同期するための規定を利用します。 デフォルトでは、同期の規定により、マシンの位置がずれる可能性のあるジョイントジョグ（ガントリーラックなど）を防ぐために、ホーミング前のジョイントジョギングが許可されていません。

システムインテグレーターは、[JOINT_N] HOME_SEQUENCE アイテムを切り替える HAL ロジックを使用して、ホーミング前にジョイントジョギングを許可できます。 このロジックは、MOTION.HOMING-INHIBIT ピンもアサートして、ジョイントジョギングが有効になっているときに誤ってホーミングが開始されないようにする必要があります。

例：ホーミングの前に個々のジョイントジョギングの正のシーケンス（1）を選択するスイッチ（ALLOW_JJOG）を使用して同期ホーミングに負のシーケンス（-1）を使用する同期ジョイント 0,1（部分的な HAL コード）：

```
loadrt mux2 names=home_sequence_mux
```

```
loadrt conv_float_s32 names=home_sequence_s32
setp home_sequence_mux.in0 -1
setp home_sequence_mux.in1 1
addf home_sequence_mux servo-thread
addf home_sequence_s32 servo-thread
...
net home_seq_float <= home_sequence_mux.out
net home_seq_float => home_sequence_s32.in
net home_seq_s32 <= home_sequence_s32.out
net home_seq_s32 => ini.0.home_sequence
net home_seq_s32 => ini.1.home_sequence
...
# allow_jjog: pin created by a virtual panel or hardware switch
net hsequence_select <= allow_jjog
net hsequence_select => home_sequence_mux.sel
net hsequence_select => motion.homing-inhibit
```

NOTE

INIHAL ピン（INI.N.HOME_SEQUENCE など）は MILLTASK が開始するまで使用できないため、上記の HAL コマンドの実行は、POSTGUIHALFILE または遅延[APPLICATION] APP =スクリプトを使用して延期する必要があります。

NOTE

複数のジョイントのジョイントジョギングをリアルタイムで同期するには、手動パルスジェネレータ（MPG）タイプのジョグピン（JOINT.N.ENABLE、JOINT.N.SCALE、JOINT.N.COUNTS）の追加の HAL 接続が必要です。

負のホームシーケンスを使用する場合のジョイントジョギングを示すシミュレーション構成の例（GANTRY_JJOG.INI）は、CONFIGS / SIM / AXIS / GANTRY /ディレクトリにあります。

6.8 旋盤構成

6.8.1 デフォルトの平面

LinuxCNC の通訳者が最初に書かれたとき、それは工場向けに設計されました。 そのため、デフォルトの平面は XY（G17）です。 通常の旋盤は XZ 平面（G18）のみを使用します。 デフォルトのプレーンを変更するには、RS274NGC セクションの.ini ファイルに次の行を配置します。

```
RS274NGC_STARTUP_CODE = G18
```

上記は g コードプログラムで上書きできるので、常に g コードファイルのプリアンブルに重要なものを設定してください。

6.8.2 INI 設定

Axis の旋盤モードには、.ini ファイルの通常の設定に加えて、またはそれを置き換えるために、次の.ini 設定が必要です。 これらの履歴設定では、アイデンティティキネマティクス（トリブキン）と、座標 x、y、z に対応する 3 つのジョイント（0、1、2）を使用します。 未使用の y 軸のジョイント 1 は必須ですが、これらの履歴構成では使用されません。 シミュレートされた旋盤構成は、これらの履歴設定を使用する場合があります。 Gmoccapy も上記の設定を使用しますが、追加の設定を提供します。詳細については、gmoccapy セクションを確認してください。

```
[DISPLAY]
DISPLAY = axis
LATHE = 1
...
[KINS]
KINEMATICS = trivkins
JOINTS = 3
[TRAJ]
COORDINATES = X Z
...
[JOINT_0]
...
[JOINT_2]
...
[AXIS_X]
...
[AXIS_Z]
...
```

joints_axes を組み込むと、coordinates =パラメーターで trivkins を指定することにより、必要な 2 つのジョイントだけでより簡単な構成を行うことができます。

```
[DISPLAY]
DISPLAY = axis
LATHE = 1
...
[KINS]
KINEMATICS = trivkins coordinates=xz
JOINTS = 2
```

```
[TRAJ]
COORDINATES = X Z
...
[JOINT_0]
...
[JOINT_1]
...
[AXIS_X]
...
[AXIS_Z]
...
```

6.9 HALTCL ファイル

HALCMD 言語は、コンポーネントと接続の指定に優れていますが、計算機能は提供していません。 その結果、INI ファイルは、高級言語で可能な明快さと簡潔さに制限されます。

HALTCL 機能は、INIT ファイルの計算、ループ、分岐、プロシージャなどに TCL スクリプトとその機能を使用する手段を提供します。 この機能を使用するには、ハーフイルに TCL 言語と拡張子.TCL を使用します。

.TCL 拡張子は、INI ファイルを処理するメインスクリプト（LINUXCNC）によって理解されます。 HALTCL ファイルは、INI ファイルの HAL セクションで識別されます（.HAL ファイルと同様）。

例

```
[HAL]
HALFILE = conventional_file.hal
HALFILE = tcl_based_file.tcl
```

適切な注意を払えば、.HAL ファイルと.TCL ファイルを混在させることができます。

6.9.1 互換性

.HAL ファイルで使用される HALCMD 言語は、実際にはより強力な汎用 TCL スクリプト言語のサブセットである単純な構文を持っています。

6.9.2 HALTCL コマンド

HALTCL ファイルは、LINUXCNC ハードウェアアブストラクションレイヤー（HAL）の特定のコマンドで拡張された TCL スクリプト言語を使用します。 HAL 固有のコマンドは次のとおりです。

```
addf, alias,
delf, delsig,
```

```
getp, gets
ptype,
stype,
help,
linkpp, linkps, linksp, list, loadrt, loadusr, lock,
net, newsig,
save, setp, sets, show, source, start, status, stop,
unalias, unlinkp, unload, unloadrt, unloadusr, unlock,
waitusr
```

TCL 組み込みコマンドとの競合が原因で、GETS コマンドと LIST コマンドに 2 つの特殊なケースが発生します。 HALTCL の場合、これらのコマンドの前にキーワード HAL を付ける必要があります。

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

6.9.3 HALTCLINIFILE 変数

INIFILE 変数には、HALCMD と HALTCL の両方からアクセスできますが、構文は異なります。

LINUXCNC INI ファイルは、SECTION および ITEM 指定子を使用して構成アイテムを識別します。

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

...

INI ファイルの値には、フォームを使用して.HAL ファイルのテキスト置換によってアクセスできます。

```
[SECTION]ITEM
```

同じ INI ファイル値は、TCL グローバル配列変数の形式を使用して.TCL ファイルでアクセスできます。

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

たとえば、次のような INI ファイルアイテム。

```
[JOINT_0]
MAX_VELOCITY = 4
```

HALCMD の場合は.HAL ファイルでは[JOINT_0] MAX_VELOCITY として、HALTCL の場合は.TCL ファイルでは$ :: JOINT_0（MAX_VELOCITY）として表されます。

INIFILES は同じ SECTION で同じ ITEM を複数回繰り返すことができるため、$ :: SECTION（ITEM）は実際には個々の値の TCL リストです。

値が 1 つだけで、それが単純な値である場合（空白のない文字と数字だけのすべての値がこのグループに含まれる）、$ :: SECTION（ITEM）をリストではないかのように扱うことができます。

値に特殊文字（引用符、中括弧文字、埋め込み空白、および TCL で特別な意味を持つその他の文字）を含めることができる場合は、値のリストと初期（場合によっては唯一の）値を区別する必要があります。 リスト。

TCL では、これは[LINDEX $ :: SECTION（ITEM）0]と書かれています。

例：次の INI 値が与えられた場合

```
[HOSTMOT2]
DRIVER=hm2_eth
IPADDR="10.10.10.10"
BOARD=7i92
CONFIG="num_encoders=0 num_pwmgens=0 num_stepgens=6"
```

そしてこの LOADRT コマンド：

```
loadrt $::HOSTMOT2(DRIVER) board_ip=$::HOSTMOT2(IPADDR) config=$::HOSTMOT2(CONFIG)
```

実行される実際のコマンドは次のとおりです。

```
loadrt hm2_eth board_ip={"10.10.10.10"} config={"num_encoders=0 num_pwmgens=0
num_stepgens  -
=6"}
```

LOADRT が中括弧を認識しないため、これは失敗します。

したがって、INI ファイルに入力されたとおりの値を取得するには、LOADRT 行を次のように書き直します。

```
loadrt $::HOSTMOT2(DRIVER) board_ip=[lindex $::HOSTMOT2(IPADDR) 0] config=[lindex $::
-
HOSTMOT2(CONFIG) 0]
```

6.9.4 .HAL ファイルを.TCL ファイルに変換する

既存の.HAL ファイルは、上記の違いに適応するために手動で編集することで.TCL ファイルに変換できます。 これらの置換を使用して変換するスクリプトを使用して、プロセスを自動化できます。

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

6.9.5 HALTCL ノート

HALTCL では、SETS および SETP コマンドの VALUE 引数は、TCL 言語の式として暗黙的に扱われます。

例

```
# set gain to convert deg/sec to units/min for JOINT_0 radius
setp scale.0.gain 6.28/360.0*$::JOINT_0(radius)*60.0
```

裸の式で空白を使用することはできません。引用符を使用してください。

```
setp scale.0.gain "6.28 / 360.0 * $::JOINT_0(radius) * 60.0"
```

LOADRT などの他のコンテキストでは、計算式に TCL EXPR コマンド（[EXPR {}]）を明示的に使用する必要があります。

例

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

6.9.6 HALTCL の例

STEPGEN ヘッドルームのトピックを検討してください。 ソフトウェア STEPGEN は、モーションプランナーで使用されるものよりも「少し高い」加速制約で最適に動作します。 したがって、HALCMD ファイルを使用する場合、INIFILE に手動で計算された値を強制します。

```
[JOINT_0]
MAXACCEL = 10.0
STEPGEN_MAXACCEL = 10.5
```

HALTCL を使用すると、TCL コマンドを使用して計算を実行し、STEPGEN_MAXACCELINIFILE アイテムを完全に削除できます。

```
setp stepgen.0.maxaccel $::JOINT_0(MAXACCEL)*1.05
```

もう 1 つの HALTCL 機能は、ループとテストです。 たとえば、多くのシミュレーター構成では、「CORE_SIM.HAL」または「CORE_SIM9.HAL」HAL ファイルを使用します。 これらは、より多くのまたはより少ない軸を接続する必要があるために異なります。 次の HALTCL コードは、TRIVKINS マシンの軸の任意の組み合わせで機能します。

```
# Create position, velocity and acceleration signals for each axis
set ddt 0
for {set jnum 0} {$jnum < $::KINS(JOINTS)} {incr jnum} {
# 'list pin' returns an empty list if the pin doesn't exist
if {[hal list pin joint.${jnum}.motor-pos-cmd] == {}} {
continue
}
net ${jnum}pos joint.${jnum}.motor-pos-cmd  => joint.$axno.motor-pos-fb \
=> ddt.$ddt.in
net ${axis}vel <= ddt.$ddt.out
incr ddt
net ${axis}vel => ddt.$ddt.in
net ${axis}acc <= ddt.$ddt.out
incr ddt
}
puts [show sig *vel]
puts [show sig *acc]
```

6.9.7 HALTCL インタラクティブ

HALRUN コマンドは HALTCL ファイルを認識します。 -T オプションを使用すると、HALTCL を TCL インタープリターとして対話的に実行できます。 この機能は、テストおよびスタンドアロンの HAL アプリケーションに役立ちます。

例

```
$ halrun -T haltclfile.tcl
```

6.9.8 HALTCL 配布の例（SIM）

CONFIGS / SIM / AXIS / SIMTCL ディレクトリには、.TCL ファイルを使用して 2 パス処理の使用と組み合わせて HALTCL 構成を示す INI ファイルが含まれています。 この例は、TCL プロシージャの使用、ループ、コメントの使用、および端末への出力を示しています。

6.10 拡張 G コードの再マップ

6.10.1 はじめに：コードを再マッピングすることによる RS274NGC インタープリターの拡張

1.1.1.1 定義：コードの再マッピング

コードを再マッピングするということは、次のいずれかを意味します。

1. 新しい（つまり、現在割り当てられていない）M コードまたは G コードのセマンティクスを定義します
2. -現在制限されている-既存のコードのセットのセマンティクスを再定義します。

6.10.1.1 なぜ RS274NGC インタープリターを拡張したいのですか？

RS274NGC インタープリターによって現在理解されているコードのセット（M、G、T、S、F）は固定されており、構成オプションによって拡張することはできません。

特に、これらのコードの一部は、実行されるステップの固定シーケンスを実装します。 M6 のように、これらのいくつかは、INI ファイルオプションを介してこれらのステップのいくつかをアクティブ化またはスキップすることで適度に構成できますが、全体的な動作はかなり厳格です。 したがって、この状況に満足している場合は、このマニュアルセクションは適していません。

多くの場合、これは、すぐに使用できない構成またはマシンをサポートするのが面倒または不可能であるか、C / C +\+言語レベルでの変更に頼る必要があることを意味します。 後者は、正当な理由で人気がありません。内部を変更するには、インタープリターの内部を深く理解する必要があり、さらに、独自のサポートの問題が発生します。 特定のパッチがメインの LINUXCNC ディストリビューションに組み込まれる可能性があると考えられますが、このアプローチの結果は、特殊なケースのソリューションの寄せ集めになります。

この欠陥の良い例は、LINUXCNC での工具交換のサポートです。ランダムな工具交換機は十分にサポートされていますが、たとえば自動工具長オフセットスイッチを使用して手動工具交換機の構成を合理的に定義することはほぼ不可能です。 工具交換後に訪問し、それに応じてオフセットを設定します。 また、非常に特殊なラックツールチェンジャー用のパッチは存在しますが、メインコードベースに戻る方法は見つかりませんでした。

ただし、これらの多くは、組み込みコードの代わりに O-WORD プロシージャを使用することで修正できます。つまり、-不十分な-組み込みコードを実行する場合は、代わりに O-WORD プロシージャを呼び出してください。 可能ではありますが、面倒です。NGC プログラムのソース編集が必要であり、不足しているコードへのすべての呼び出しを O-WORD プロシージャ呼び出しに置き換えます。

最も単純な形式では、再マップされたコードは、O-WORD プロシージャへの自発的な呼び出しにすぎません。 これは舞台裏で行われます。手順は構成レベルでは表示されますが、NGC プログラムレベルでは表示されません。

一般に、再マップされたコードの動作は、次の方法で定義できます。

- **目的の動作を実装する O-WORD サブルーチンを定義します**
- **または、インタプリタの動作を拡張する PYTHON 関数を使用することもできます。**

M コードと G コード、および O ワードのサブルーチン呼び出しを結合する方法には、かなり異なる構文があります。

たとえば、O-WORD プロシージャは、次のような特定の構文を持つ位置パラメータを取ります。

```
o<test> call [1.234] [4.65]
```

一方、M コードまたは G コードは通常、必須またはオプションの単語パラメータを取ります。 たとえば、G76（スレッド化）には P、Z、I、J、および K ワードが必要であり、オプションで R、Q、H、E、および L ワードを使用します。

したがって、コード X に遭遇したときはいつでも、プロシージャ Y を呼び出すだけでは不十分です。少なくともパラメータのチェックと変換を行う必要があります。 これには、新しいコードとそれに対応する NGC プロシージャの間にいくつかのグルーコードが必要であり、NGC プロシージャに制御を渡す前に実行する必要があります。

RS274NGC 言語には、必要な効果を達成するための内省的な機能とインタープリターの内部データ構造へのアクセスがないため、このグルーコードを O-WORD プロシージャ自体として記述することは不可能です。 -再び--C/C + \ +でグルーコードを実行することは柔軟性がなく、したがって不十分な解決策になります。

EMBEDDED PYTHON の適合性単純な状況を簡単にし、複雑な状況を解決できるようにするために、接着剤の問題は次のように対処されます。

- 単純な状況では、組み込みの接着手順（ARGSPEC）が最も一般的なパラメーター受け渡し要件をカバーします
- T、M6、M61、S、F の再マッピングについては、ほとんどの状況をカバーする標準の PYTHON 接着剤があります。標準接着剤を参照してください。
- より複雑な状況では、独自の PYTHON 接着剤を作成して、新しい動作を実装できます。

インタプリタに埋め込まれた PYTHON 関数は、最初はグルーコードとして始まりましたが、それをはるかに超えて非常に便利であることがわかりました。 PYTHON に精通しているユーザーは、やや面倒な RS274NGC 言語にまったく頼ることなく、純粋な PYTHON で再マップされたコード、接着剤、O-WORD プロシージャなどを簡単に記述できることに気付くでしょう。

組み込み PYTHON に関する一言多くの人が C/C ++モジュールによる PYTHON インタープリターの拡張に精通しており、これは LINUXCNC で、PYTHON スクリプトからタスク、HAL、およびインタープリターの内部にアクセスするために頻繁に使用されます。 PYTHON の拡張とは、基本的に次のことを意味します。PYTHON スクリプトは運転席にあるまま実行され、C / C +\+で記述された拡張モジュールをインポートして使用することで PYTHON 以外のコードにアクセスできます。 この例は、LINUXCNC HAL、GCODE、および EMC モジュールです。

埋め込まれた PYTHON は少し異なり、あまり知られていません。メインプログラムは C / C ++で記述されており、サブルーチンのように PYTHON を使用する場合があります。 これは強力な拡張メカニズムであり、多くの成功したソフトウェアパッケージに見られるスクリプト拡張の基礎です。 埋め込まれた PYTHON コードは、同様の拡張モジュールメソッドを介して C / C +\+変数および関数にアクセスできます。

6.10.2 入門

コードの定義には、次の手順が含まれます。

- コードを選択します-未割り当てのコードを使用するか、既存のコードを再定義します
- パラメータの処理方法を決定する
- 結果を処理するかどうか、およびどのように処理するかを決定します
- 実行順序を決定します。

1.1.1.1 組み込みのリマップ

現在、STDGLUE.PY で利用できる完全な PYTHON のみのリマップが 2 つあります。

- IGNORE_M6
- INDEX_LATHE_TOOL_WITH_WEAR

これらは旋盤で使用するためのものです。

旋盤はツールのインデックス作成に M6 を使用せず、T コマンドを使用します。

この再マップは、工具オフセットに摩耗オフセットも追加します。

すなわち。 T201 は（ツール 2 のツールオフセットを使用して）ツール 2 にインデックスを付け、摩耗オフセット 1 を追加します。

ツールテーブルでは、10000 を超えるツール番号は摩耗オフセットです。

つまり、ツールテーブルでは、ツール 10001 は摩耗オフセット 1 になります。

それらを使用するために INI で必要なものは次のとおりです。

```
REMAP=T python=index_lathe_tool_with_wear
REMAP=M6 python=ignore_m6
[PYTHON]
# where to find the Python code:
# code specific for this configuration
PATH_PREPEND=./
# generic support code - make sure this actually points to python-stdglue
PATH_APPEND=../../nc_files/remap_lib/python-stdglue/
# import the following Python module
TOPLEVEL=toplevel.py
# the higher the more verbose tracing of the Python plugin
LOG_LEVEL = 0
```

また、構成フォルダーに必要な PYTHON ファイルを追加する必要があります。

既存の構成をアップグレードする

6.10.2.1 コードを選ぶ

現在、再定義できる既存のコードはごくわずかですが、再マッピングによって利用できるようになる可能性のある無料のコードが多数あることに注意してください。 再定義された既存のコードを開発するときは、割り当てられていない G コードまたは M コードから始めて、既存の動作と新しい動作の両方を実行できるようにすることをお勧めします。 完了したら、既存のコードを再定義して、再マッピング設定を使用します。

- ユーザー定義に公開されている未使用の M コードの現在のセットは、ここにあります。
- 未割り当ての G コードがここに一覧表示されます。
- 再マップされる可能性のある既存のコードがここにリストされています。

6.10.2.2 パラメータ処理

新しいコードが NGC プロシージャによって定義され、いくつかのパラメータが必要であると仮定します。その一部は必須であり、その他はオプションである可能性があります。 プロシージャに値をフィードするための次のオプションがあります。

1. 現在のブロックから単語を抽出し、パラメーターとしてプロシージャに渡します（X22.34 や P47 など）
2. INI ファイル変数を参照する
3. グローバル変数を参照する（＃2200 = 47.11 または＃<_ GLOBAL_PARAM> = 315.2 など）

最初の方法は、位置などの動的な性質のパラメータに適しています。 現在のブロックのどの単語が新しいコードにとって意味があるかを定義し、それが NGC プロシージャにどのように渡されるかを指定する必要があります。 簡単な方法は、ARGSPEC ステートメントを使用することです。 カスタムプロローグはより良いエラーメッセージを提供するかもしれません。

TO INI ファイル変数を使用すると、マシンのセットアップ情報を参照する場合に最も役立ちます。たとえば、工具長センサーの位置などの固定位置です。 この方法の利点は、現在実行している NGC ファイルに関係 なく、構成に合わせてパラメーターが固定されることです。

グローバル変数の参照は常に可能ですが、見落とされがちです。

パラメータとして使用できる単語の数には限りがあるため、多くのパラメータが必要な場合は、2 番目と 3 番目の方法にフォールバックする必要がある場合があります。

6.10.2.3 結果の処理

たとえば、無効なパラメータの組み合わせが渡された場合、新しいコードは成功または失敗する可能性があります。 または、手順を実行して結果を無視することを選択することもできます。その場合、実行する作業はそれほど多くありません。

EPILOG ハンドラーは、再マップ手順の結果の処理に役立ちます。参照セクションを参照してください。

6.10.2.4 実行シーケンス

実行可能な G コードワードはモーダルグループに分類され、モーダルグループはそれらの相対的な実行動作も定義します。

G コードブロックの行に複数の実行可能ワードが含まれている場合、これらのワードは、ブロックに表示される順序ではなく、事前定義された実行順序で実行されます。

新しい実行可能コードを定義するとき、インタープリターはコードがこのスキームのどこに適合するかをまだ知りません。 このため、コードを実行するための適切なモーダルグループを選択する必要があります。

6.10.2.5 再マップされたコードの最小限の例

ピースがどのように組み合わされるかを理解するために、かなり最小限であるが完全に再マップされたコード定義を調べてみましょう。 未割り当ての M コードを選択し、次のオプションを INI ファイルに追加します。

```
[RS274NGC]
```

```
REMAP=M400 modalgroup=10 argspec=Pq ngc=myprocedure
```

一言で言えば、これは次のことを意味します。

- M400 コードは、必須パラメーター P とオプションパラメーター Q を取ります。現在のブロック内の他の単語は、M400 コードに関して無視されます。 P ワードが存在しない場合は、エラーで実行に失敗します。
- M400 コードが検出されたら、実行順序に従って、他のモーダルグループ 10M コードに沿って MYPROCEDURE.NGC を実行します。
- P と Q の値は、ローカルの名前付きパラメーターとしてプロシージャで使用できます。 これは、＃<P>および＃<Q>と呼ばれる場合があります。 この手順では、Q ワードが EXISTS 組み込み関数とともに存在するかどうかをテストできます。

MYPROCEDURE.NGC ファイルは、[DISPLAY]NC_FILES または[RS274NGC]SUBROUTINE_PATH ディレクトリに存在する必要があります。

REMAP パラメータの詳細については、以下の参照セクションを参照してください。

6.10.3 再マッピングの構成

1.1.1.1 REMAP ステートメント

コードを再マップするには、INI ファイルの RS274NG セクションで REMAP オプションを使用してコードを定義します。 再マップされたコードごとに 1 つの REMAP 行を使用します。

REMAP の構文は次のとおりです。

REMAP=<code> <options>

ここで、<code>は、T、M6、M61、S、F（既存のコード）のいずれか、または未割り当ての M コードまたは G コードのいずれかです。

<CODE>パラメータを省略するとエラーになります。

REMAP ステートメントのオプションは、空白で区切られています。 オプションはキーワードと値のペアであり、現在は次のとおりです。

modalgroup=<modal group>

G-CODES

現在サポートされている唯一のモーダルグループは 1 です。これは、グループが指定されていない場合のデフォルト値でもあります。 グループ 1 は、他の G コードと一緒に実行することを意味します。

M-CODES

現在サポートされているモーダルグループは、5、6、7、8、9、10 です。 モーダルグループが指定されていない場合、デフォルトで 10 になります（ブロック内の他のすべての単語の後に実行します）。

T,S,F

これらの場合、モーダルグループは固定され、MODALGROUP=オプションはすべて無視されます。

ARGSPEC=<ARGSPEC>

ARGSPEC パラメーターオプションの説明を参照してください。 オプション。

NGC=<NGC_BASENAME>

O ワードサブルーチンファイル名のベース名。 .NGC 拡張子を指定しないでください。 [DISPLAY] PROGRAM_PREFIX で指定されたディレクトリで指定されたディレクトリで検索され、次に[RS274NGC]SUBROUTINE_PATH で検索されました。 PYTHON=と相互に排他的です。 NGC=と PYTHON=の両方を省略するとエラーになります。

PYTHON=<PYTHON FUNCTION NAME>

NGC O-WORD プロシージャを呼び出す代わりに、PYTHON 関数を呼び出します。 この関数は、MODULE_BASENAME.OWORD モジュールで定義されている必要があります。 NGC=と相互に排他的です。

PROLOG=<PYTHON FUNCTION NAME>

NGC プロシージャを実行する前に、この PYTHON 関数を呼び出します。 この関数は、MODULE_BASENAME.REMAP モジュールで定義されている必要があります。 オプション。

EPILOG=<PYTHON FUNCTION NAME>

NGC プロシージャを実行した後、この PYTHON 関数を呼び出します。 この関数は、MODULE_BASENAME.REMAP モジュールで定義されている必要があります。 オプション。

PYTHON、PROLOG、EPILOG オプションでは、PYTHON INTERPRETER プラグインを構成し、適切な PYTHON 関数を定義して、これらのオプションで参照できるようにする必要があります。

新しいコードを定義するための構文と、既存のコードを再定義するための構文は同じです。

6.10.3.1 便利な REMAP オプションの組み合わせ

ARGSPEC オプションの多くの組み合わせが可能ですが、すべてが意味をなすわけではないことに注意してください。 次の組み合わせは便利なイディオムです。

ARGSPEC=<WORDS> NGC=<PROCNAME> MODALGROUP=<GROUP>

標準の ARGSPEC パラメーター変換を使用して NGC プロシージャーを呼び出すための推奨される方法。 ARGSPEC が十分に良い場合に使用されます。 Tx および M6/M61 ツールの変更コードを再マッピングするには不十分であることに注意してください。

PROLOG=<PYTHONPROLOG> NGC=<PROCNAME> EPILOG=<PYTHONEPILOG> MODALGROUP=<GROUP>

PYTHON プロローグ関数を呼び出して準備手順を実行してから、NGC プロシージャを呼び出します。 完了したら、PYTHON エピログ関数を呼び出して、G コードで処理できないクリーンアップまたは結果抽出作業を実行します。 インタープリターの内部変数のほとんどすべて、および一部の内部関数はプロローグおよびエピローグハンドラーからアクセスできるため、コードを NGC プロシージャに再マッピングする最も柔軟な方法。 また、自分を吊るすための長いロープ。

PYTHON=<PYTHONFUNCTION> MODALGROUP=<GROUP>

引数を変換せずに PYTHON 関数を直接呼び出します。 コードを再マッピングして PYTHON に直接移行する最も強力な方法。 NGC の手順が必要ない場合、または NGC が邪魔をしている場合は、これを使用します。

ARGSPEC=<WORDS> PYTHON=<PYTHONFUNCTION> MODALGROUP=<GROUP>

ARGSPEC 単語を変換し、キーワード引数辞書として PYTHON 関数に渡します。 ブロックに渡された単語を自分で調べるのが面倒な場合に使用します。

達成したいのが G コードから PYTHON コードを呼び出すことだけである場合、O-WORD プロシージャのような PYTHON 関数を呼び出すためのやや簡単な方法があることに注意してください。

6.10.3.2 ARGSPEC パラメーター

引数の指定（キーワード ARGSPEC）は、NGC プロシージャに渡される必須およびオプションの単語と、そのコードを実行するためのオプションの前提条件を記述します。

ARGSPEC は、クラス[@ A-KMNP-ZA-KMNP-Z^>]の 0 個以上の文字で構成されます。 空にすることもできます（ARGSPEC =のように）。 空の ARGSPEC、または ARGSPEC 引数がまったくない場合は、再マップされたコードがブロックからパラメーターを受け取らないことを意味します。 存在する余分なパラメータは無視されます。

RS274NGC ルールが引き続き適用されることに注意してください。たとえば、G コードのコンテキストでのみ軸ワード（X、Y、Z など）を使用できます。

ABCDEFGHIJKMPQRSTUVWXYZ

必須の単語パラメータを定義します。大文字は、対応する単語が現在のブロックに存在する必要があることを指定します。 単語の値は、対応する名前を持つローカルの名前付きパラ

メータとして渡されます。 @文字が ARGSPEC に存在する場合、それは位置パラメーターとして渡されます。以下を参照してください。

abcdefghijkmpqrstuvwxyz

オプションの単語パラメータを定義します。小文字は、対応する単語が現在のブロックに存在する可能性があることを示します。 単語が存在する場合、単語の値はローカルの名前付きパラメータとして渡されます。 @文字が ARGSPEC に存在する場合、それは位置パラメーターとして渡されます。以下を参照してください。

@

@（アットマーク）は、@オプションに続いて定義された順序で、単語を位置パラメーターとして渡すように ARGSPEC に指示します。 位置パラメータの受け渡しを使用する場合、プロシージャは単語が存在するかどうかを判断できないことに注意してください。以下の例を参照してください。

注記

これは、既存の NGC プロシージャを再マップされたコードとしてパッケージ化するのに役立ちます。 既存の手順では、位置パラメータが必要です。 @オプションを使用すると、ローカルの名前付きパラメーターを参照するように書き換えることを回避できます。

---

^

^（CARET）文字は、現在のスピンドル速度がゼロ（スピンドル実行中）より大きくなければならないことを指定します。そうでない場合、コードは適切なエラーメッセージで失敗します。

>

>（大なり記号）文字は、現在のフィードがゼロより大きくなければならないことを指定します。そうでない場合、コードは適切なエラーメッセージで失敗します。

n

n（大なり記号）文字は、`n`local 名前付きパラメーターで現在の行番号を渡すことを指定します。

デフォルトでは、パラメータはローカルの名前付きパラメータとして NGC プロシージャに渡されます。 これらのローカルパラメータは、プロシージャの実行開始時にすでに設定されているように見えます。これは、既存のセマンティクスとは異なります（ローカル変数は、値 0.0 で始まり、明示的に値を割り当てる必要があります）。

オプションの単語パラメータは、EXISTS（#<word>）イディオムによって存在をテストできます。

NGC プロシージャに渡される名前付きパラメータの例コードが次のように定義されていると仮定します。

```
REMAP=M400 modalgroup=10 argspec=Pq ngc=m400
```

m400.ngc は次のようになります。

```
o<m400> sub
(P is required since it’s uppercase in the argspec)
(debug, P word=#<P>)
(the q argspec is optional since its lowercase in the argspec. Use as follows:)
o100 if [EXISTS[#<q>]]
(debug, Q word set: #<q>)
o100 endif
o<m400> endsub
M2
```

- M400 の実行は、ユーザー定義の M400：missing：P というメッセージで失敗します。
- M400 P123 を実行すると、P word=123.000000 が表示されます。
- M400 P123 Q456 を実行すると、P ワード= 123.000000 および Q ワードセット：456.000000 が表示されます。

NGC プロシージャに渡される位置パラメータの例コードが次のように定義されていると仮定します。

```
REMAP=M410 modalgroup=10 argspec=@PQr ngc=m410
```

m410.ngc は次のようになります。

```
o<m410> sub
(debug, [1]=#1 [2]=#2 [3]=#3)
o<m410> endsub
M2
```

- M410 P10 を実行すると、m410.ngc が表示されます：[1] = 10.000000 [2] = 0.00000
- M410 P10 Q20 を実行すると、m410.ngc が表示されます：[1] = 10.000000 [2] = 20.000000

注意：複数のオプションのパラメータワードを区別する機能が失われ、オプションのパラメータが存在するが値が 0 であるか、まったく存在しないかがわかりません。

Python 関数に渡す名前付きパラメータの簡単な例 NGC プロシージャなしで新しいコードを定義することが可能です。 これは簡単な最初の例です。より複雑な例は次のセクションにあります。

コードが次のように定義されていると仮定します

```
REMAP=G88.6 modalgroup=1 argspec=XYZp python=g886
```

これは、module_basename.remap モジュールで Python 関数 g886 を実行するようにインタプリタに指示します。これは次のようになります。

```
from interpreter import INTERP_OK
from emccanon import MESSAGE
def g886(self, **words):
for key in words:
MESSAGE("word '%s' = %f" % (key, words[key]))
if words.has_key('p'):
MESSAGE("the P word was present")
MESSAGE("comment on this line: '%s'" % (self.blocks[self.remap_level].comment))
return INTERP_OK
```

g88.6 x1 y2 z3 g88.6 x1 y2 z3 p33（ここにコメント）

組み込み Python 環境が徐々に導入されていることに気付くでしょう。詳細については、こちらをご覧ください。 Python の再マッピング関数では、そもそも Python 関数を実行するため、Python のプロローグ関数やエピローグ関数を使用しても意味がないことに注意してください。

高度な例：純粋な Python で再マップされたコードインタープリターと emccanon モジュールは、ほとんどのインタープリターと一部の Canon 内部を公開するため、これまで C / C +\+でのコーディングが必要だった多くのことを Python で実行できるようになりました。

次の例は、nc_files / involute.py スクリプトに基づいていますが、パラメーターの抽出とチェックを行う G コードとして缶詰になっています。 また、インタープリターを再帰的に呼び出す方法も示しています（self.execute（）を参照）。 そのような定義を仮定すると（注：これは argspec を使用しません）：

```
REMAP=G88.1 modalgroup=1 py=involute
```

以下にリストされている python/remap.py のインボリュート関数は、現在のブロックからすべての単語を直接抽出します。 インタプリタエラーは Python 例外に変換される可能性があることに注意してください。 これは先読み時間であることを忘れないでください。実行時間エラーをこの方法でトラップすることはできません。

```
import sys
import traceback
from math import sin,cos
from interpreter import *
from emccanon import MESSAGE
```

```
from util import lineno, call_pydevd
# raises InterpreterException if execute() or read() fails
throw_exceptions = 1
def involute(self, **words):
""" remap function with raw access to Interpreter internals """
if self.debugmask & 0x20000000: call_pydevd() # USER2 debug flag
if equal(self.feed_rate,0.0):
return "feedrate > 0 required"
if equal(self.speed,0.0):
return "spindle speed > 0 required"
plunge = 0.1 # if Z word was given, plunge - with reduced feed
# inspect controlling block for relevant words
c = self.blocks[self.remap_level]
x0 = c.x_number if c.x_flag else 0
y0 = c.y_number if c.y_flag else 0
a = c.p_number if c.p_flag else 10
old_z = self.current_z
if self.debugmask & 0x10000000:
print "x0=%f y0=%f a=%f old_z=%f" % (x0,y0,a,old_z)
try:
#self.execute("G3456") # would raise InterpreterException
self.execute("G21",lineno())
self.execute("G64 P0.001",lineno())
self.execute("G0 X%f Y%f" % (x0,y0),lineno())
if c.z_flag:
feed = self.feed_rate
self.execute("F%f G1 Z%f" % (feed * plunge, c.z_number),lineno())
self.execute("F%f" % (feed),lineno())
for i in range(100):
t = i/10.
x = x0 + a * (cos(t) + t * sin(t))
y = y0 + a * (sin(t) - t * cos(t))
self.execute("G1 X%f Y%f" % (x,y),lineno())
if c.z_flag: # retract to starting height
self.execute("G0 Z%f" % (old_z),lineno())
except InterpreterException,e:
msg = "%d: '%s' - %s" % (e.line_number,e.line_text, e.error_message)
return msg
return INTERP_OK
```

これまでに説明した例は、configs / sim / axis / remap/getting-started と完全に機能する構成にあります。

6.10.4 再マッピングのための既存の構成のアップグレード

REMAP ステートメントを使用するための最小の前提条件は次のとおりです。

- Python プラグインは、ini ファイルで[PYTHON] TOPLEVEL=<path-to-toplevel-script>を指定してアクティブ化する必要があります。
- トップレベルのスクリプトは、最初は空にすることができる remap モジュールをインポートする必要がありますが、インポートが適切に行われている必要があります。
- Python インタープリターは上記の remap.py モジュールを見つける必要があるため、Python モジュールが存在するディレクトリへのパスを[PYTHON] PATH_APPEND=<path-to-your-local-Python-directory>で追加する必要があります。
- 推奨：再マップモジュールに stdglue ハンドラーをインポートします。 この場合、Python は stdglue.py も見つける必要があります。ディストリビューションからコピーするだけなので、必要に応じてローカルで変更を加えることができます。 インストールによって、stdglue.py へのパスが異なる場合があります。

構成が/home/ user / xxx の下にあり、ini ファイルが/home/user/xxx/xxx.ini であると仮定して、次のコマンドを実行します。

```
$ cd /home/user/xxx
$ mkdir python
$ cd python
$ cp /usr/share/linuxcnc/ncfiles/remap_lib/python-stdglue/stdglue.py .
$ echo 'from stdglue import *' >remap.py
$ echo 'import remap' >toplevel.py
```

次に、/home/user/xxx/xxx.ini を編集して、以下を追加します。

```
[PYTHON]
TOPLEVEL=/home/user/xxx/python/toplevel.py
PATH_APPEND=/home/user/xxx/python
```

次に、LinuxCNC にエラーメッセージが表示されないことを確認します。ターミナルウィンドウから次のコマンドを実行します。

```
$ cd /home/user/xxx
$ linuxcnc xxx.ini
```

6.10.5 ツール変更関連コードの再マッピング：T、M6、M61

1.1.1.1 概要

LinuxCNC の内部に慣れていない場合は、最初に「ツールの変更が現在どのように機能するか」セクションをお読みください（悲惨ですが必要です）。 既存のコードを再マッピングする場合は、インタープリターのこのコードの組み込み機能を完全に無効にすることに注意してください。

したがって、再マップされたコードは、マシンを好きなように移動するためのコマンドを生成するだけでなく、インタープリターとタスクを満足させるために必要なこのシーケンスからのステップを複製する必要があります。

ただし、これは、task および iocontrol でのツール変更関連コマンドの処理には影響しません。 これは、ステップ 6b を実行しても、iocontrol がその処理を実行することを意味します。

決定、決定：

- O-word プロシージャを使用しますか、それともすべて Python コードで実行しますか？
- iocontrol HAL シーケンス（tool-prepare/tool-prepared および tool-change/tool-changed ピン）で十分ですか、それともツールチェンジャーに別の種類の HAL インタラクションが必要ですか（たとえば、別のピンに関連するより多くの HAL ピン） 相互作用シーケンス）？

答えに応じて、4 つの異なるシナリオがあります。

- O-word プロシージャを使用する場合、プロローグ関数とエピローグ関数が必要です
- すべての Python コードを使用し、O-word プロシージャを使用しない場合は、Python 関数で十分です。
- iocontrol ピンを使用する場合、O-word プロシージャまたは Python コードには主に移動が含まれます
- iocontrol によって提供されるよりも複雑な相互作用が必要な場合は、motion.digital*ピンと motion.analog*ピンを使用して独自の相互作用を完全に定義し、それらをループして iocontrol ピンを本質的に無視する必要があります。

Note

O-word プロシージャが嫌いで、Python が好きな場合は、すべてを Python で自由に実行できます。その場合、REMAP ステートメントに python=<function>仕様が含まれているだけです。 しかし、ほとんどの人が O-word プロシージャに精通しているため、O-word プロシージャの使用に興味があると仮定して、最初の例としてそれを行います。

したがって、最初の例の全体的なアプローチは次のようになります。

1. 柔軟性を高めるために、O-word プロシージャで G コードを可能な限り使用したいと考えています。 これには、通常は iocontrol によって処理されるすべての HAL インタラクションが含まれます。これは、移動、プローブ、HAL ピン I/O などを巧妙に処理したいためです。
2. インタプリタを満足させるために必要な範囲で Python コードを最小限に抑え、タスクに実際に何かを実行させるようにします。 それはプロローグとエピローグの Python 関数に入ります。

6.10.5.1 再マップされたツール変更コードを使用した iocontrol の役割の理解

Iocontrol は、使用する場合と使用しない場合がある 2 つの HAL 相互作用シーケンスを提供します。

- SELECT_POCKET（）canon コマンドによってキューに入れられた NML メッセージが実行されると、XXXX ピンの設定に加えて、iocontrol で「ツールの準備を上げてツールの準備がハイになるのを待つ」HAL シーケンスがトリガーされます。
- CHANGE_TOOL（）canon コマンドによってキューに入れられた NML メッセージが実行されると、XXXX ピンの設定に加えて、iocontrol で「ツール変更を発生させてツール変更がハイになるのを待つ」HAL シーケンスがトリガーされます。

決定する必要があるのは、既存の iocontrolHAL シーケンスがチェンジャーを駆動するのに十分であるかどうかです。 おそらく、別の相互作用シーケンスが必要です。たとえば、より多くの HAL ピン、またはより複雑な相互作用です。 回答に応じて、既存の iocontrol HAL シーケンスを引き続き使用するか、独自のシーケンスを定義する場合があります。

ドキュメントのために、これらの iocontrol シーケンスを無効にして、独自のシーケンスをロールします。結果は既存のインタラクションのように見えますが、独自の O-word プロシージャで実行されるため、これらを完全に制御できます。

したがって、motion.digital-*ピンと motion.analog-*ピン、および関連する M62 .. M68 コマンドを使用して、O-word プロシージャで独自の HAL インタラクションを実行します。これらのコマンドは、iocontrol を効果的に置き換えます。 tool-prepare/toolprepared および tool-change/tool-changed シーケンス。 そこで、既存の iocontrol ピンを機能的に置き換えるピンを定義し、先に進んで iocontrol の相互作用をループにします。 この例では、次の対応を使用します。

例の Iocontrol ピンの対応

iocontrol.0pin	motion pin
tool-prepare	digital-out-00
tool-prepared	digital-in-00
tool-change	digital-out-01
tool-changed	digital-in-01
tool-prep-number	analog-out-00
tool-prep-pocket	analog-out-01

```
tool-number                    analog-out-02
```

M6 コマンドを再定義し、それを O-word プロシージャに置き換えたいと仮定しますが、それ以外は引き続き機能するはずです。

したがって、O-word の手順では、ここで概説した手順を置き換える必要があります。 これらの手順を確認すると、すべてではありませんが、ほとんどの手順で NGC コードを使用できることがわかります。 したがって、NGC が処理できないものは、Python のプロローグ関数とエピローグ関数で実行されます。

6.10.5.2 M6 交換の指定

アイデアを伝えるために、組み込みの M6 セマンティクスを独自のセマンティクスに置き換えるだけです。 それが機能したら、先に進んで、O-word 手順に適合すると思われるアクションを配置できます。

手順を実行すると、次のことがわかります。

1. すでに実行されている T コマンドを確認します-Python プロローグで実行します
2. カッター補正がアクティブになっているかどうかを確認します-Python プロローグで実行します
3. 必要に応じてスピンドルを停止します-NGC で実行できます
4. クイルアップ-NGC で実行できます
5. TOOL_CHANGE_AT_G30 が設定されている場合：
   a. 該当する場合は、A、B、および C インデクサーを移動します-NGC で実行できます
   b. G30 位置への迅速な移動を生成します-NGC で実行できます
6. CHANGE_TOOLCanon コマンドをタスクに送信します-Python エピログで実行します
7. 新しいツールに従って番号パラメータ 5400-5413 を設定します-Python エピログで実行します
8. ツールの変更が完了するまで、先読みのためのインタプリタの呼び出しを停止するようにタスクに通知します-Python エピログで実行します

したがって、プロローグとエピローグが必要です。 M6 リマップの ini ファイルの呪文が次のようになっていると仮定しましょう。

```
REMAP=M6 modalgroup=6 prolog=change_prolog ngc=change epilog=change_epilog
```

したがって、ステップ 1 と 2 をカバーするプロローグは次のようになります。いくつかの変数を再マップ手順に渡して、そこで検査および変更したり、メッセージで使用したりすることができます。

それらは次のとおりです。tool_in_spindle、selected_tool（ツール番号）およびそれぞれのポケット current_pocket および selected_pocket：

```
def change_prolog(self, **words):
    try:
        if self.selected_pocket < 0:
            return "M6: no tool prepared"
        if self.cutter_comp_side:
            return "Cannot change tools with cutter radius compensation on"
        self.params["tool_in_spindle"] = self.current_tool
        self.params["selected_tool"] = self.selected_tool
        self.params["current_pocket"] = self.current_pocket
        self.params["selected_pocket"] = self.selected_pocket
        return INTERP_OK
    except Exception, e:
        return "M6/change_prolog: %s" % (e)
```

ほとんどのプロローグ関数は非常によく似ていることがわかります。最初にコードを実行するためのすべての前提条件が保持されていることをテストし、次に環境を準備します。変数を挿入したり、NGC コードでは簡単に実行できない準備処理ステップを実行したりします。 次に、INTERP_OK を返し、NGC プロシージャに渡します。

O-word プロシージャの最初の反復は刺激的ではありません。パラメータが正しいことを確認し、正の値を返すことで成功を通知します。 手順 3～5 は、最終的にここで説明されます（ini ファイル設定を参照する変数については、ここを参照してください）。

```
O<change> sub
(debug, change: current_tool=#<current_tool>)
(debug, change: selected_pocket=#<selected_pocket>)
;
; insert any g-code which you see fit here, eg:
; G0 #<_ini[setup]tc_x> #<_ini[setup]tc_y> #<_ini[setup]tc_z>
;
O<change> endsub [1]
m2
```

change.ngc が成功したと仮定して、手順 6～8 をモップアップする必要があります。

```
def change_epilog(self, **words):
    try:
        if self.return_value > 0.0:
```

```
            # commit change
            self.selected_pocket = int(self.params["selected_pocket"])
            emccanon.CHANGE_TOOL(self.selected_pocket)
            # cause a sync()
            self.tool_change_flag = True
            self.set_tool_parameters()
            return INTERP_OK
        else:
            return "M6 aborted (return code %.1f)" % (self.return_value)
    except Exception, e:
        return "M6/change_epilog: %s" % (e)
```

この交換用 M6 は、手順 3〜5 に NGC コードを入力する必要があることを除いて、組み込みコードと互換性があります。

繰り返しになりますが、ほとんどのエピローグには共通のスキームがあります。最初に、再マップ手順で問題がなかったかどうかを判断し、次に NGC コードでは実行できないコミットおよびクリーンアップアクションを実行します。

6.10.5.3 再マップされた M6 を使用した iocontrol の構成

操作の順序が変更されたことに注意してください。チェンジャーを動かすための HAL ピンの設定/読み取り、ツールの変更の確認など、O-word 手順に必要なすべてのことを行います。おそらく motion.digital-*と motion-を使用します。 アナログ-*IO ピン。 最終的に CHANGE_TOOL（）コマンドを実行すると、すべての移動と HAL の相互作用がすでに完了しています。

通常、iocontrol はここで概説されているようにそのことを実行します。 ただし、HAL ピンを小刻みに動かす必要はもうありません。iocontrol に残されているのは、準備と変更が完了したことを受け入れることだけです。

これは、対応する iocontrol ピンが機能しなくなったことを意味します。 したがって、次のように構成することにより、変更をすぐに確認するように iocontrol を構成します。

```
# loop change signals when remapping M6
net tool-change-loop iocontrol.0.tool-change iocontrol.0.tool-changed
```

何らかの理由で Tx を再マップ（準備）する場合は、対応する iocontrol ピンもループする必要があります。

6.10.5.4 変更を書いて O-word 手順を準備する

ncfiles / remap_lib / python-stdglue / stdglue.py にある標準のプロローグとエピローグは、いくつかの公開されたパラメーターを再マップ手順に渡します。

公開されたパラメーターは、現在のリマップに関連するインタープリター内部変数に対応する、リマッププロシージャに表示される名前付きローカル変数です。 公開されたパラメータは、それぞれのプロローグで設定され、エピローグで検査されます。 それらは再マップ手順で変更でき、変更はエピローグで取得されます。 再マップ可能な組み込みコードの公開パラメーターは次のとおりです。

- T（prepare_prolog）：#<ツール>、#<ポケット>
- M6（change_prolog）：#<tool_in_spindle>、#<selected_tool>、#<current_pocket>、#<selected_pocket>
- M61（settool_prolog）：#<ツール>、#<ポケット>
- S（setspeed_prolog）：#<速度>
- F（setfeed_prolog）：#<feed>

追加のパラメーターを表示する必要がある場合は、それをプロローグに追加するだけで済みます。実際には、すべてのインタープリター内部が Python に表示されます。

6.10.5.5 M6 を含む組み込みコードに最小限の変更を加える

通常、コードを再マッピングすると、そのコードのすべての内部処理が完全に無効になることに注意してください。

ただし、状況によっては、ツール長プローブのように、既存の M6 組み込み実装の周りにいくつかのコードを追加するだけで十分な場合がありますが、それ以外の場合は、組み込み M6 の動作を保持します。

これは一般的なシナリオである可能性があるため、再マップされたコードの組み込み動作が再マップ手順内で利用できるようになりました。 インタプリタは、動作を再定義することになっているプロシージャ内で再マップされたコードを参照していることを検出します。 この場合、組み込みの動作が使用されます。これは現在、セットに対して有効になっています：M6、M61、T、S、F）。 そうしないと、独自の再マップ手順内でコードを参照するとエラー、つまり再マッピングの再帰になることに注意してください。

ビルトインを少しひねると、次のようになります（M6 の場合）。

```
REMAP=M6 modalgroup=6 ngc=mychange
```

```
o<mychange> sub
M6 (use built in M6 behavior)
(.. move to tool length switch, probe and set tool length..)
o<mychange> endsub
m2
```

警告

組み込みコードを再定義するときは、G コードまたは M コードに先行ゼロを指定しないでください。たとえば、REMAP =M01...ではなく REMAP=M1..と言います。

組み込みコードを拡張する際の独自の作業の推奨開始点である完全な構成については、configs / sim / axis / remap/extend-builtins ディレクトリを参照してください。

6.10.5.6 T（準備）置換の指定

デフォルトの実装に自信がある場合は、これを行う必要はありません。 ただし、再マッピングは、現在の実装の欠陥を回避する方法でもあります。たとえば、「ツールで準備された」ピンが設定されるまでブロックしないようにします。

たとえば、次のようにできます。-再マップされた T で、「tool-prepare」ピンに相当するものを設定しますが、ここでは「toolprepared」を待ちません-対応する再マップされた M6 で、「tool- O-word 手順の最初に「準備済み」。

繰り返しになりますが、iocontrol tool-prepare / tool-prepared ピンは未使用で、motion。*ピンに置き換えられるため、これらのピンはループする必要があります。

```
# loop prepare signals when remapping T
net tool-prep-loop iocontrol.0.tool-prepare iocontrol.0.tool-prepared
```

したがって、再マップされた T のセットアップは次のとおりです。

```
REMAP=T prolog=prepare_prolog epilog=prepare_epilog ngc=prepare
```

```
def prepare_prolog(self,**words):
    try:
        cblock = self.blocks[self.remap_level]
        if not cblock.t_flag:
            return "T requires a tool number"
        tool = cblock.t_number
```

```
        if tool:
            (status, pocket) = self.find_tool_pocket(tool)
            if status != INTERP_OK:
                return "T%d: pocket not found" % (tool)
        else:
            pocket = -1 # this is a T0 - tool unload
        # these variables will be visible in the ngc oword sub
        # as #<tool> and #<pocket> local variables, and can be
        # modified there - the epilog will retrieve the changed
        # values
        self.params["tool"] = tool
        self.params["pocket"] = pocket
        return INTERP_OK
    except Exception, e:
        return "T%d/prepare_prolog: %s" % (int(words['t']), e)
```

最小の ngc 準備手順も次のようになります。

```
o<prepare> sub
; returning a positive value to commit:
o<prepare> endsub [1]
m2
```

そしてエピローグ：

```
def prepare_epilog(self, **words):
    try:
        if self.return_value > 0:
            self.selected_tool = int(self.params["tool"])
            self.selected_pocket = int(self.params["pocket"])
            emccanon.SELECT_POCKET(self.selected_pocket, self.selected_tool)
            return INTERP_OK
        else:
            return "T%d: aborted (return code %.1f)" % (int(self.params["tool"]),self.  -
                return_value)
    except Exception, e:
        return "T%d/prepare_epilog: %s" % (tool,e)
```

prepare_prolog と prepare_epilog は、nc_files / remap_lib / python-stdglue/stdglue.py によって提供される標準の接着剤の一部です。 このモジュールは、ほとんどの標準的な再マッピング状況を一般的な方法でカバーすることを目的としています。

6.10.5.7 エラー処理：中止の処理

組み込みのツール変更手順には、プログラムの中止に対処するためのいくつかの注意事項があります（たとえば、変更中に Axis で Escape を押す）。 再マップされた関数にはこれがないため、再マップされたコードが中止された場合は、明示的なクリーンアップが必要になる場合があります。 特に、リマッププロシージャは、アボート後にアクティブにすることが望ましくないモーダル設定を確立する場合があります。 たとえば、リマップ手順にモーションコード（G0、G1、G38 ..）があり、リマップが中止された場合、最後のモーダルコードはアクティブのままになります。 ただし、リマップが中止されたときにモーダルモーションをキャンセルすることをお勧めします。

これを行う方法は、[RS274NGC]ON_ABORT_COMMAND 機能を使用することです。 この ini オプションは、タスクが何らかの理由でプログラムの実行を中止した場合に実行される O ワードプロシージャ呼び出しを指定します。

```
[RS274NGC]
ON_ABORT_COMMAND=O <on_abort> call
```

提案された on_abort プロシージャは次のようになります（ニーズに適合）：

```
o<on_abort> sub
G54 (origin offsets are set to the default)
G17 (select XY plane)
G90 (absolute)
G94 (feed mode: units/minute)
M48 (set feed and speed overrides)
G40 (cutter compensation off)
M5 (spindle off)
G80 (cancel modal motion)
M9 (mist and coolant off)
o<on_abort> endsub
m2
```

警告

これを含め、O-word サブルーチンで M2 を使用しないでください。 見つけにくいエラーが発生します。 たとえば、サブルーチンで M2 を使用すると、サブルーチンが適切に終了せず、メインプログラムではなく、サブルーチン NGC ファイルが開いたままになります。

---

on_abort.ngc がインタープリターの検索パスに沿っていることを確認してください（NC_FILES ディレクトリが内部プロシージャで乱雑にならないように推奨される場所：SUBROUTINE_PATH）。 on_abort は、条件付きクリーンアップに使用される可能性のあるアボートプロシージャを呼び出す原因を示す単一のパラメータを受け取ります。

その手順のステートメントは、通常、HAL ピンが適切にリセットされるなど、中絶後の状態がクリーンアップされていることを保証します。 例については、configs / sim / axis / remap/rack-toolchange を参照してください。

エピローグから INTERP_ERROR を返すことによって再マップされたコードを終了すると（前のセクションを参照）、on_abort プロシージャも呼び出されることに注意してください。

6.10.5.8 エラー処理：再マップされたコード NGC プロシージャの失敗

ハンドラープロシージャでエラー状態が発生したと判断した場合は、M2 を使用してハンドラーを終了しないでください。上記を参照してください。

オペレーターのエラーメッセージを表示して現在のプログラムを停止するだけで十分な場合は、（abort、<message>）機能を使用して、ハンドラーをエラーメッセージで終了します。 この例のように、テキスト内の番号付き、名前付き、ini、および HAL パラメーターを置き換えることができることに注意してください（tests / interp / abort-hot-comment / test.ngc も参照）。

```
o100 if [..] (some error condition)
(abort, Bad Things! p42=#42 q=#<q> ini=#<_ini[a]x> pin=#<_hal[component.pin])
o100 endif
```

Note

ini および HAL 変数の拡張はオプションであり、INI ファイルで無効にすることができます

よりきめ細かい回復アクションが必要な場合は、前の例で説明したイディオムを使用してください。

- エラー状態を通知するだけの場合でも、エピローグ関数を定義します
- ハンドラーから負の値を渡してエラーを通知します
- エピローグ関数の戻り値を調べます。
- 必要な回復アクションを実行します
- ハンドラーからエラーメッセージ文字列を返します。これにより、インタープリターのエラーメッセージが設定され、プログラムが中止されます（（abort、message =

このエラーメッセージは UI に表示され、INTERP_ERROR を返すと、このエラーは他のランタイムエラーと同様に処理されます。

（abort、msg）とエピローグからの INTERP_ERROR の戻りの両方により、定義されている場合は ON_ABORT ハンドラーも呼び出されることに注意してください（前のセクションを参照）。

6.10.6 他の既存のコードの再マッピング：S、M0、M1、M60

1.1.1.1 自動ギア選択は S を再マッピングします（スピンドル速度を設定します）v

再マップされた S コードの潜在的な用途は、速度に応じた自動ギア選択です。 再マップ手順では、現在のギア設定を前提として達成可能な望ましい速度をテストし、そうでない場合はギアを適切に変更します。

6.10.6.1 M0、M1、M60 の動作を調整する

M0 / M1 を再マッピングするユースケースは、既存のコードの動作をカスタマイズすることです。たとえば、M0 または M1 プログラムの一時停止中にスピンドル、ミスト、フラッドをオフにし、プログラムが再開されたときにこれらの設定をオンに戻すことが望ましい場合があります。

それを行う完全な例については、上記のように M1 を適応させる configs / sim / axis / remap /extend-builtins/を参照してください。

6.10.7 新しい G コードサイクルの作成

ここで使用されている G コードサイクルは、次のように動作することを意味します。

- 最初の呼び出しで、関連する単語が収集され、G コードサイクルが実行されます。
- 後続の行がこのコードに適用可能なパラメータワードを継続しているが、新しい G コードがない場合は、それに応じてパラメータが変更された状態で前の G コードが再実行されます。

例：次の ini セグメントを持つ再マップされた G コードサイクルとして G84.3 が定義されていると仮定します（cycle_prolog および cycle_epilog の詳細な説明については、ここを参照してください）。

```
[RS274NGC]
# A cycle with an oword procedure: G84.3 <X- Y- Z- Q- P->
REMAP=G84.3 argspec=xyzabcuvwpr prolog=cycle_prolog ngc=g843 epilog=cycle_epilog
modalgroup  -
=1
```

次の行を実行します。

```
g17
(1) g84.3 x1 y2 z3 r1
(2) x3 y4 p2
(3) x6 y7 z5
(4) G80
```

次の原因になります（平面が XY であるため、R は粘着性があり、Z は粘着性があることに注意してください）。

1. g843.ngc は、単語 x = 1、y = 2、z = 3、r=1 で呼び出されます。
2. g843.ngc は、単語 x = 3、y = 4、z = 3、p = 2、r=1 で呼び出されます。
3. g843.ngc は、単語 x = 6、y = 7、z = 3、r=1 で呼び出されます。
4. G84.3 サイクルはキャンセルされます。

これは、新しいサイクルを作成するだけでなく、サイクルとして動作しない既存の G コードを再パッケージ化するための簡単な方法を提供します。 たとえば、G33.1 リジッドタッピングコードはサイクルとして動作しません。 このようなラッパーを使用すると、G33.1 を使用しながら、サイクルとして動作する新しいコードを簡単に作成できます。

この機能の完全な例については、configs / sim / axis / remap/cycle を参照してください。 これには 2 つのサイクルが含まれています。1 つは上記のような NGC プロシージャを使用し、もう 1 つは Python のみを使用したサイクルの例です。

6.10.8 組み込み Python の構成

Python プラグインは、そのように構成されている場合、インタープリターとタスクの両方を提供するため、ini ファイルに独自のセクション PYTHON があります。

1.1.1.1 Python プラグイン：ini ファイルの構成

```
[PYTHON]
```

TOPLEVEL=<filename>

起動時に実行する最初の Python スクリプトのファイル名。 このスクリプトは、パッケージ名の構造を設定する役割を果たします。以下を参照してください。

PATH_PREPEND=<directory>

このディレクトリの前に PYTHON_PATH を追加します。 繰り返しグループ。

PATH_APPEND=<directory>

このディレクトリを PYTHON_PATH に追加します。 繰り返しグループ。

LOG_LEVEL=<integer>

プラグイン関連のアクションのログレベル。 問題が疑われる場合は、これを増やしてください。 非常に冗長になる可能性があります。

RELOAD_ON_CHANGE=[0|1]

ファイルが変更された場合は、TOPLEVEL スクリプトをリロードします。 デバッグには便利ですが、現在、実行時のオーバーヘッドが発生します。 本番構成ではこれをオフにします。

PYTHON_TASK=[0|1]

Python タスクプラグインを起動します。実験的です。 xxx を参照してください。

6.10.8.1 インタプリタからの Python ステートメントの実行

コマンドのアドホック実行のために、Python ホットコメントが追加されました。 Python の出力はデフォルトで stdout に送られるため、結果を表示するには、ターミナルウィンドウから LinuxCNC を起動する必要があります。 例（例：MDI ウィンドウ内）：

```
;py,print 2*3
```

インタープリターインスタンスはここでは self として使用できるため、次のコマンドを実行することもできます。

```
;py,print self.tool_table[0].toolno
```

emcStatus 構造にもアクセスできます。

```
;py,from emctask import *
;py,print emcstat.io.aux.estop
```

6.10.9 RS274NGC インタープリターでの組み込み Python のプログラミング

1.1.1.1 Python プラグインの名前空間

名前空間は次のように配置されることが期待されます。

oword

このモジュールの呼び出し可能オブジェクトはすべて、PythonO-word プロシージャの候補です。 同じ名前の NGC プロシージャをテストする前に、Python oword モジュールがチェックされることに注意してください。実際、oword の名前は、同じベース名の NGC ファイルを非表示にします。

remap

argspec prolog、epilog、または python オプションで参照される Python 呼び出し可能オブジェクトは、ここにあると予想されます。

namedparams

このモジュールの Python 関数は、事前定義された名前付きパラメーターの名前空間を拡張または再定義します。事前定義されたパラメーターの追加を参照してください。

task

ここでは、タスク関連の呼び出し可能オブジェクトが期待されます。

6.10.9.1 Python から見たインタプリタ

インタプリタは、src / emc /rs274ngc で定義されている既存の C++クラス（Interp）です。 概念的には、すべての oword。<function>および remap。<function> Python 呼び出しは、この Interp クラスのメソッドですが、このクラスの明示的な Python 定義はなく（Boost.Python ラッパーインスタンスです）、最初のパラメーターとしてを受け取ります。 内部にアクセスするために使用できます。

6.10.9.2 インタープリター__init__および__delete__関数

TOPLEVEL モジュールが関数__init__を定義している場合、インタープリターが完全に構成されると呼び出されます（ini ファイルが読み取られ、状態がワールドモデルと同期されます）。

TOPLEVEL モジュールが関数__delete__を定義している場合、インタプリタがシャットダウンされる前、および永続パラメータが PARAMETER_FILE に保存された後に 1 回呼び出されます。

注_現時点では、__ delete__ハンドラーは、gcode モジュールのインポートによって作成されたインタープリターインスタンスでは機能しません。 そこで同等の機能が必要な場合（ほとんどありません）、Pythonatexit モジュールを検討してください。

```
# this would be defined in the TOPLEVEL module
def __init__(self):
    # add any one-time initialization here
    if self.task:
    # this is the milltask instance of interp
        pass
    else:
        # this is a non-milltask instance of interp
        pass

def __delete__(self):
    # add any cleanup/state saving actions here
    if self.task: # as above
        pass
    else:
        pass
```

この関数は、たとえば remap または oword 関数で後で必要になる可能性のある、Python 側の属性を初期化し、PARAMETER_FILE が提供する範囲を超えて状態を保存または復元するために使用できます。

ミルタスクインタープリターインスタンスでのみ発生するセットアップまたはクリーンアップアクションがある場合（gcode Python モジュールにあり、プレビュー/進行状況の表示目的を提供するインタープリターインスタンスとは対照的に）、これは評価することでテストできます。 self.task。

__init__と__delete__の使用例は、configs / sim / axis / remap / cycle / python / toplevel.py にあり、ncfiles / remap_lib / python-stdglue / stdglue.py でサイクルを処理するために必要な属性を初期化します（そして configs/にインポートします）。 sim / axis / remap / cycle / ython / remap.py）。

6.10.9.3 呼び出し規約：NGC から Python

Python コードは、次の状況で NGC から呼び出されます。

- 通常のプログラム実行中：

- O <proc>呼び出しのような O-word 呼び出しが実行され、名前 oword.proc が定義され、呼び出し可能である場合
- ; py、<Python ステートメント>のようなコメントが実行されたとき

- 再マップされたコードの実行中：任意の prolog =、python =、epilog=ハンドラー。

O-wordPython サブルーチンの呼び出し

引数：

self

インタプリタインスタンス

*args

実際の位置パラメータのリスト。 実際のパラメーターの数は異なる場合があるため、次のスタイルの宣言を使用するのが最適です。

```
# this would be defined in the oword module
def mysub(self, *args):
print "number of parameters passed:", len(args)
for a in args:
print a
```

O-word Python サブルーチンの戻り値 NGC プロシージャが値を返すのと同じように、O-wordPython サブルーチンも戻ります。 彼らは次のいずれかを期待されています：

- 値を返さない（return ステートメントまたは値なし）
- float または int 値
- 文字列。これはエラーメッセージであることを意味します。プログラムを中止してください。（abort、msg）のように機能します。

その他の戻り値タイプでは、Python 例外が発生します。

呼び出し元の NGC 環境では、次の事前定義された名前付きパラメーターを使用できます。

#<_value>

最後に呼び出されたプロシージャによって返される値。 起動時に 0.0 に初期化されます。 Interp で self.return_value（float）として公開されます。

#<_value_returned>

最後に呼び出されたプロシージャが明示的な値で return`または`endsub を返したことを示します。 true の場合は 1.0。 呼び出しごとに 0.0 に設定します。 Interp で公開されたのは self.value_returned（int）でした。

例については、tests / interp/value-returned も参照してください。

prolog=および epilog=サブルーチンの呼び出し規約引数は次のとおりです。

self

インタプリタインスタンス

words

キーワードパラメータ辞書。 argspec が存在する場合、単語はそれに応じて現在のブロックから収集され、便宜上辞書に渡されます（単語は呼び出し元のブロックから直接取得することもできますが、これにはインタープリターの内部に関する知識が必要です）。 argspec が渡されなかった場合、またはオプションの値のみが指定され、呼び出し元のブロックにこれらの値が存在しなかった場合、この dict は空です。 単語名は小文字に変換されます。

呼び出し例：

```
def minimal_prolog(self, **words): # in remap module
print len(words)," words passed"
for w in words:
print "%s: %s" % (w, words[w])
if words['p'] < 78: # NB: could raise an exception if p were optional
return "failing miserably"
return INTERP_OK
```

戻り値：

INTERP_OK

成功したらこれを返します。 これをインタプリタからインポートする必要があります。

"a message text"

ハンドラーから文字列を返すということは、これがエラーメッセージであることを意味し、プログラムを中止します。 （abort、msg）のように機能します。

python=サブルーチンの呼び出し規約引数は次のとおりです。

self

インタプリタインスタンス

words

キーワードパラメータ辞書。 プロローグおよびエピローグと同じ kwargs 辞書（上記を参照）。

最小の python=関数の例：

```
def useless(self, **words): # in remap module
return INTERP_OK
```

戻り値：

INTERP_OK

成功したらこれを返す

"a message text"

ハンドラーから文字列を返すということは、これがエラーメッセージであることを意味し、プログラムを中止します。 （abort、msg）のように機能します。

ハンドラーがキューバスター操作（ツールの変更、プローブ、HAL ピンの読み取り）を実行する必要がある場合、次のステートメントで実行を一時停止することになっています。

INTERP_EXECUTE_FINISH を生成します

これは、先読みを停止し、キューに入れられたすべての操作を実行し、キューバスター操作を実行し、インタープリターの状態をマシンの状態と同期させてから、インタープリターに続行するように信号を送るタスクに信号を送ります。 この時点で、yield..ステートメントに続くステートメントで関数が再開されます。

キューバスターの処理：プローブ、ツールの変更、および HAL ピンの待機キューバスターは、そのような操作が呼び出された時点でプロシージャを中断するため、インタープリター synch（）の後でプロシージャを再起動する必要があります。 これが発生した場合、プロシージャは、再起動され

たかどうか、およびどこから続行するかを知る必要があります。 Python ジェネレーターメソッドは、プロシージャの再起動を処理するために使用されます。

これは、単一のリスタートポイントでの呼び出しの継続を示しています。

```
def read_pin(self,*args):
# wait 5secs for digital-input 00 to go high
emccanon.WAIT(0,1,2,5.0)
# cede control after executing the queue buster:
yield INTERP_EXECUTE_FINISH
# post-sync() execution resumes here:
pin_status = emccanon.GET_EXTERNAL_DIGITAL_INPUT(0,0);
print "pin status=",pin_status
```

注意

歩留まり機能は壊れやすいです。 以下の制限は、yieldINTERP_EXECUTE_FINISH の使用に適用されます。

- イールド INTERP_EXECUTE_FINISH を実行する Python コードは、リマッププロシージャの一部である必要があります。 イールドは Python の oword プロシージャでは機能しません。
- 通常の Python の yield ステートメントとは異なり、yieldINTERP_EXECUTE_FINISH ステートメントを含む Python リマップサブルーチンは値を返さない場合があります。
- イールドに続くコードは、self.execute（ "<mdi command>"）のように、インタープリターを再帰的に呼び出さない場合があります。 これはインタプリタのアーキテクチャ上の制限であり、大幅な再設計なしでは修正できません。

6.10.9.4 呼び出し規約：Python から NGC

NGC コードは、次の場合に Python から実行されます。

- メソッド self.execute（<NGC コード> [、<行番号>]）が実行されます
- 再マップされたコードの実行中に、prolog =関数が定義されている場合、ngc=で指定された NGC プロシージャがその直後に実行されます。

プロローグハンドラーはハンドラーを呼び出しませんが、たとえば事前定義されたローカルパラメーターを設定することにより、呼び出し環境を準備します。

パラメータをプロローグに挿入し、エピローグで取得する概念的には、プロローグとエピローグは、O-word プロシージャと同じ呼び出しレベルで実行されます。つまり、サブルーチン呼び出しが設定された後、サブルーチン endsub または return の前に実行されます。

これは、プロローグで作成されたローカル変数が O-word プロシージャのローカル変数になり、エピローグの実行時に O-word プロシージャで作成されたローカル変数に引き続きアクセスできることを意味します。

self.params 配列は、番号付きおよび名前付きパラメーターの読み取りと設定を処理します。 名前付きパラメーターが_（アンダースコア）で始まる場合、それはグローバルパラメーターであると見なされます。 そうでない場合は、呼び出し元のプロシージャに対してローカルです。 また、1..30 の範囲の番号付きパラメーターは、ローカル変数のように扱われます。 元の値は、O-word プロシージャからの return/endsub で復元されます。

これは、O-word プロシージャへの、または O-word プロシージャからのパラメータの挿入と抽出を示す再マップされたコードの例です。

```
REMAP=m300 prolog=insert_param ngc=testparam epilog=retrieve_param modalgroup=10
```

```
def insert_param(self, **words): # in the remap module
print "insert_param call level=",self.call_level
self.params["myname"] = 123
self.params[1] = 345
self.params[2] = 678
return INTERP_OK
def retrieve_param(self, **words):
print "retrieve_param call level=",self.call_level
print "#1=", self.params[1]
print "#2=", self.params[2]
try:
print "result=", self.params["result"]
except Exception,e:
return "testparam forgot to assign #<result>"
return INTERP_OK
```

```
o<testparam> sub
(debug, call_level=#<_call_level> myname=#<myname>)
; try commenting out the next line and run again
#<result> = [#<myname> * 3]
#1 = [#1 * 5]
#2 = [#2 * 3]
o<testparam> endsub
m2
```

self.params（）は、現在定義されているすべての変数名のリストを返します。 myname はローカルであるため、エピローグが終了すると消えます。

Python からのインタープリターの呼び出し次のように、Python コードからインタープリターを再帰的に呼び出すことができます。

```
self.execute(<NGC code>[,<line number>])
```

例：

```
self.execute("G1 X%f Y%f" % (x,y))
self.execute("O <myprocedure> call", currentline)
```

戻り値が<INTERP_MIN_ERROR であることをテストすることをお勧めします。 多数の execute（）ステートメントを使用している場合は、以下のように InterpreterException をトラップする方がおそらく簡単です。

警告

この場合、前のセクションで説明したパラメーターの挿入/取得方法は機能しません。 単純な NGC コマンドまたはプロシージャ呼び出しとプロシージャへの高度なイントロスペクションを実行するだけで十分であり、ローカルの名前付きパラメータを渡す必要はありません。 再帰呼び出し機能は壊れやすいです。

---

execute（）中のインタープリター例外 interpreter.throw_exceptions がゼロ以外（デフォルトは 1）で、self.execute（）がエラーを返すと、例外 InterpreterException が発生します。 InterpreterException には次の属性があります。

line_number

エラーが発生した場所

line_text

エラーの原因となる NGC ステートメント

error_message

通訳者のエラーメッセージ

エラーは、次の Python の方法でトラップできます。

```
import interpreter
interpreter.throw_exceptions = 1
...
try:
self.execute("G3456") # raise InterpreterException
```

```
except InterpreterException,e:
msg = "%d: '%s' - %s" % (e.line_number,e.line_text, e.error_message)
return msg # replace builtin error message
```

キヤノンキヤノンレイヤーは事実上すべて無料の機能です。 例：

```
import emccanon
def example(self,*args):
....
emccanon.STRAIGHT_TRAVERSE(line,x0,y0,z0,0,0,0,0,0,0)
emccanon.STRAIGHT_FEED(line,x1,y1,z1,0,0,0,0,0,0)
...
return INTERP_OK
```

実際の canon 関数は、src / emc / nml_intf / canon.hh で宣言され、src / emc / task/emccanon.cc に実装されています。 Python 関数の実装は、src / emc / rs274ncg/canonmodule.cc にあります。

6.10.9.5 内蔵モジュール

次のモジュールが組み込まれています。

interpreter

: Interp クラスの内部を公開します。 src / emc / rs274ngc / interpmodule.cc、および tests / remap/introspect 回帰テストを参照してください。

emccanon

: src / emc / task/emccanon.cc のほとんどの呼び出しを公開します。

emctask

: emcStatus クラスインスタンスを公開します。 src / emc / task/taskmodule.cc を参照してください。 ユーザーインターフェイスに使用される gcode モジュールを使用する場合は存在しません。インタープリターの milltask インスタンスにのみ存在します。

6.10.10 事前定義された名前付きパラメーターの追加

インタープリターには、NGC 言語レベルから内部状態にアクセスするための事前定義された名前付きパラメーターのセットが付属しています。 これらのパラメータは読み取り専用でグローバルであるため、に割り当てることはできません。

namedparams モジュールで関数を定義することにより、追加のパラメーターを追加できます。 関数の名前は、新しい定義済みの名前付きパラメーターの名前を定義します。これは、任意の式で参照できるようになりました。

名前付きパラメーターを追加または再定義するには、次のようにします。

- インタプリタが見つけられるように namedparams モジュールを追加します
- 関数ごとに新しいパラメータを定義します（以下を参照）。 これらの関数は、パラメーターとして self（インタープリターインスタンス）を受け取るため、aribtrary 状態にアクセスできます。 任意の Python 機能を使用して値を返すことができます。
- TOPLEVEL スクリプトからそのモジュールをインポートします

```
# namedparams.py
# trivial example
def _pi(self):
return 3.1415926535
```

```
#<circumference> = [2 * #<radius> * #<_pi>]
```

namedparams.py の関数は、float または int 値を返すことが期待されています。 文字列が返された場合、これによりインタープリターのエラーメッセージが設定され、実行が中止されます。

これはグローバルの RS274NGC 規則であるため、先頭にアンダースコアが付いた関数がパラメーターとして追加されます。

namedparams モジュールに同じ名前の Python 関数を追加することにより、既存の事前定義されたパラメーターを再定義することができます。 この場合、起動時に警告が発生します。

上記の例はそれほど有用ではありませんが、ほとんどすべてのインタープリターの内部状態に Python からアクセスできるため、この方法で任意の述語を定義できることに注意してください。 もう少し高度な例については、tests / remap/predefined-named-params を参照してください。

6.10.11 標準の接着剤ルーチン

多くの再マッピングタスクは非常に似ているため、単一の Python モジュールで作業中のプロローグルーチンとエピローグルーチンの収集を開始しました。 これらは現在、ncfiles / remap_lib / python-stdglue / stdglue.py にあり、次のルーチンを提供します。

1.1.1.1 T：prepare_prolog と prepare_epilog

これらは、TxToolPrepare の NGC プロシージャをラップします。

prepare_prolog のアクション以下のパラメーターが NGC プロシージャーに表示されます。

- ＃<tool>-T ワードのパラメータ
- ＃<ポケット>-対応するポケット

ツール番号 0 が要求された場合（ツールのアンロードを意味します）、対応するポケットは-1 として渡されます。

次の場合はエラーになります。

- T パラメータとして工具番号が指定されていません
- ツールがツールテーブルに見つかりません。

[EMCIO] RANDOM_TOOLCHANGER = 1 パラメータを設定しない限り、工具とポケット番号は同じであり、工具テーブルのポケット番号は無視されることに注意してください。 これは現在制限です。

P R E PA RE_EPILOG のアクション

- NGC プロシージャは正の値を返すことが期待されます。そうでない場合、戻り値を含むエラーメッセージが表示され、インタプリタは中止されます。
- NGC プロシージャが T コマンド（組み込みの T 動作を参照）を実行した場合、それ以上のアクションは実行されません。 これは、たとえば、他のステートメントの前後にある組み込みの動作を最小限に調整するために使用できます。
- それ以外の場合、＃<tool>および＃<pocket>パラメーターはサブルーチンのパラメーター空間から抽出されます。 これは、NGC プロシージャがこれらの値を変更する可能性があり、エピローグが変更された値を考慮に入れることを意味します。
- 次に、Canon コマンド SELECT_POCKET（＃<pocket>、＃<tool>）が実行されます。

6.10.11.1 M6：change_prolog および change_epilog

これらは、M6 ツール変更の NGC 手順をラップします。

C H A NG E_PROLOG のアクション

- 次の 3 つの手順は、iocontrol-v2 コンポーネントが使用されている場合にのみ適用できます。
  - パラメーター 5600（障害インジケーター）がゼロより大きい場合、これは Toolchanger の障害を示し、次のように処理されます。
  - パラメータ 5601（エラーコード）が負の場合、これはハードフォールトを示し、プロローグはエラーメッセージで中止されます。
  - パラメータ 5601（エラーコード）がゼロより大きい場合、これはソフト障害を示します。 情報メッセージが表示され、プロローグが続行されます。
- ポケットが選択される原因となる先行する T コマンドがなかった場合、プロローグはエラーメッセージで中止されます。

- カッター半径補正がオンの場合、プロローグはエラーメッセージで中止されます。

次に、次のパラメータが NGC プロシージャにエクスポートされます。

- ＃<tool_in_spindle>：現在ロードされているツールのツール番号
- ＃<selected_tool>：選択されたツール番号
- ＃<selected_pocket>：選択したツールのポケット番号

C H AN GELILONG のアクション

- NGC プロシージャは正の値を返すことが期待されます。そうでない場合、戻り値を含むエラーメッセージが表示され、インタプリタは中止されます。
- パラメーター 5600（障害インジケーター）がゼロより大きい場合、これは Toolchanger 障害を示し、次のように処理されます（iocontrol-v2 のみ）。
  - パラメータ 5601（エラーコード）が負の場合、これはハードフォールトを示し、エピローグはエラーメッセージで中止されます。
  - パラメータ 5601（エラーコード）がゼロより大きい場合、これはソフト障害を示します。 情報メッセージが表示され、エピローグが続行されます。
- NGC プロシージャが M6 コマンド（組み込みの M6 動作を参照）を実行した場合、それ以上のアクションは実行されません。 これは、たとえば、他のステートメントの前後にある組み込みの動作を最小限に調整するために使用できます。
- それ以外の場合、＃<selected_pocket>パラメーターは、サブルーチンのパラメータースペースから抽出され、インタープリターの current_pocket 変数を設定するために使用されます。 この場合も、プロシージャはこの値を変更する可能性があり、エピローグは変更された値を考慮に入れます。
- 次に、Canon コマンド CHANGE_TOOL（＃<selected_pocket>）が実行されます。
- 新しい工具パラメータ（オフセット、直径など）が設定されます。

6.10.11.2 G コードサイクル：cycle_prolog および cycle_epilog

これらは NGC プロシージャをラップするため、サイクルとして機能できます。つまり、実行が終了した後もモーションコードが保持されます。 次の行にパラメータワード（新しい X、Y 値など）だけが含まれている場合は、最初の呼び出しで指定されたパラメータのセットにマージされた新しいパラメータワードを使用してコードが再度実行されます。

これらのルーチンは、argspec=<words>パラメーターと連動して機能するように設計されています。これは簡単に使用できますが、現実的なシナリオでは、argspec を回避し、より適切なエラーメッセージを表示するために、ブロックを手動でより徹底的に調査します。

推奨される argspec は次のとおりです。

```
REMAP=G<somecode> argspec=xyzabcuvwqplr prolog=cycle_prolog ngc=<ngc procedure>
epilog=  -
cycle_epilog modalgroup=1
```

これにより、cycle_prolog は、ブロック内の軸ワードの互換性を判断できます。以下を参照してください。

C Y CL E_PROLOG のアクション

- 現在のブロックから渡された単語が、CannedCycleErrors で概説されている条件を満たすかどうかを判断します。
  - 軸の単語を<x>、＃<y>などとしてエクスポートします。 異なるグループ（XYZ）（UVW）の軸ワードが一緒に使用されている場合、または（ABC）のいずれかが指定されている場合は、失敗します。
  - L-を＃<l>としてエクスポートします。 指定しない場合、デフォルトは 1 です。
  - P-を＃<p>としてエクスポートします。 p が 0 未満の場合は失敗します。
  - R-を＃<r>としてエクスポートします。 r が指定されていない場合は失敗し、指定されている場合は 0 以下になります。
  - 送り速度がゼロの場合、または逆時間送りまたはカッター補正がオンの場合は失敗します。
- これがサイクル G コードの最初の呼び出しであるかどうかを判別します（そうである場合）。
  - （argspec に従って）渡された単語をスティッキーパラメーターのセットに追加します。これは、複数の呼び出しにわたって保持されます。
- そうでない場合（新しいパラメーターを含む継続行）：
  - merge the words passed in into the existing set of sticky parameters.
- スティッキーパラメータのセットを NGC プロシージャにエクスポートします。

C Y C LE_EPILOG のアクション

- 現在のコードが実際にサイクルであったかどうかを判断します。そうである場合は、次のようにします。
  - 現在のモーションモードを保持して、モーションコードのない継続行が同じモーションコードを実行するようにします。

6.10.11.3 S（速度の設定）：setspeed_prolog および setspeed_epilog

TBD

6.10.11.4 F（フィードの設定）：setfeed_prolog および setfeed_epilog

TBD

6.10.11.5 M61 ツール番号の設定：settool_prolog および settool_epilog

TBD

6.10.12 再マップされたコードの実行

1.1.1.1 再マップ中の NGC プロシージャー呼び出し環境

通常、O-word プロシージャは、位置パラメータを使用して呼び出されます。 このスキームは、特にオプションのパラメータが存在する場合、非常に制限されます。 したがって、呼び出し規約は、Python キーワード引数モデルにリモートで類似したものを使用するように拡張されました。

gcode / main サブルーチンへのリンクを参照してください：sub、end sub、return、call。

6.10.12.1 ネストされた再マップされたコード

再マップされたコードは、プロシージャ呼び出しと同じようにネストできます。つまり、NGC プロシージャが他の再マップされたコードを参照している再マップされたコードは正しく実行されます。

ネストレベルの再マップの最大数は現在 10 です。

6.10.12.2 再マップ中のシーケンス番号

シーケンス番号は、O ワード呼び出しの場合と同様に伝播および復元されます。 回帰テストについては、tests / remap / nested-remaps / word を参照してください。これは、3 レベルの深さのネストされたリマップ中のシーケンス番号の追跡を示しています。

6.10.12.3 デバッグフラグ

次のフラグは、再マッピングと Python 関連の実行に関連しています。

EMC_DEBUG_OWORD	0x00002000 traces execution of O-word subroutines
EMC_DEBUG_REMAP	0x00004000 traces execution of remap-related code
EMC_DEBUG_PYTHON	0x00008000 calls to the Python plug in
EMC_DEBUG_NAMEDPARAM	0x00010000 trace named parameter access
EMC_DEBUG_PYTHON_TASK	0x00040000 trace the task Python plug in
EMC_DEBUG_USER1	0x10000000 user-defined - not interpreted by LinuxCNC
EMC_DEBUG_USER2	0x20000000 user-defined - not interpreted by LinuxCNC

または、必要に応じて、これらのフラグを[EMC]DEBUG 変数に追加します。 デバッグフラグの現在のリストについては、src / emc / nml_intf/debugflags.h を参照してください。

6.10.12.4 埋め込まれた Python コードのデバッグ

埋め込まれた Python コードのデバッグは、通常の Python スクリプトのデバッグよりも難しく、限られた数のデバッガーしか存在しません。 実用的なオープンソースベースのソリューションは、Eclipse IDE、PydDevEclipse プラグインおよびそのリモートデバッグ機能を使用することです。

このアプローチを使用するには：

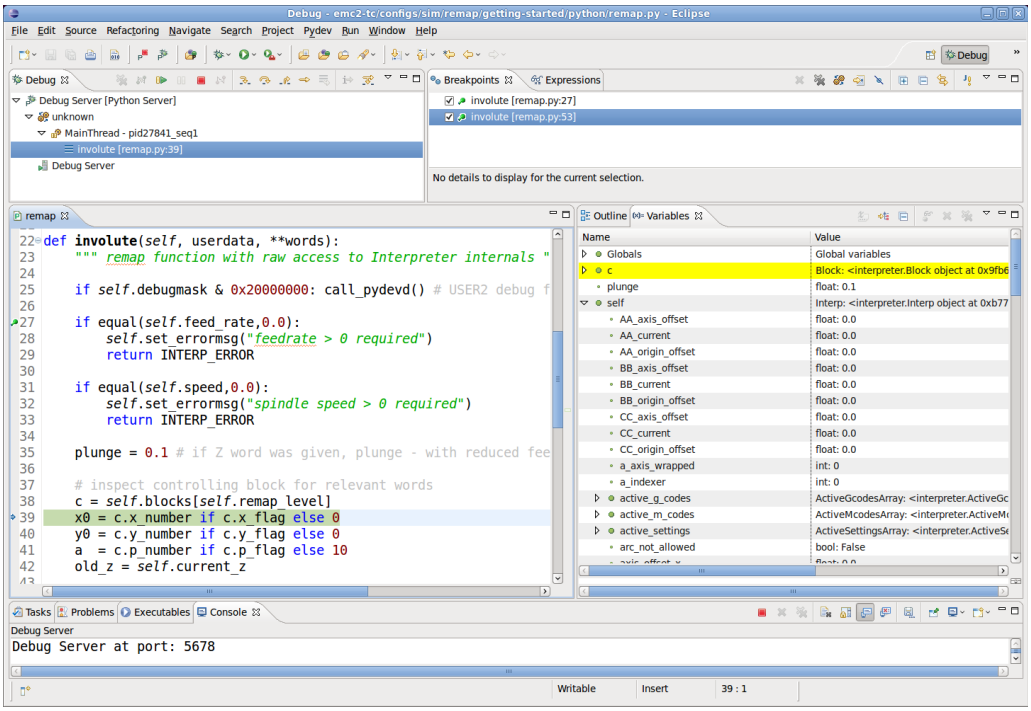
- Ubuntu ソフトウェアセンターから Eclipse をインストールします（最初の選択を選択してください）
- Pydev アップデートサイトから PyDev プラグインをインストールします
- LinuxCNC ソースツリーを Eclipse プロジェクトとしてセットアップする
- Eclipse で Pydev デバッグサーバーを起動します
- 埋め込まれた Python コードが、そのプラグインに付属している pydevd.py モジュールを見つけることができることを確認してください。これは、Eclipse インストールディレクトリの下のどこかに埋め込まれています。 このディレクトリの場所を反映するように、util.py の pydevd 変数を設定します。
- Python モジュールに pydevd をインポートします-util.py と remap.py の例を参照してください
- ある時点でモジュールで pydevd.settrace（）を呼び出して、Eclipse Python デバッグサーバーに接続します。ここでは、通常どおり、コードにブレークポイントを設定したり、変数を調べたり、ステップを実行したりできます。

警告

Eclipse と Pydev デバッグサーバーが起動されていない場合、pydevd.settrace（）は実行をブロックします。

最後の 2 つのステップをカバーするために：o <pydevd>プロシージャは、MDI モードからデバッガーに入るのに役立ちます。 ブレークポイントを設定するには、util.py の call_pydevd 関数と remap.involute の使用法も参照してください。

これは、上からインボリュート手順をデバッグする Eclipse/PyDevd のスクリーンショットです。



詳細については、configs / sim / axis / remap / getting-started/python の Python コードを参照してください。

6.10.13 Axis プレビューと再マップされたコードの実行

再マップされたコードのツールパスを完全にプレビューするには、いくつかの予防措置を講じる必要があります。 何が起こっているのかを理解するために、プレビューと実行のプロセスを確認しましょう（これは Axis のケースをカバーしていますが、他のケースも同様です）。

まず、2 つの独立したインタプリタインスタンスが関係していることに注意してください。

- スタートボタンを押すとプログラムを実行し、実際にマシンを動かすミルタスクプログラムの 1 つのインスタンス
- ツールパスプレビューを生成することを主な目的とするユーザーインターフェイスの 2 番目のインスタンス。 これは、ロードされるとプログラムを実行しますが、実際にはマシンの動きを引き起こしません。

ここで、再マップ手順に G38 プローブ操作が含まれていると仮定します。たとえば、自動工具長タッチオフによる工具交換の一部としてです。 プローブが失敗した場合、それは明らかにエラーになるため、メッセージを表示してプログラムを中止します。

では、この手順のプレビューはどうですか？ もちろん、プレビュー時には、プローブが成功したか失敗したかはわかりませんが、プローブの最大深度を確認し、成功したと想定して実行を継続し、さらなる動きをプレビューすることをお勧めします。 また、プローブ失敗メッセージを表示してプレビュー中に中止しても意味がありません。

この問題に対処する方法は、プロシージャがプレビューモードで実行されるか実行モードで実行されるかをプロシージャでテストすることです。 これは、#<_ task>の事前定義された名前付きパラ

メーターをテストすることで確認できます。実際の実行中は1、プレビュー中は0になります。 完全な使用例については、configs / sim / axis / remap / manual-toolchange-with-tool-length-switch / nc_subroutines/manual_change.ngc を参照してください。

Embedded Python 内では、self.task をテストすることでタスクインスタンスをチェックできます。これは、milltask インスタンスでは1、プレビューインスタンスでは0になります。

6.10.14 再マップ可能なコード

1.1.1.1 再マッピング可能な既存のコード

再定義できる既存のコードの現在のセットは次のとおりです。

- Tx（準備）
- M6（変更ツール）
- M61（工具番号設定）
- M0（実行中のプログラムを一時停止します）
- M1（オプションの停止スイッチがオンの場合は実行中のプログラムを一時停止します）
- M60（パレットシャトルを交換してから、実行中のプログラムを一時的に一時停止します）
- S（設定スピンドル速度）
- F（セットフィード）

現在、M61 を使用するには、iocontrol-v2 を使用する必要があることに注意してください。

6.10.14.1 現在割り当てられていない G コード：

現在割り当てられていない G コード（再マッピング用）は、次の表の空白の領域から選択する必要があります。 リストされているすべての G コードは、LinuxCNC の現在の実装ですでに定義されており、新しい G コードを再マップするために使用することはできません。 （LinuxCNC に新しい G コードを追加する開発者は、これらのテーブルにも新しい G コードを追加することをお勧めします。）

表 8.2：Table_of_Allocated_G-codes_00-09

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
00	G00									
01	G01									
02	G02									
03	G03									
04	G04									
05	G05	G05.1	G05.2	G05.3						
06										
07	G07									
08	G08									
09										

表 8.5：Table_of_Allocated_G-codes_30-39

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
30	G30	G30.1								
31										
32										
33	G30	G30.1								
34										
35										
36										
37										
38										
39										

表 8.6：Table_of_Allocated_G-codes_40-49

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
40	G40									
41	G41	G41.1								
42	G42	G42.1								
43	G43	G43.1								
44										
45										
46										
47										
48										
49	G40									

表 8.7：Table_of_Allocated_G-codes_50-59

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
50										
51										
52										
53	G53									
54	G54									
55	G55									
56	G56									
57	G57									
58	G58									
59	G59	G59.1	G59.2	G59.3						

表 8.8：Table_of_Allocated_G-codes_60-69

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
60	G60									
61	G61	G61.1								
62										
63										
64	G64									
65										
66										
67										
68										
69										

表 8.9：Table_of_Allocated_G-codes_70-79

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
70										
71										
72										
73										
74										
75										
76	G76									
77										
78										
79										

表 8.10：Table_of_Allocated_G-codes_80-89

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
80	G80									
81	G81									
82	G82									
83	G83									
84	G84									
85	G85									
86	G86									
87	G87									
88	G88									
89	G89									

表 8.11：Table_of_Allocated_G-codes_90-99

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
90	G90	G90.1								
91	G91	G91.1								
92	G92	G92.1	G92.2	G92.3						
93	G93									
94	G94									
95	G95									
96	G96									
97	G97									
98	G98									
99	G99									

6.10.14.2 現在割り当てられていない M コード：

これらの M コードは、現在 LinuxCNC の現在の実装では定義されておらず、新しい M コードを定義するために使用される可能性があります。（LinuxCNC で新しい M コードを定義する開発者は、このテーブルからそれらを削除することをお勧めします。）

表 8.12：Table_of_Unallocated_M-codes_00-99

#	Mx0	Mx1	Mx2	Mx3	Mx4	Mx5	Mx6	Mx7	Mx8	Mx9
00-09										
10-19	M10	M11	M12	M13	M14	M15	M16	M17	M18	
20-29	M20	M21	M22	M23	M24	M25	M26	M27	M28	M29
30-39		M31	M32	M33	M34	M35	M36	M37	M38	M39
40-49	M40	M41	M42	M43	M44	M45	M46	M47		
50-59					M54	M55	M56	M57	M58	M59
60-69										
70-79					M74	M75	M76	M77	M78	M79
80-89	M80	M81	M82	M83	M84	M85	M86	M87	M88	M89
90-99	M90	M91	M92	M93	M94	M95	M96	M97	M98	M99

M100 から M199 までのすべての M コードは、すでにユーザー定義の M コードであり、再マップしないでください。

M200 から M999 までのすべての M コードは、再マッピングに使用できます。

6.10.14.3 先読み時間と実行時間

foo

6.10.14.4 プラグイン/ピクルスハック

foo

6.10.14.5 モジュール、メソッド、クラスなどのリファレンス

foo

6.10.15 はじめに：タスク実行の拡張

foo

1.1.1.1 なぜタスク実行を変更したいのですか？

foo

6.10.15.1 ダイアグラム：タスク、interp、iocontrol、UI（??）

foo

6.10.16 タスク実行のモデル

foo

1.1.1.1 従来の iocontrol/iocontrolv2 の実行

foo

6.10.16.1 IO 手順の再定義

foo

6.10.16.2 実行時の Python プロシージャ

foo

6.10.17 LinuxCNC プログラム実行の簡単な調査

コードの再マッピングを理解するには、再マッピングに関連する限り、タスクとインタープリターの実行を調査することが役立つ場合があります。

1.1.1.1 インタープリターの状態

概念的には、インタプリタの状態は、次のカテゴリに分類される変数で構成されます。

1. 構成情報（通常は INI ファイルから）
2. ワールドモデル-実際のマシンの状態の表現
3. モーダル状態と設定
4. インタプリタの実行状態

（3）は、個々の NGC コードの実行間で繰り越される状態を指します。たとえば、スピンドルがオンになり、速度が設定されると、オフになるまでこの設定のままになります。 同じことが、フィード、ユニット、モーションモード（フィードまたは高速）などの多くのコードにも当てはまります。

（4）サブルーチン、インタープリター変数など、現在実行されているブロックに関する情報を保持します。この状態のほとんどは、かなり非体系的な構造体_setup に集約されます（interp_internals.hh を参照）。

6.10.17.1 タスクとインタプリタの相互作用、キューイングと先読み

LinuxCNC のタスク部分は、実際のマシンコマンド（移動、HAL インタラクションなど）の調整を担当します。 それ自体は RS274NGC 言語を処理しません。 これを行うために、タスクはインタプリタを呼び出して、MDI または現在のファイルのいずれかから次のコマンドを解析して実行します。

インタプリタの実行は、実際に何かを動かす正規のマシン操作を生成します。 ただし、これらはすぐには実行されず、キューに入れられます。 これらのコードの実際の実行は、LinuxCNC のタスク部分で行われます。canon コマンドはそのインタープリターキューからプルされ、実行されて実際のマシンが移動します。

これは、通常、インタプリタが実際のコマンドの実行よりもはるかに進んでいることを意味します。プログラムの解析は、目立った動きが始まる前に終了する可能性があります。 この動作は先読みと呼ばれます。

6.10.17.2 機械位置の予測

先読み中に正規のマシン操作を事前に計算するには、インタープリターが Gcode の各行の後にマシンの位置を予測できる必要がありますが、これが常に可能であるとは限りません。

相対移動を行う簡単なサンプルプログラム（G91）を見て、マシンが x = 0、y = 0、z=0 で起動すると仮定します。 相対的な動きは、次の動きの結果が前の動きの位置に依存することを意味します。

```
N10 G91
N20 G0 X10 Y-5 Z20
N30 G1 Y20 Z-5
N40 G0 Z30
N50 M2
```

ここで、通訳者は各ラインの機械位置を明確に予測できます。

N20 の後：x = 10 y = -5 z = 20; N30 の後：x = 10 y = 15 z = 15; N40 の後：x = 10 y = 15 z = 45 なので、プログラム全体を解析して、かなり前もって正規の操作を生成できます。

6.10.17.3 キューバスターは位置予測を破る

ただし、完全な先読みは、インタプリタがプログラムのすべての行の位置への影響を事前に予測できる場合にのみ可能です。 変更された例を見てみましょう。

```
N10 G91
N20 G0 X10 Y-5 Z20
N30 G38.3 Z-10
N40 O100 if [#5070 EQ 0]
N50 G1 Y20 Z-5
N60 O100 else
```

```
N70 G0 Z30
N80 O100 endif
N90 G1 Z10
N95 M2
```

N90 で移動を事前に計算するには、通訳者はマシンが行 N80 の後にある場所を知る必要があります。これは、プローブコマンドが成功したかどうかによって異なります。これは、実際に実行されるまでわかりません。

そのため、一部の操作はそれ以上の先読みと互換性がありません。 これらはキューバスターと呼ばれ、次のとおりです。

- M66 で HAL ピンの値を読み取る：HAL ピンの値は予測できません
- M6 で新しい工具をロードする：工具形状は予測できません
- G38.n でプローブを実行する：最終的な位置と成功/失敗は予測できません

6.10.17.4 キューバスターの扱い方 v

インタプリタがキューバスターに遭遇したときはいつでも、先読みを停止し、関連する結果が利用可能になるまで待つ必要があります。 これが機能する方法は次のとおりです。

- このようなコードが検出されると、インタープリターは特別な戻りコードをタスク（INTERP_EXECUTE_FINISH）に返します。
- このリターンコードは、タスクに信号を送り、今のところ先読みを停止し、これまでに構築されたすべてのキューに入れられた正規コマンド（キューバスターである最後のコマンドを含む）を実行してから、インタープリターの状態をワールドモデルと同期します。 技術的には、これは、HAL ピン値を反映するように内部変数を更新し、M6 の後にツールの形状をリロードし、プローブの結果を伝達することを意味します。
- インタプリタの synch（）メソッドはタスクによって呼び出され、それを実行します。さらに実行するために関連するすべてのワールドモデルの実際の値を読み取ります。
- この時点で、タスクは先に進み、プログラムが終了するか、別のキューバスターが発生するまで、インタプリタを呼び出してさらに先読みします。

6.10.17.5 語順と語順

互換性がある場合は、NGC ブロックに 1 つまたは複数の単語が存在する可能性があります（相互に排他的で、異なる行にある必要がある単語もあります）。 ただし、実行モデルでは、ソース行に表示されるかどうかに関係なく、コードの実行順序が厳密に規定されています（G コードの実行順序）。

6.10.17.6 構文解析

行が（MDI モードまたは現在の NGC ファイルから）読み取られると、その行が解析され、フラグとパラメーターが構造体ブロック（struct _setup、member block1）に設定されます。 この構造体は、現在のソース行に関するすべての情報を保持しますが、現在の行のコードの異なる順序とは無関係です。複数のコードに互換性がある限り、ソースの順序はすべて、構造体ブロックに設定された同じ変数になります。 解析直後に、ブロック上のすべてのコードの互換性がチェックされます。

6.10.17.7 実行

解析が成功した後、ブロックは execute_block（）によって実行され、ここでは実行順序に従ってさまざまなアイテムが処理されます。

「キューバスター」が見つかった場合、対応するフラグがインタープリター状態（toolchange_flag、input_flag、probe_flag）に設定され、インタープリターは INTERP_EXECUTE_FINISH 戻り値を返し、今のところ先読み停止を通知し、呼び出し元（タスク）に再同期します。 すべてのアイテムが実行された後にキューバスターが見つからない場合、INTERP_OK が返され、先読みが続行される可能性があることを通知します。

同期後に先読みが続くと、タスクはインタープリターの read（）操作の実行を再開します。 次の読み取り操作中に、上記のフラグがチェックされ、対応する変数が設定されます（synch（）が実行されたばかりであるため、値は現在のものになります）。 これは、次のコマンドが適切に設定された変数コンテキストですでに実行されていることを意味します

6.10.17.8 プロシージャの実行

O-word プロシージャは、キューバスターの処理を少し複雑にします。 ネストされたプロシージャのどこかにキューバスターが見つかり、INTERP_EXECUTE_FINISH が返されたときにプロシージャ呼び出しが半終了する可能性があります。 タスクは、ワールドモデルを同期し、プロシージャが実行されている限り（call_level> 0）、解析と実行を続行することを確認します。

6.10.17.9 ツールの変更が現在どのように機能するか

LinuxCNC で発生するアクションは少し複雑ですが、それらの動作を自分のニーズに適合させるために着手する前に、現在何が発生しているかを全体的に把握する必要があります。

既存のコードを再マッピングすると、そのコードのすべての内部処理が完全に無効になることに注意してください。 つまり、おそらく NGC Oword または Python プロシージャを介して記述された、目的の動作を超えて、インタープリターの内部アクションを複製する必要があります。これにより、

既存のコードが完全に置き換えられます。 プロローグとエピローグのコードは、これを行う場所です。

ツール情報の伝達方法いくつかのプロセスがツール情報に関心を持っています。タスクとそのインタープリター、およびユーザーインターフェイスです。 また、halui プロセス。

ツール情報は、すべての関係者によって共有される emcStatus 構造に保持されます。 そのフィールドの 1 つは、ツールテーブルファイルからロードされた説明（ツール番号、直径、フロントアングル、バックアングル、旋盤の方向、ツールオフセット情報）を保持する toolTable 配列です。

この構造でツール情報を実際に設定する信頼できるソースおよび唯一のプロセスは、iocontrol プロセスです。 他のすべてのプロセスは、この構造を参照するだけです。 インタプリタは、実際にはツールテーブルのローカルコピーを保持しています。

不思議なことに、現在の emcStatus 構造には Python ステートメントからアクセスできます。 たとえば、現在ロードされているツールに対する通訳者の認識には、次の方法でアクセスします。

```
;py,from interpreter import *
;py,print this.tool_table[0]
```

グローバル emcStatus 構造のフィールドを表示するには、次のことを試してください。

```
;py,from emctask import *
;py,print emcstat.io.tool.pocketPrepped
;py,print emcstat.io.tool.toolInSpindle
;py,print emcstat.io.tool.toolTable[0]
```

結果を表示するには、ターミナルウィンドウから LinuxCNC を起動する必要があります。

6.10.17.10 Tx（準備ツール）のしくみ

Tx コマンドのインタプリタアクション

インタプリタが行うのは、toolnumber パラメータを評価し、対応するポケットを検索し、後で使用するためにそれを selected_pocket 変数に記憶し、canon コマンド（SELECT_POCKET）をキューに入れることだけです。 src / emc / rs274/interp_execute.cc の Interp::convert_tool_select を参照してください。

SELECT_POCKET でのタスクアクションタスクが SELECT_POCKET を処理するために移動すると、タスクは EMC_TOOL_PREPARE メッセージを iocontrol プロセスに送信します。このプロセスは、LinuxCNC のほとんどのツール関連アクションを処理します。

現在の実装では、タスクは実際に iocontrol がチェンジャーのポジショニング操作を完了するのを待機しますが、これは IMO では必要ありません。チェンジャーの準備とコードの実行を並行して進めることができるという考えを打ち破ります。

EMC_TOOL_PREPARE での Iocontrol アクション iocontrol は selectpocket コマンドを検出すると、関連する HAL ピンを小刻みに動かします。「tool-prep-number」ピンを設定して次のツールを示し、「tool-prepare」ピンを上げて待機します。 「ツールで準備された」ピンがハイになります。

チェンジャーが「ツール準備済み」をアサートして応答すると、チェンジャーは準備フェーズが完了したと見なし、タスクを続行するように通知します。 （繰り返しますが、この待機は厳密に必要な IMO ではありません）

Tx のプロローグとエピローグの構築 configs/sim/axis の Python 関数 prepare_prolog と prepare_epilog を参照してください

6.10.17.11 M6（変更ツール）のしくみ

適応させる前に、これを完全に理解する必要があります。 これは、再マップされた M6 のプロローグおよびエピローグハンドラーの作成に非常に関連しています。 既存のコードを再マッピングするということは、通常行われる内部手順を無効にし、自分の目的のために必要な限りそれらを複製することを意味します。

C に精通していない場合でも、src / emc / rs274/interp_convert.cc の Interp::convert_tool_change コードを確認することをお勧めします。

M6 コマンドのインタープリターアクション

通訳者が M6 を見ると、次のようになります。

1. T コマンドがすでに実行されているかどうかをチェックし（テスト設定->selected_pocket が>= 0 になる）、そうでない場合はツール変更メッセージに対してツールの準備が必要-Txx-で失敗します。
2. カッター補正がアクティブになっているかどうかを確認し、失敗した場合は、カッター半径補正がオンになっている工具を変更できません。
3. 「TOOL_CHANGE_WITH_SPINDLE_ON」ini オプションが設定されている場合を除いて、スピンドルを停止します。
4. 「TOOL_CHANGE_QUILL_UP」ini オプションが設定されている場合、高速 Z アップ移動を生成します。
5. TOOL_CHANGE_AT_G30 が設定されている場合：
   a. 該当する場合は、A、B、および C インデクサーを移動します
   b. G30 位置への迅速な移動を生成します

6. 選択したポケットをパラメータとして、CHANGE_TOOLcanon コマンドを実行します。
   CHANGE_TOOL は次のようになります。
   a. generate a rapid move to TOOL_CHANGE_POSITION if so set in ini
   b. EMC_TOOL_LOADNML メッセージをタスクにエンキューします。
7. 新しいツールに従って、番号パラメータ 5400-5413 を設定します
8. M6 はキューバスターであるため、INTERP_EXECUTE_FINISH を返すことにより、先読みのためのインタープリターの呼び出しを停止するようにタスクに通知します。

CHANGE_TOOL コマンドを検出した場合のタスクは、EMC_TOOL_LOAD メッセージを送信し、iocontrol が処理を完了するまで待機することで、iocontrol にバックスを渡すだけです。

EMC_TOOL_LOAD での IOCONTROL アクション

1. 「ツール変更」ピンをアサートします
2. 「工具交換」ピンがアクティブになるのを待ちます
3. それが起こったとき：
   a. 「ツール交換」のアサートを解除する
   b. 「tool-prep-number」ピンと「tool-prep-pocket」ピンをゼロに設定します
   c. ポケットをパラメータとして load_tool（）関数を実行します。

最後のステップでは、実際に emcStatus 構造体にツールテーブルエントリを設定します。 実際に実行されるアクションは、RANDOM_TOOLCHANGER ini オプションが設定されているかどうかによって異なりますが、プロセスの最後に、toolTable[0]は現在スピンドルにあるツールを反映しています。

それが起こったとき：

1. iocontrol はタスクに先に進むように信号を送ります
2. タスクは、何が変更されたかを確認するために、synch（）操作を実行するようにインタープリターに指示します
3. インタプリタ synch（）は、変更されたツールテーブルなど、必要なワールドモデルからすべての情報を取得します。

そこから、通訳者は世界モデルの完全な知識を持ち、先読みを続けます。

M6 のプロローグとエピローグの構築 configs/sim/axis の Python 関数 change_prolog と change_epilog を参照してください

6.10.17.12 M61（工具番号変更）のしくみ

M61 には、負でない `Q`パラメータ（工具番号）が必要です。 ゼロの場合、これはツールのアンロードを意味し、そうでない場合は現在のツール番号を Q に設定します。

M61 の代替品の作成 M61 の Python 再定義の例は、set_tool_number 関数にあります。

configs / sim / axis / remap / toolchange / python/toolchange.py にあります。

6.10.18 状態

1. RELOAD_ON_CHANGE 機能はかなり壊れています。 Python ファイルを変更してから再起動してください。
2. M61（再マップされているかどうかに関係なく）は iocontrol で壊れており、実際に機能するには iocontrol-v2 が必要です。

6.10.19 変更点

- エラーメッセージを返し失敗するメソッドは、以前は self.set_errormsg（text）の後に INTERP_ERROR を返していました。 これは、Python ハンドラーまたは oword サブルーチンから文字列を返すだけで置き換えられました。 これにより、エラーメッセージが設定され、プログラムが中止されます。 以前は、Pythonoword サブルーチンを中止するクリーンな方法はありませんでした。

6.10.20 デバッグ

ini ファイルの[EMC]セクションで、DEBUG パラメータを変更して、LinuxCNC を端末から起動したときにさまざまなレベルのデバッグメッセージを取得できます。

```
Debug level, 0 means no messages. See src/emc/nml_intf/debugflags.h for others
DEBUG = 0x00000002 # configuration
DEBUG = 0x7FFFDEFF # no interp,oword
DEBUG = 0x00008000 # py only
DEBUG = 0x0000E000 # py + remap + Oword
DEBUG = 0x0000C002 # py + remap + config
DEBUG = 0x0000C100 # py + remap + Interpreter
DEBUG = 0x0000C140 # py + remap + Interpreter + NML msgs
DEBUG = 0x0000C040 # py + remap + NML
DEBUG = 0x0003E100 # py + remap + Interpreter + oword + signals + namedparams
DEBUG = 0x10000000 # EMC_DEBUG_USER1 - trace statements
DEBUG = 0x20000000 # EMC_DEBUG_USER2 - trap into Python debugger
DEBUG = 0x10008000 # USER1, PYTHON
```

```
DEBUG = 0x30008000 # USER1,USER2, PYTHON # USER2 will cause involute to try to
connect to  -
pydev
DEBUG = 0x7FFFFFFF # All debug messages
```

6.11 ムーブオフコンポーネント

MOVEOFF HAL コンポーネントは、オフセットを実装するための HAL のみの方法です。 重要な制限と警告については、マンページ（$ MAN MOVEOFF）を参照してください。

ムーブオフコンポーネントは、カスタム HAL 接続を使用してジョイント位置をオフセットするために使用されます。 プログラムが一時停止している間のオフセット機能の実装は、入力ピンの適切な接続でサポートされています。 9 つのジョイントがサポートされています。

軸オフセットピンの値（OFFSET-IN-M）は、出力ピン（OFFSET-CURRENT-M、POS-PLUSOFFSET-M、FB-MINUSOFFSET-M）に継続的に適用されます（値、速度、および加速度の制限を尊重します）。 両方のイネーブル入力ピン（APPLY-OFFSETS と MOVEENABLE）は TRUE です。 2 つの有効化入力は内部で ANDED されます。 オフセットが適用されている間に APPLY-OFFSETS ピンがディアサートされると、警告ピンが設定され、メッセージが発行されます。 警告ピンは、オフセットが削除されるか、オフセットの適用ピンが削除されるまで TRUE のままです。 が設定されています。

通常、MOVE-ENABLE ピンは外部コントロールに接続され、APPLY-OFFSETS ピンは HALUI.PROGRAM.IS-PAUSED（一時停止中のオフセットのみ）に接続されるか、TRUE（継続的に適用されるオフセット）に設定されます。

有効な入力のいずれかが非アクティブ化されると、適用されたオフセットは自動的にゼロに戻ります（制限を尊重します）。 ゼロ値の許容誤差は、イプシロン入力ピンの値によって指定されます。

ウェイポイントは、移動コンポーネントが有効になっているときに記録されます。 ウェイポイントは、WAYPOINT-SAMPLE-SECS ピンと WAYPOINT-THRESHOLD ピンで管理されます。 バックトラックイネーブルピンが TRUE の場合、自動リターンパスは記録されたウェイポイントに従います。 ウェイポイントに使用可能なメモリが使い果たされると、オフセットがフリーズし、ウェイポイント制限ピンがアサートされます。 この制限は、バックトラックイネーブルピンの状態に関係なく適用されます。 元の位置（オフセット以外の位置）に戻すには、イネーブルピンをディアサートする必要があります。

移動は速度と加速度の設定を尊重するポイントツーポイントであるため、ウェイポイントをバックトラックすると移動速度が遅くなります。 速度と加速度の制限ピンを動的に管理して、常にオフセットを制御できます。

BACKTRACK-ENABLE が FALSE の場合、自動復帰移動は調整されず、各軸は独自の速度でゼロに戻ります。 この状態で制御されたパスが必要な場合は、イネーブルピンをディアサートする前に、各軸を手動でゼロに戻す必要があります。 WAYPOINT-SAMPLE-SECS、WAYPOINT-THRESHOLD、および EPSILON ピンは、コンポーネントがアイドル状態のときにのみ評価されます。

オフセットが適用された出力ピンは、プログラムの再開を管理できるように、GUI に現在の状態を示すために提供されています。 APPLY-OFFSETS ピンがディアサートされたときにオフセットがゼロ以外の場合（たとえば、一時停止中にオフセットしたときにプログラムを再開した場合）、オフセットはゼロに戻され（制限を尊重）、エラーメッセージが発行されます。

警告

オフセットが有効化および適用され、何らかの理由でマシンがオフになっている場合、イネーブリングピンと OFFSET-IN-M 入力を管理する外部 HAL ロジックは、マシンが後で再びオンになったときの状態に責任があります。

この HAL のみのオフセット手段は、通常 LINUXCNC には認識されておらず、GUI プレビューディスプレイでも使用できません。 LINUXCNC によって管理されるソフト制限を超えるオフセット移動に対する保護は提供されません。 ソフトリミットは尊重されないため、オフセット移動はハードリミット（またはリミットスイッチがない場合はクラッシュ）に遭遇する可能性があります。 移動を制限するには、OFFSET-MIN-M および OFFSET-MAX-M 入力を使用することをお勧めします。 ハード制限をトリガーすると、マシンの電源がオフになります。上記の注意を参照してください。

OFFSET-IN-M 値は、INIFILE 設定で設定するか、GUI で制御するか、他の HAL コンポーネントおよび接続で管理することができます。 オフセットの方向と量が明確に定義されているが、オフセットをゼロに戻すためにイネーブルピンを非アクティブ化するための制御方法が必要な単純な場合には、固定値が適切な場合があります。 GUI は、ユーザーが各軸のオフセット値を設定、インクリメント、デクリメント、および累積する手段を提供し、イネーブルピンをディアサートする前に OFFSET-IN-M 値をゼロに設定する場合があります。

ACCEL、VEL、MIN、MAX、EPSILON、WAYPOINT-SAMPLE-SECS、および WAYPOINT-THRESHOLD のデフォルト値は、特定のアプリケーションに適していない場合があります。 この HAL コンポーネントは、LINUXCNC によって他の場所で適用される制限を認識していません。 ユーザーは、ハードウェアで使用する前に、シミュレーターアプリケーションで使用法をテストし、すべての危険を理解する必要があります

コンポーネントと GUI（MOVEOFF_GUI）を示す SIM 構成は、次の場所にあります。

- CONFIGS / SIM / AXIS / MOVEOFF（AXIS-UI）
- CONFIGS / SIM / TOUCHY / NGCGUI（TOUCHY-UI）

6.11.1 既存の構成の変更

システム提供のハーフイル（LIB：HOOKUP_MOVEOFF.TCL）を使用して、MOVEOFF コンポーネントを使用するように既存の構成を適合させることができます。 追加の INI ファイル設定は、オフセットを制御するための単純な GUI（MOVEOFF_GUI）の使用をサポートします。

システムハーフイル（LIB：HOOKUP_MOVEOFF.TCL）が構成 INI ファイルで適切に指定されている場合、次のようになります。

1. 元の JOINT.N.MOTOR-POS-CMD および JOINT.N.MOTOR-POS-FB ピン接続を切断します
2. INI ファイルで識別されたすべての軸に対応するように設定されたパーソナリティを使用して（MV という名前を使用して）MOVEOFF コンポーネントをロード（LOADRT）します。
3. 必要な順序で MOVEOFF コンポーネント関数を追加（ADDF）します
4. ムーブオフコンポーネントを使用するには、JOINT.N.MOTOR-POS-CMD ピンと JOINT.N.MOTOR-POS-FB ピンを再接続します
5. 追加の INI ファイル設定に従って、各軸のムーブオフコンポーネントの動作パラメータと制限を設定します

注：MOVEOFF_GUI アプリケーションは、NEMATICS_TYPE=KINEMATICS_SUPPORTED モジュールで既知のキネマティクスモジュールを使用する構成をサポートします。 アイデンティティキンの場合、MOVEOFF_GUI は、コマンドラインパラメータ-AXESAXISNAMES で指定された各軸名を対応するジョイントに割り当てます。

次のように既存の構成を変更します。

[HAL] HALUI の INI ファイルエントリがあることを確認し、LIB：HOOKUP_MOVEOFF.TCL の新しい [HAL]HALFILE エントリを作成します。 LIB：HOOKUP_MOVEOFF.TCL のエントリは、JOINT.N.MOTOR-POSCMD、JOINT.N.MOTOR-POS-FB、およびこれらのピンに接続されているすべてのコンポーネント（PID およびエンコーダー）のピンを接続するハーフイルのすべての HALFILE=エントリの後に続く必要があります。 たとえば、サーボシステムのコンポーネント）。

```
[HAL]
HALUI = halui
HALFILE = existing_configuration_halfile_1
...
HALFILE = existing_configuration_halfile_n
HALFILE = LIB:hookup_moveoff.tcl
```

使用中の各軸の軸ごとの設定に INI ファイルエントリを追加します（エントリが定義されていない場合は、[AXIS_N]セクションの対応するエントリが使用され、エントリが見つからない場合は、

MOVEOFF コンポーネントのデフォルトが使用されます） 注：軸ごとのオフセット設定にコンポーネントのデフォルトまたは[AXIS_N]セクション値を使用することはお勧めしません。

```
[MOVEOFF_n]
MAX_LIMIT =
MIN_LIMIT =
MAX_VELOCITY =
MAX_ACCELERATION
```

MOVEOFF コンポーネント設定の INI ファイルエントリを追加します（MOVEOFF のデフォルトを使用しないでください）。

```
[MOVEOFF]
EPSILON =
WAYPOINT_SAMPLE_SECS =
WAYPOINT_THRESHOLD =
```

MOVEOFF_GUI は、追加の必要な接続を確立し、次のポップアップ GUI を提供するために使用されます。

1. オフセットを有効/無効にするためのコントロールトグルボタンを提供します
2. バックトラッキングを有効/無効にするためのコントロールトグルボタンを提供します
3. 各軸オフセットをインクリメント/デクリメント/ゼロにするための制御プッシュボタンを提供します
4. 各軸のオフセット電流値を表示します
5. 現在のオフセットステータス（無効、アクティブ、削除など）を表示します

付属のコントロールボタンは、ムーブオフコンポーネントのムーブイネーブルピンの状態に応じてオプションです。 MOVEOFF_GUI の開始時にピン MV.MOVE-ENABLE が接続されていない場合は、オフセットを有効にするための表示とコントロールの両方が提供されます。 この場合、MOVEOFF_GUI は、MOVEOFF コンポーネントの MOVE-ENABLE ピン（MV.MOVE-ENABLE という名前）、オフセット（MV.MOVE-OFFSET-IN-M）、およびバックトラッキングイネーブル（MV.BACKTRACK-ENABLE）を管理します。

MOVEOFF_GUI の開始時に MV.MOVE-ENABLE ピンが接続されている場合、MOVEOFF_GUI は表示を提供しますが、コントロールは提供しません。 このモードは、ジョグホイールまたはオフセット入力とイネーブルピンを制御する他の方法（MV.OFFSET-IN-M、MV.MOVE-ENABLE、MV.BACKTRACK-ENABLE）を使用する構成をサポートします。

MOVEOFF_GUI は、MOVEOFF コンポーネントピンに必要な接続（MV.POWER_ON および MV.APPLY-OFFSETS）を作成します。 MV.POWER_ON ピンは MOTION.MOTION 対応ピンに接続されています（必要に応じて新しい信号が自動的に作成されます）。 MV.APPLY-OFFSETS は HALUI.PROGRAM.IS-PAUSED に接続されるか、コマンドラインオプションに応じて 1 に設定されます-MODE [ONPAUSE | いつも]。 必要に応じて、新しい信号が自動的に作成されます。

MOVEOFF_GUI を使用するには、次のように INI ファイル[アプリケーション]セクションにエントリを追加します。

```
[APPLICATIONS]
# Note: a delay (specified in seconds) may be required if connections
# are made using postgui halfiles ([HAL]POSTGUI_HALFILE=)
DELAY = 0
APP = moveoff_gui option1 option2 ...
```

ハーフイル LIB：HOOKUP_MOVEOFF.TCL を使用して MOVEOFF コンポーネントをロードおよび接続する場合、MV.MOVE-ENABLE ピンは接続されず、MOVEOFF_GUI によって提供されるローカルコントロールが使用されます。 これは、既存の INI 構成を変更するときに MOVEOFF コンポーネントをテストまたはデモンストレーションするための最も簡単な方法です。

オフセット値とステータスに MOVEOFF_GUI 表示を使用しているときに外部コントロールを有効にするには、LIB：HOOKUP_MOVEOFF.TCL に続くハーフイルが追加の接続を行う必要があります。 たとえば、提供されているデモンストレーション構成（CONFIGS / SIM / AXIS / MOVEOFF / *。INI）は、単純なシステムハーフイル（LIB：MOVEOFF_EXTERNAL.HAL という名前）を使用して MV.MOVE-ENABLE、MV.OFFSET-IN-M を接続します。 および MV.BACTRACK-信号へのピンを有効にします。

```
[HAL]
HALUI = halui
...
HALFILE = LIB:hookup_moveoff.tcl
HALFILE = LIB:moveoff_external.hal
```

LIB：MOVEOFF_EXTERNAL.HAL（3 軸構成の場合）によって行われる接続は次のとおりです。

```
net external_enable mv.move-enable
net external_offset_0 mv.offset-in-0
net external_offset_1 mv.offset-in-1
net external_offset_2 mv.offset-in-2
net external_backtrack_en mv.backtrack-enable
```

これらの信号（EXTERNAL_ENABLE、EXTERNAL_OFFSET_M、EXTERNAL_BACKTRACK_EN）は、後続の HALFILE（POSTGUI_HALFILE を含む）によって管理され、現在のオフセット値とオフセットステータスに MOVEOFF_GUI 表示を使用しながら、コンポーネントのカスタマイズされた制御を提供できます。

MOVEOFF_GUI は、コマンドラインオプションを使用して設定されます。 MOVEOFF_GUI の操作の詳細については、MAN ページを参照してください。

```
$ man moveoff_gui
```

MOVEOFF_GUI のコマンドラインオプションの簡単なリストについては、コマンドラインヘルプオプションを使用してください。

```
$ moveoff_gui --help
Usage:
moveoff_gui [Options]
Options:
[--help | -? | -- -h ] (This text)
[-mode [onpause | always]] (default: onpause)
(onpause: show gui when program paused)
(always: show gui always)
[-axes axisnames] (default: xyz (no spaces))
(letters from set of: x y z a b c u v w)
(example: -axes z)
(example: -axes xz)
(example: -axes xyz)
[-inc incrementvalue] (default: 0.001 0.01 0.10 1.0 )
(specify one per -inc (up to 4) )
(example: -inc 0.001 -inc 0.01 -inc 0.1 )
[-size integer] (default: 14
(Overall gui popup size is based on font size)
[-loc center|+x+y] (default: center)
(example: -loc +10+200)
[-autoresume] (default: notused)
(resume program when move-enable deasserted)
[-delay delay_secs] (default: 5 (resume delay))
Options for special cases:
[-noentry] (default: notused)
(don't create entry widgets)
[-no_resume_inhibit] (default: notused)
```

(do not use a resume-inhibit-pin)
[-no_pause_requirement] (default: notused)
(no check for halui.program.is-paused)
[-no_cancel_autoresume] (default: notused)
(useful for retraact offsets with simple)
(external control)
[-no_display] (default: notused)
(Use when both external controls and displays)
(are in use (see Note)))
Note: If the moveoff move-enable pin (mv.move-enable) is connected when
moveoff_gui is started, external controls are required and only
displays are provided.

6.12 ステッパー構成

6.12.1 前書き

標準のステッパーマシンをセットアップするための推奨される方法は、ステップ構成ウィザードを使用することです。 ステッパー設定ウィザードの章を参照してください。

この章では、ステッパーベースのシステムを手動でセットアップするためのより一般的な設定のいくつかについて説明します。 これらのシステムは、ステップおよび方向信号を受け入れるドライブを備えたステッピングモーターを使用しています。

モーターは開ループで動作するため（モーターからのフィードバックは返されません）、これはより簡単なセットアップの 1 つですが、モーターが停止したりステップを失ったりしないように、システムを適切に構成する必要があります。

この章のほとんどは、LINUXCNC とともにリリースされたサンプル構成に基づいています。 構成は STEPPER_INCH と呼ばれ、構成ピッカーを実行することで見つけることができます。

6.12.2 最大ステップレート

ソフトウェアステップ生成では、最大ステップレートは、ステップアンドディレクション出力の 2 つの BASE_PERIOD ごとに 1 ステップです。 要求される最大ステップレートは、軸の MAX_VELOCITY とその INPUT_SCALE の積です。 要求されたステップレートが達成できない場合、特に高速ジョグおよび G0 移動中に、次のエラーが発生します。

ステッパードライバーが直交入力を受け入れることができる場合は、このモードを使用してください。直交信号では、BASE_PERIOD ごとに 1 つのステップが可能であり、最大ステップレートが 2 倍になります。

他の解決策は、BASE_PERIOD（これを低く設定しすぎると、マシンが応答しなくなったり、ロックアップしたりする）、INPUT_SCALE（ステッパードライバーで異なるステップサイズを選択できる場合は、プーリー比を変更する）の 1 つ以上を減らすことです。 または親ねじピッチ）、または MAX_VELOCITY および STEPGEN_MAXVEL。

BASE_PERIOD、INPUT_SCALE、および MAX_VELOCITY の有効な組み合わせが受け入れられない場合は、ハードウェアステップ生成（LINUXCNC でサポートされているユニバーサルステッパーコントローラー、MESA カードなど）の使用を検討してください。

6.12.3 ピン配列

LINUXCNC の主な欠陥の 1 つは、ソースコードを再コンパイルせずにピン配置を指定できないことでした。 LINUXCNC ははるかに柔軟性があり、（ハードウェアアブストラクションレイヤーのおかげで）どの信号をどこに送るかを簡単に指定できるようになりました。 HAL の詳細については、HAL の基本を参照してください。

HAL の概要とチュートリアルで説明されているように、HAL 内には信号、ピン、およびパラメーターがあります。

NOTE

短くするために 1 つの軸を提示していますが、他のすべての軸は類似しています。

ピン配置に関連するものは次のとおりです。

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

.INI ファイルで選択した内容に応じて、STANDARD_PINOUT.HAL または XYLOTEX_PINOUT.HAL のいずれかを使用します。 これらは、さまざまな信号とピンをリンクする方法を HAL に指示する 2 つのファイルです。さらに、STANDARD_PINOUT.HAL を調査します。

1.1.1.1 標準ピン配置 HAL

このファイルにはいくつかの HAL コマンドが含まれており、通常は次のようになります。

```
# 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 spindle.0.on => parport.0.pin-09-out
###
### You might use something like this to enable chopper drives when machine ON
### the Xen signal is defined in core_stepper.hal
###
# net Xen => parport.0.pin-01-out
###
### If you want active low for this pin, invert it like this:
###
# setp parport.0.pin-01-out-invert 1
###
### A sample home switch on the X axis (axis 0). make a signal,
### link the incoming parport pin to the signal, then link the signal
### to LinuxCNC's axis 0 home switch input pin
###
# net Xhome parport.0.pin-10-in => joint.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.
###
# net homeswitches <= parport.0.pin-10-in
# net homeswitches => joint.0.home-sw-in
# net homeswitches => joint.1.home-sw-in
# net homeswitches => joint.2.home-sw-in
###
### Sample separate limit switches on the X axis (axis 0)
###
# net X-neg-limit parport.0.pin-11-in => joint.0.neg-lim-sw-in
# net X-pos-limit parport.0.pin-12-in => joint.0.pos-lim-sw-in
###
### Just like the shared home switches example, you can wire together
### limit switches. Beware if you hit one, LinuxCNC will stop but can't tell
### you which switch/axis has faulted. Use caution when recovering from this.
###
# net Xlimits parport.0.pin-13-in => joint.0.neg-lim-sw-in joint.0.pos-lim-sw-in
```

#で始まる行はコメントであり、その唯一の目的は、読者にファイルを案内することです。

6.12.3.1 概要

STANDARD_PINOUT.HAL が実行/解釈されるときに実行される操作がいくつかあります。

- PARPORT ドライバーがロードされます（詳細については、PARPORT の章を参照してください）。
- パーポートドライバーの読み取りおよび書き込み機能は、ベーススレッドに割り当てられます。1)
- 軸 X、Y、Z のステップおよび方向信号は、パーポートのピンにリンクされます。
- さらに I / O 信号が接続されます（ESTOP ループバック、TOOLCHANGER ループバック）
- スピンドルオン信号が定義され、パーポートピンにリンクされます

1) LINUXCNC セットアップで最速のスレッド。通常、コードは数十マイクロ秒ごとに実行されます。

6.12.3.2 STANDARD_PINOUT.HAL の変更

STANDARD_PINOUT.HAL ファイルを変更する場合、必要なのはテキストエディタだけです。 ファイルを開き、変更する部分を見つけます。

たとえば、X 軸のステップと方向信号のピンを変更する場合は、PARPORT.0.PIN-XX-OUT 名の番号を変更するだけです。

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

次のように変更できます。

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

または基本的にあなたが好きな他のアウトピン。

ヒント：同じピンに複数の信号が接続されていないことを確認してください。

6.12.3.3 信号の極性を変更する

外部ハードウェアが「アクティブロー」信号を予期している場合は、対応する-INVERT パラメーターを設定します。 たとえば、スピンドル制御信号を反転するには、次のようにします。

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

6.12.3.4 PWM スピンドル速度制御の追加

スピンドルを PWM 信号で制御できる場合は、PWMGEN コンポーネントを使用して信号を作成します。

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

これは、PWM に対するスピンドルコントローラーの応答が単純であることを前提としています。 0%PWM は 0 RPM、10%PWM は 180 RPM などです。スピンドルを回転させるために必要な最小 PWM がある場合は、NIST-LATHE の例に従ってください。 スケールコンポーネントを使用するためのサンプル構成。

6.12.3.5 イネーブル信号の追加

一部の増幅器（ドライブ）は、モーターの動きを受け入れて命令する前に、イネーブル信号を必要とします。 このため、XEN、YEN、ZEN と呼ばれるシグナルがすでに定義されています。

それらを接続するには、次の例を使用します。

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

すべてのドライブを有効にする単一のピンを持つことができます。 設定に応じて、またはいくつか。 ただし、通常、1 つの軸に障害が発生すると、他のすべてのドライブも無効になるため、すべてのドライブに対して 1 つのイネーブル信号/ピンのみを使用するのが一般的な方法であることに注意してください。

6.12.3.6 外部非常停止ボタン

STANDARD_PINOUT.HAL ファイルは、外部非常停止ボタンがないことを前提としています。 外部非常停止の詳細については、ESTOP_LATCH のマニュアルページを参照してください。

7章 コントロールパネル

7.1 PYTHON 仮想コントロールパネル

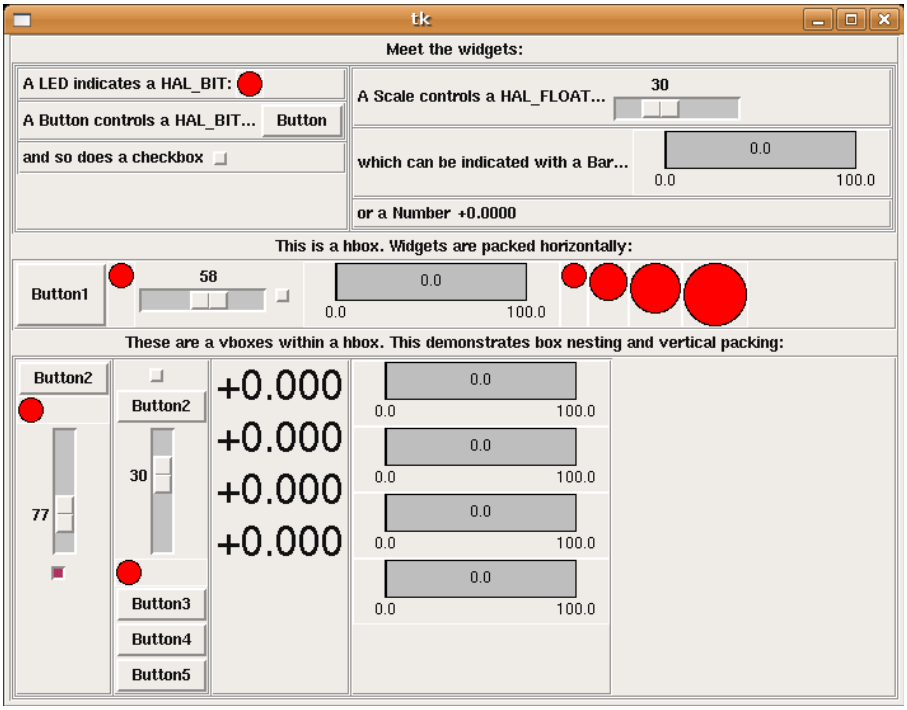
7.1.1 前書き

Python 仮想コントロールパネル PyVCP（Python 仮想コントロールパネル）は、インテグレーターがボタンとインジケーターを使用して AXIS インターフェイスをカスタマイズして特別なタスクを実行できるように設計されています。

ハードウェアマシンのコントロールパネルは、多くの I / O ピンを消費する可能性があり、高価になる可能性があります。 ここで、仮想コントロールパネルに利点があり、PyVCP の構築に費用はかかりません。

仮想コントロールパネルは、ラダーロジックのデバッグ中に実際の I / O デバイスを一時的に置き換えるためのテストや監視、または物理パネルを構築して I / O ボードに配線する前にシミュレートするために使用できます。

次の図は、PyVCP ウィジェットの多くを示しています。



7.1.2 パネル構造

PyVCP パネルのレイアウトは、<pyvcp>と</ pyvcp>の間のウィジェットタグを含む XML ファイルで指定されます。 例えば：

このテキストを tiny.xml というファイルに配置して、

```
halcmd loadusr pyvcp -c mypanel tiny.xml
```

PyVCP は、2 つのウィジェット、「これは LED インジケーターです」というテキストのラベルと HALBIT 信号の状態を表示するために使用される LED を含むパネルを作成します。 また、mypanel という名前の HAL コンポーネントも作成されます（このパネルのすべてのウィジェットは、mypanel で始まるピンに接続されています）。 <led>タグ内に<halpin>タグが存在しなかったため、PyVCP は LED ウィジェット mypanel.led.0 の HAL ピンに自動的に名前を付けます。

ウィジェットとそのタグおよびオプションのリストについては、以下のウィジェットリファレンスを参照してください。

パネルを作成したら、HAL 信号を PyVCP ピンとの間で接続するには、halcmd を使用します。

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

HAL を初めて使用する場合は、インテグレータマニュアルの HAL の基本の章から始めることをお勧めします。

7.1.3 安全

PyVCP ファイルの一部は Python コードとして評価され、Python プログラムで利用可能な任意のアクションを実行できます。 信頼できるソースからの PyVCP.xml ファイルのみを使用してください。

7.1.4 AXIS

AXIS は PyVCP と同じ GUI ツールキット（Tkinter）を使用するため、AXIS ユーザーインターフェイスの右側または下部に PyVCP パネルを含めることができます。 典型的な例を以下に説明します。

パネルを記述した PyVCPXML ファイルを、.ini ファイルと同じディレクトリに配置します。 バーウィジェットを使用して現在のスピンドル速度を表示するとします。 次のものを spindle.xml というファイルに配置します。

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

ここでは、ラベルとバーウィジェットを備えたパネルを作成し、バーに接続された HAL ピンにスピンドル速度という名前を付け、バーの最大値を 5000 に設定するように指定しました（すべてのオプションについては、以下のウィジェットリファレンスを参照してください）。 。 AXIS にこのファイルを認識させ、起動時に呼び出すには、.ini ファイルの[DISPLAY]セクションで次のように指定する必要があります。

```
PYVCP = spindle.xml
```

パネルが AXIS ユーザーインターフェイスの下部に表示される場合は、.ini ファイルの[DISPLAY]セクションで次のように指定する必要があります。

```
PYVCP_POSITION = BOTTOM
```

BOTTOM 以外の場合、またはこの変数を省略すると、PYVCP パネルが右側に配置されます。

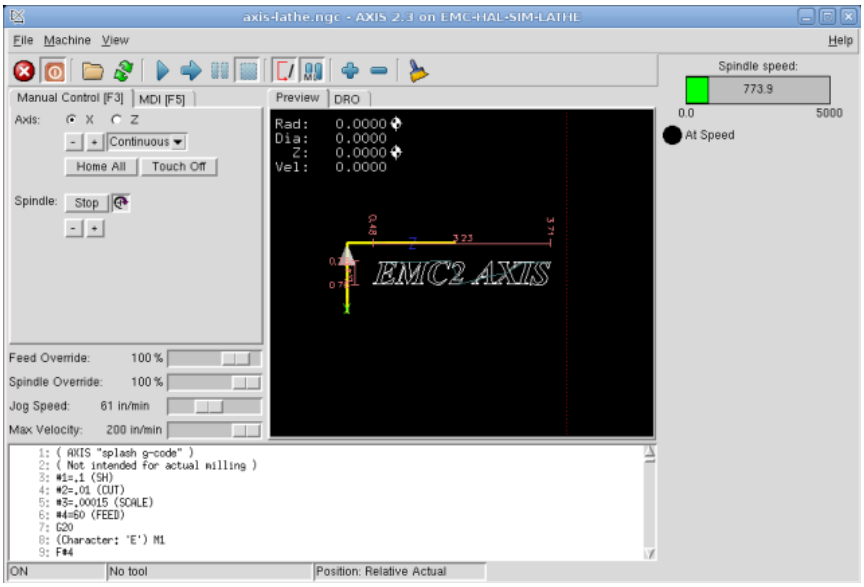
ウィジェットに実際にスピンドル速度を表示させるには、適切な HAL 信号に接続する必要があります。 AXIS および PyVCP が開始されると実行される.hal ファイルは、.ini ファイルの[HAL]セクションで指定できます。

```
POSTGUI_HALFILE = spindle_to_pyvcp.hal
```

この変更により、spland_to_pyvcp.hal で指定された HAL コマンドが実行されます。 この例では、内容は次のようになります。

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

スピンドル-rpm-filtered と呼ばれる信号がすでに存在すると仮定します。 AXIS と一緒に実行する場合、すべての PyVCP ウィジェットの HAL ピンの名前は pyvcp で始まることに注意してください。



これは、新しく作成された PyVCP パネルが AXIS でどのように表示されるかを示しています。 sim / lathe 構成はすでにこのように構成されています。

7.1.5 スタンドアロン

このセクションでは、LinuxCNC のマシンコントローラーの有無にかかわらず、PyVCP パネルを単独で表示する方法について説明します。 LinuxCNC でスタンドアロン PyVCP パネルをロードするには、次のコマンドを使用します。

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

フローティングパネルまたは AXIS 以外の GUI を備えたパネルが必要な場合にこれを使用します。

- -Wn panelname-HAL は、コンポーネント panelname のロードが完了する（HAL スピークで準備が整う）のを待ってから、さらに HAL コマンドを処理します。 PyVCP パネルは HAL ピンをエクスポートし、他の HAL

コンポーネントはそれらに接続するためにそれらが存在する必要があるため、これは重要です。 大文字の W と小文字の n に注意してください。 -Wn オプションを使用する場合は、-c オプションを使用してパネルに名前を付ける必要があります。

- pyvcp <-g> <-c> panel.xml-xml パネルファイルからオプションのジオメトリや panelname を使用してパネルを構築します。 panel.xml は、.xml で終わる任意の名前にすることができます。 .xml ファイルは、パネルの作成方法を説明するファイルです。 パネルが HAL スクリプトのあるディレクトリにない場合は、パス名を追加する必要があります。
- -g <WxH> <+ X + Y>-パネルを作成するときに使用するジオメトリを指定します。 構文は、幅 x 高さ+ X アンカー+ Y アンカーです。 サイズまたは位置、あるいはその両方を設定できます。 アンカーポイントは、パネルの左上隅です。 例は-g250x500 + 800 + 0 です。これにより、パネルが幅 250 ピクセル、高さ 500 ピクセルに設定され、X800Y0 に固定されます。
- -c panelname-PyVCP に、コンポーネントの名前とウィンドウのタイトルを指示します。 パネル名は、スペースを含まない任意の名前にすることができます。

LinuxCNC なしでスタンドアロンの PyVCP パネルをロードするには、次のコマンドを使用します。

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

pyvcp パネルをロードするための最小コマンドは次のとおりです。

```
loadusr pyvcp mypanel.xml
```

テストやスタンドアロンの DRO など、LinuxCNC のマシンコントローラーのないパネルが必要な場合は、これを使用します。

loadusr コマンドは、HAL が完了するまで閉じるのを停止するコンポーネントもロードするときに使用されます。 パネルをロードしてから、loadusr -wclassicladder を使用して ClassicLadder をロードした場合、CL は、CL を閉じるまで HAL（およびパネル）を開いたままにします。 上記の-Wn は、コンポーネント-Wn "name"の準備が整うのを待つことを意味します。 （名前は任意の名前にすることができます。大文字の W と小文字の n に注意してください。）-c は、panel_file_name.xml の情報を使用して、panelname という名前のパネルを作成するように PyVCP に指示します。 panel_file_name.xml という名前は任意の名前にすることができますが、.xml で終わる必要があります。これは、パネルの作成方法を説明するファイルです。 パネルが HAL スクリプトのあるディレクトリにない場合は、パス名を追加する必要があります。

パネルが HAL によるコマンドの続行/シャットダウンを停止する場合に使用するオプションのコマンド。 他のコンポーネントをロードした後、最後の HAL コマンドを次のようにします。

```
waituser panelname
```

これは、HAL コマンドを続行する前に、コンポーネント panelname が閉じるのを待つように HAL に指示します。 これは通常、パネルが閉じられたときに HAL がシャットダウンするように、最後のコマンドとして設定されます。

7.1.6 ウィジェット

HAL 信号には、ビットと数値の 2 つのバリエーションがあります。 ビットはオフ/オン信号です。 数値は、float、s32、または u32 にすることができます。 HAL データ型の詳細については、「HAL データ」セクションを参照してください。 PyVCP ウィジェットは、インジケーターウィジェットを使用してシグナルの値を表示するか、コントロールウィジェットを使用してシグナル値を変更することができます。 したがって、HAL 信号に接続できる PyVCP ウィジェットには 4 つのクラスがあります。 ヘルパーウィジェットの 5 番目のクラスを使用すると、パネルを整理してラベルを付けることができます。

1. ビット信号を示すためのウィジェット：led、rectled
2. ビット信号を制御するためのウィジェット：ボタン、チェックボタン、ラジオボタン
3. 数字信号を示すためのウィジェット：数字、s32、u32、バー、メーター
4. 数値信号を制御するためのウィジェット：スピンボックス、スケール、ジョグホイール
5. ヘルパーウィジェット：hbox、vbox、table、label、labelframe

1.1.1.1 構文

各ウィジェットについて簡単に説明し、次に使用するマークアップとスクリーンショットを示します。 メインウィジェットタグ内のすべてのタグはオプションです。

7.1.6.1 一般的注意事項

現在、タグベースの構文と属性ベースの構文の両方がサポートされています。 たとえば、次の XML フラグメントは同じように扱われます。

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

と

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

属性ベースの構文を使用する場合、次のルールを使用して属性値を Python 値に変換します。

1. 属性の最初の文字が次のいずれかである場合、Python 式として評価されます：{（['"
2. 文字列が int（）によって受け入れられる場合、値は整数として扱われます
3. 文字列が float（）によって受け入れられる場合、値は浮動小数点として扱われます
4. それ以外の場合、文字列は文字列として受け入れられます。

タグベースの構文を使用する場合、タグ内のテキストは常に Python 式として評価されます。

以下の例は、さまざまな形式を示しています。

コメントコメントを追加するには、コメントに xml 構文を使用します。

```
<!-- My Comment -->
```

XML ファイルの編集テキストエディタで XML ファイルを編集します。 ほとんどの場合、ファイルを右クリックして、テキストエディタなどで開くを選択できます。

色

色は、Gray75 または 16 進数＃0000ff という名前の X11rgb 色を使用して指定できます。 完全なリストはここ http://sedition.com/-perl/rgb.html にあります。

一般的な色（数字の付いた色はその色の色合いを示します）

- 白
- 黒
- 青と青 1-4
- シアンとシアン 1-4
- 緑と緑 1-4
- 黄色と黄色 1-4
- 赤と赤 1-4
- 紫と紫 1-4
- 灰色と灰色 0〜100

HAL ピン HAL ピンは、ウィジェットを何かに接続する手段を提供します。 ウィジェットの HAL ピンを作成したら、.hal ファイルの net コマンドを使用してウィジェットを別の HAL ピンに接続できます。 net コマンドの詳細については、「HAL コマンド」セクションを参照してください。

7.1.6.2 ラベル

ラベルは、パネルにテキストを追加する方法です。

- <label> </ label>-ラベルを作成します
- <text> "text" </ text>-ラベルに配置するテキスト。空白のラベルをスペーサーとして使用して他のオブジェクトを整列させることができます。
- <font>（ "Helvetica"、20）</ font>-テキストのフォントとサイズを指定します
- <relief> FLAT </ relief>-ラベル（FLAT、RAISED、SUNKEN）の周囲の境界線を指定します。デフォルトは FLAT です。
- <bd> n </ bd>-ここで、n は、RAISED または SUNKEN ボーダーが使用されている場合のボーダー幅です。
- <padx> n </ padx>-ここで、n は余分な水平方向の余分なスペースの量です。
- <pady> and </ paddy>-ここで、n は余分な垂直方向の余分なスペースの量です。

ラベルには、<disable_pin> True </ disable_pin>を追加したときに作成されるオプションの無効化ピンがあります。

```
<label>
    <text>"This is a Label:"</text>
    <font>("Helvetica",20)</font>
</label>
```

上記のコードはこの例を生成しました。

7.1.6.3 Multi_Label

テキストラベルの拡張。

選択可能なテキストラベル。関連するビットピンがアクティブになると、最大 6 つのラベル凡例を表示できます。

各凡例ピンを信号に接続し、信号が TRUE の場合に説明ラベルを取得します。

複数の凡例ピンが TRUE の場合、最も大きい番号の TRUE 凡例が表示されます。

<disable_pin> True </ disable_pin>で無効化ピンが作成され、そのピンが true に設定されている場合、ラベルはグレー表示された状態に変わります。

```
<multilabel>
<legends>["Label1", "Label2", "Label3", "Label4", "Label5", "Label6"]</legends>
<font>("Helvetica",20)</font>
<disable_pin>True</disable_pin>
</multilabel>
```

上記の例では、次のピンが作成されます。

```
pyvcp.multilabel.0.disable
pyvcp.multilabel.0.legend0
pyvcp.multilabel.0.legend1
pyvcp.multilabel.0.legend2
pyvcp.multilabel.0.legend3
pyvcp.multilabel.0.legend4
pyvcp.multilabel.0.legend5
```

複数のマルチラベルがある場合、作成されたピンはこの pyvcp.multilabel.1.legend1 のように番号をインクリメントします。

7.1.6.4 LEDs

LED は、ビットハルピンのステータスを示すために使用されます。 LED の色は、ハルピンが true の場合は on_color になり、それ以外の場合は off_color になります。

- <led> </ led>-丸い LED を作ります

- <rectled> </ rectled>-長方形の LED を作成します
- <halpin> name </ halpin>-ピンの名前。デフォルトは led.n です。ここで、n は LED ごとにインクリメントされる整数です。
- <size> n </ size> -n は LED のサイズ（ピクセル単位）で、デフォルトは 20 です。
- <on_color> color </ on_color>-ピンが真の場合の LED の色を設定します。 デフォルトは緑です。 詳細については、色を参照してください。
- <off_color> color </ off_color>-ピンが false の場合の LED の色を設定します。 デフォルトは赤です
- <height> and </ weight> -LED の高さをピクセル単位で設定します
- <width> n </ width> -LED の幅をピクセル単位で設定します
- <disable_pin> false </ disable_pin> -true の場合、LED に無効ピンを追加します。
- <disabled_color> color </ disabled_color>-ピンが無効になっているときの LED の色を設定します。

丸い LED

```
<led>
    <halpin>"my-led"</halpin>
    <size>50</size>
    <on_color>"green"</on_color>
    <off_color>"red"</off_color>
</led>
```

上記のコードはこの例を生成しました

長方形 LED これは LED ウィジェットの変形です。

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

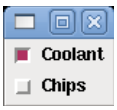
上記のコードはこの例を生成しました。 また、浮き彫りのある縦の箱を示しています。

7.1.6.5 ボタン

ボタンは、BIT ピンを制御するために使用されます。 ボタンを押し続けるとピンが True に設定され、ボタンを離すとピンが False に設定されます。 ボタンは、次のオプションオプションを使用できます。

- <padx> n </ padx>-ここで、n は余分な水平方向の余分なスペースの量です。
- <pady> and </ paddy>-ここで、n は余分な垂直方向の余分なスペースの量です。
- <activebackground> "color" </ activebackground>-色の上にカーソルを置きます。
- <fg> "color" </ fg>-前景色。
- <bg> "color" </ bg>-背景色。
- <disable_pin> True </ disable_pin>-ピンを無効にします。

上記のコードはこの例を生成しました。 クーラントチェックボタンがチェックされます。 チェックボタンの位置を揃えるために、チップテキストの余分なスペースに注意してください。



ラジオボタンラジオボタンは、ハルピンの 1 つを true に設定します。 他のピンは false に設定されています。 initval フィールドは、パネルが表示されたときにデフォルトの選択を選択するように設定できます。 1 つのラジオボタンのみを TRUE（1）に設定するか、TRUE に設定された最大番号のピンのみがその値を持ちます。

```
<radiobutton>
    <choices>["one","two","three"]</choices>
    <halpin>"my-radio"</halpin>
    <initval>0</initval>
</radiobutton>
```

上記のコードはこの例を生成しました。

上記の例の HAL ピンには、my-radio.one、my-radio.two、および my-radio.three という名前が付けられていることに注意してください。 上の画像では、1 つが選択された値です。 このタグ<orient> HORIZONTAL </ orient>を使用して、水平に表示します。

7.1.6.6 数字表示

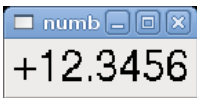
数値表示では、次の書式設定オプションを使用できます

- <font>（ "フォント名"、n）</ font>ここで、n はフォントサイズです。
- <width> n </ width>ここで、n は使用されるスペースの全体の幅です。
- <justify> pos </ justify>ここで、pos は LEFT、CENTER、または RIGHT です（機能しません）
- <padx> n </ padx>ここで、n は余分な水平方向の余分なスペースの量です
- <pady> and </ paddy>ここで、n は余分な垂直方向の余分なスペースの量です

数値数値ウィジェットは、浮動小数点信号の値を表示します。

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

上記のコードはこの例を生成しました。



- <font> -Tkinter のフォントタイプとサイズの仕様です。 少なくともサイズ 200 まで表示されるフォントの 1 つは、宅配便の 10 ピッチであるため、非常に大きな Number ウィジェットの場合は次のように指定できます。

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

- <format>-数値の表示方法を決定する、指定された C スタイルの形式です。

s32 番号 s32 番号ウィジェットは、s32 番号の値を表示します。 構文は、名前が<s32>であることを除いて、number と同じです。 幅が、使用する予定の最大数をカバーするのに十分な幅であることを確認してください。

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

上記のコードはこの例を生成しました。

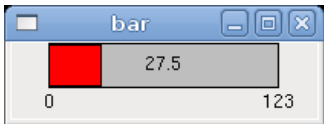
u32 番号 u32 番号ウィジェットは、u32 番号の値を表示します。 構文は、名前が<u32>であることを除いて、number と同じです。

バーバーウィジェットは、バー表示を使用してグラフィカルに、および数値的に FLOAT 信号の値を表示します。 バーの色は、その範囲全体で 1 つの色として設定するか（デフォルトでは fillcolor を使用）、ハルピンの値に応じて色を変更するように設定できます（range1、range2 range3 はすべて設定する必要があり、2 つの範囲のみが必要な場合は、2 を設定します） それらの同じ色に）。

- <halpin> "my-bar" </ halpin>テキスト、ピン名 pyvcp.my-bar を設定します
- <min_> 0 </ min_>番号、最小スケールを設定します
- <max_> 140 </ max_>番号、最大スケールを設定します
- <format> "3.1f" </ format>テキスト、Python の数値フォーマットを使用して数値フォーマットを設定します
- <bgcolor> "grey" </ bgcolor>テキスト、背景色を設定します
- <fillcolor> "red" </ fillcolor>テキスト、塗りつぶしの色を設定します
- <range1> 0,100、 "green" </ range1>数値、数値、テキスト、最初の範囲と色を設定します
- <range2> 101,135、 "orange" </ range2>数値、数値、テキスト、最初の範囲と色を設定します
- <range3> 136、150、 "red" </ range3>数値、数値、テキスト、最初の範囲と色を設定します
- <canvas_width> 200 </ canvas_width>番号、全体の幅を設定します
- <canvas_height> 50 </ canvas_height>番号、全体の高さを設定します
- <bar_height> 30 </ bar_height>数値、バーの高さを設定します。canvas_height 未満である必要があります
- <bar_width> 150 </ bar_width>数値、バー幅を設定します。canvas_width 未満である必要があります

```
<bar>
    <halpin>"my-bar"</halpin>
    <min_>0</min_>
    <max_>123</max_>
    <format>"3.1f"</format>
    <bgcolor>"grey"</bgcolor>
    <fillcolor>"red"</fillcolor>
    <range1>0,100,"green"</range1>
    <range2>101,135,"orange"</range2>
    <range3>136, 150,"red"</range3>
    <canvas_width>200</canvas_width>
    <canvas_height>50</canvas_height>
    <bar_height>30</bar_height>
    <bar_width>150</bar_width>
</bar>
```

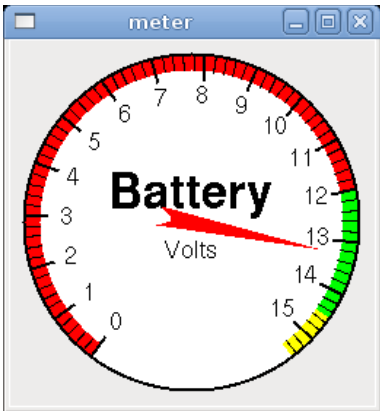
上記のコードはこの例を生成しました。



Meter Meter は、従来のダイヤルインジケーターを使用して FLOAT 信号の値を表示します。

```
<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>
```

上記のコードはこの例を生成しました。



7.1.6.7 数値入力

SpinboxSpinbox は FLOAT ピンを制御します。 矢印を押すか、スピンボックスをポイントしてマウスホイールを回転させることにより、ピンの値を増減します。 param_pin フィールドが TRUE（1）に設定されている場合、スピンボックスを初期値に設定し、HID 入力なしでその値をリモートで変更するために使用できるピンが作成されます。

```
<spinbox>
    <halpin>"my-spinbox"</halpin>
    <min_>-12</min_>
    <max_>33</max_>
```

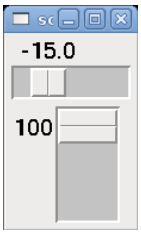
```
    <initval>0</initval>
    <resolution>0.1</resolution>
    <format>"2.3f"</format>
    <font>("Arial",30)</font>
    <param_pin>1</param_pin>
</spinbox>
```

上記のコードはこの例を生成しました。

スケールスケールは、フロートまたは s32 ピンを制御します。 ピンの値を増減するには、スライダーをドラッグするか、スケールをポイントしてマウスホイールを回転させます。 ハルピンには、float ピンと s32 ピンを形成するために-f と-i の両方が追加されます。 幅は、垂直方向のスライダーの幅と水平方向のスライダーの高さです。 param_pin フィールドが TRUE（1）に設定されている場合、スピンボックスを初期値に設定し、HID 入力なしでその値をリモートで変更するために使用できるピンが作成されます。

```
<scale>
    <font>("Helvetica",16)</font>
    <width>"25"</width>
    <halpin>"my-hscale"</halpin>
    <resolution>0.1</resolution>
    <orient>HORIZONTAL</orient>
    <initval>-15</initval>
    <min_>-33</min_>
    <max_>26</max_>
    <param_pin>1</param_pin>
</scale>
<scale>
    <font>("Helvetica",16)</font>
    <width>"50"</width>
    <halpin>"my-vscale"</halpin>
    <resolution>1</resolution>
    <orient>VERTICAL</orient>
    <min_>100</min_>
    <max_>0</max_>
    <param_pin>1</param_pin>
</scale>
```

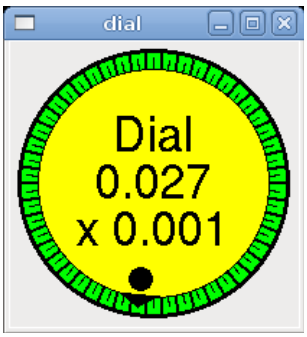
上記のコードはこの例を生成しました。



ダイヤルダイヤルは HAL フロートを出力し、マウスホイールとドラッグの両方に反応します。 ダブル左クリックして解像度を上げ、ダブル右クリックして解像度を 1 桁下げます。 出力は、最小値と最大値によって制限されます。 <cpr>は、リングの外側にある目盛りの数です（数字が大きいことに注意してください）。 param_pin フィールドが TRUE（1）に設定されている場合、スピンボックスを初期値に設定し、HID 入力なしでその値をリモートで変更するために使用できるピンが作成されます。

```
<dial>
    <size>200</size>
    <cpr>100</cpr>
    <min_>-15</min_>
    <max_>15</max_>
    <text>"Dial"</text>
    <initval>0</initval>
    <resolution>0.001</resolution>
    <halpin>"anaout"</halpin>
    <dialcolor>"yellow"</dialcolor>
    <edgecolor>"green"</edgecolor>
    <dotcolor>"black"</dotcolor>
    <param_pin>1</param_pin>
</dial>
```

上記のコードはこの例を生成しました。

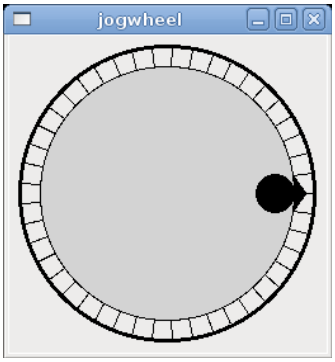


ジョグホイールジョグホイールは、円を描くようにドラッグするか、マウスホイールを回転させることにより、ホイールが回転するとカウントアップまたはカウントダウンする FLOAT ピンを出力することにより、実際のジョグホイールを模倣します。

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

```
    <size>250</size>
</jogwheel>
```

上記のコードはこの例を生成しました。



7.1.6.8 画像

画像表示は.gif 画像形式のみを使用します。 すべての画像は同じサイズである必要があります。 イメージは、ini ファイルと同じディレクトリ（または、コマンドラインから halrun / halcmd を使用して実行している場合は現在のディレクトリ）にある必要があります。

画像ビット image_bit は、ハルピンを true または false に設定することにより、2 つの画像を切り替えます。

<image name='fwd' file='fwd.gif'/>

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

この例は、上記のコードから作成されました。 2 つの画像ファイル fwd.gif と rev.gif を使用します。selectimage が false の場合は FWD が表示され、selectimage が true の場合は REV が表示されます。

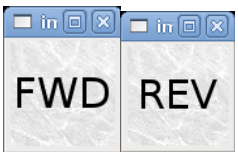
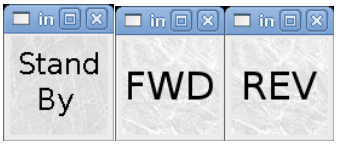


Image u32 image_u32 は image_bit と同じですが、基本的に無制限の数の画像があり、画像リストの最初の画像が 0、2 番目の画像が 1 などの整数値に halpin を設定して画像を選択します。

```
<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>
```

上記のコードは、stb.gif 画像を追加して次の例を作成しました



負の最小値がない限り、最大値より高く設定されていても、デフォルトは最小値であることに注意してください。

7.1.6.9 コンテナ

コンテナは、他のウィジェットを含むウィジェットです。 コンテナは、他のウィジェットをグループ化するために使用されます。

境界コンテナの境界は、2 つのタグを一緒に使用して指定されます。 <relief>タグは境界線のタイプを指定し、<bd>は境界線の幅を指定します。

- <relief> type </ relief> -type が FLAT、SUNKEN、RAISED、GROOVE、または RIDGE の場合
- <bd> n </ bd>-ここで、n は境界線の幅です。

```
<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>
```

上記のコードはこの例を生成しました。

塗りつぶしコンテナの塗りつぶしは、<boxfill fill = "" />タグで指定されます。 有効なエントリは、none、x、y、およびその両方です。 x 塗りつぶしは水平塗りつぶし、y 塗りつぶしは垂直塗りつぶしです

- <boxfill fill = "style" />-style が none、x、y、またはその両方の場合。 デフォルトは、Vbox の場合は x、Hbox の場合は y です。

アンカーコンテナアンカーは、<boxanchor アンカー= "" />タグで指定されます。 アンカーは、各スレーブをその区画内のどこに配置するかを指定します。 有効なエントリは、center、n、s、e、w、center、north、south、east、および west です。 sw、se、nw、ne などの組み合わせも有効です。

- <boxanchor アンカー= "position" />-位置は center、n、s、e、w、ne、nw、se、または sw です。 デフォルトは中央です。

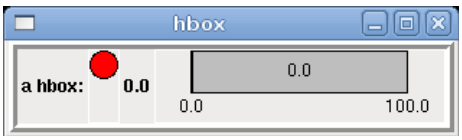
コンテナの展開 expand は、ブール値の<box expand expand = "" />タグで指定されます。 有効なエントリは「はい」、「いいえ」です。

- <boxexpand expand = "boolean" />-ブール値が yes または no の場合。 デフォルトは yes です。

Hbox ウィジェットを水平方向に並べてスタックする場合は、Hbox を使用します。

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

上記のコードはこの例を生成しました。

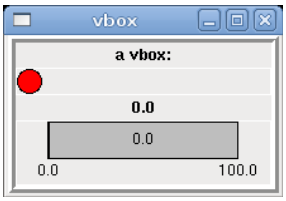


Hbox 内では、<boxfill fill = "" />、<boxanchor アンカー= "" />、および<boxexpand expand = "" />タグを使用して、ウィンドウが再表示されたときのボックス内のアイテムの動作を選択できます。 サイズ。Hbox のデフォルトは fill = "y"、anchor = "center"、expand = "yes"です。

Vbox ウィジェットを互いに垂直に積み重ねる場合は、Vbox を使用します。

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

上記のコードはこの例を生成しました。



Vbox 内では、<boxfill fill = "" />、<boxanchor アンカー= "" />、および<boxexpand expand = "" />タグを使用して、ウィンドウが再表示されたときのボックス内のアイテムの動作を選択できます。 サイズ。 Hbox のデフォルトは fill = "x"、anchor = "center"、expand = "yes"です。

ラベルフレームラベルフレームは、左上隅に溝とラベルが付いたフレームです。

```
<labelframe text="Group Title">
    <font>("Helvetica",16)</font>
    <hbox>
    <led/>
    <led/>
    </hbox>
</labelframe
```

上記のコードはこの例を生成しました。

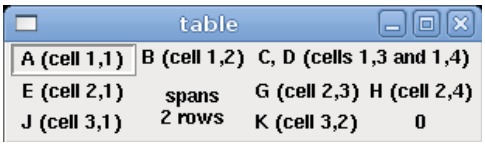
テーブルテーブルは、行と列のグリッドでのレイアウトを可能にするコンテナです。 各行は<tablerow />タグで始まります。 含まれているウィジェットは、<tablespan rows = cols = />タグを使用して、行または列にまたがることができます。 含まれているウィジェットが「スティック」するセルの側面は、<tablesticky sticky = />タグを使用して設定できます。 テーブルは、その柔軟な行と列を拡張します。

例：

```
<table flexible_rows="[2]" flexible_columns="[1,4]">
<tablesticky sticky="new"/>
```

```
<tablerow/>
    <label>
        <text>" A (cell 1,1) "</text>
        <relief>RIDGE</relief>
        <bd>3</bd>
    </label>
    <label text="B (cell 1,2)"/>
    <tablespan columns="2"/>
    <label text="C, D (cells 1,3 and 1,4)"/>
<tablerow/>
    <label text="E (cell 2,1)"/>
    <tablesticky sticky="nsew"/>
    <tablespan rows="2"/>
    <label text="'spans\n2 rows'"/>
    <tablesticky sticky="new"/>
    <label text="G (cell 2,3)"/>
    <label text="H (cell 2,4)"/>
<tablerow/>
    <label text="J (cell 3,1)"/>
    <label text="K (cell 3,2)"/>
    <u32 halpin="test"/>
</table>
```

上記のコードはこの例を生成しました。

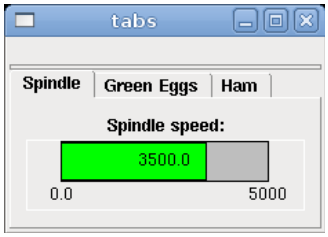


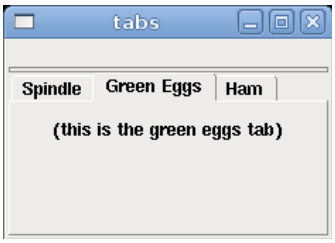
タブタブ付きインターフェースは、かなりのスペースを節約できます。

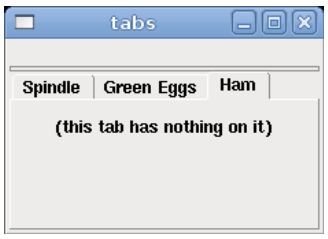
```
<tabs>
    <names> ["spindle","green eggs"]</names>
</tabs>
<tabs>
    <names>["Spindle", "Green Eggs", "Ham"]</names>
    <vbox>
        <label>
            <text>"Spindle speed:"</text>
        </label>
    <bar>
        <halpin>"spindle-speed"</halpin>
        <max_>5000</max_>
```

```
    </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>
```

上記のコードは、選択された各タブを示すこの例を生成しました。







7.2 PyVCP の例

7.2.1 AXIS

AXIS の右側に接続されている AXIS インターフェイスで使用する PyVCP パネルを作成するには、次の基本的な操作を行う必要があります。

- パネルの説明を含む.xml ファイルを作成し、それを config ディレクトリに置きます。
- PyVCP エントリを ini ファイルの[DISPLAY]セクションに.xml ファイル名で追加します。

- POSTGUI_HALFILE エントリを、postguiHAL ファイル名の名前で ini ファイルの[HAL]セクションに追加します。
- パネルの HAL ピンへのリンクを postgui.hal ファイルに追加して、PyVCP パネルを LinuxCNC に接続します。

7.2.2 フローティングパネル

任意のインターフェイスで使用できるフローティング PyVCP パネルを作成するには、次の基本的なことを行う必要があります。

- パネルの説明を含む.xml ファイルを作成し、それを config ディレクトリに置きます。
- loadusr 行を.hal ファイルに追加して、各パネルをロードします。
- パネルの HAL ピンへのリンクを postgui.hal ファイルに追加して、PyVCP パネルを LinuxCNC に接続します。

以下は、2 つの PyVCP パネルをロードし、HAL の接続名がわかるようにそれぞれに名前を付ける loadusr コマンドの例です。

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

-Wn は、続行する前に HAL に名前がロードされるのを待機させます。 pyvcp -c は、PyVCP にパネルの名前を付けます。

panel1.xml の HAL ピンには btnpanel。<pinname>という名前が付けられます。

panel2.xml の HAL ピンには sppanel という名前が付けられます。<ピン名>

loadusr 行が、PyVCP ピンを使用するネットの前にあることを確認してください。

7.2.3 ジョグボタン

この例では、X、Y、および Z のジョグボタンを備えた PyVCP パネルを作成します。この構成は、Stepconf ウィザードで生成された構成に基づいて構築されます。 最初に Stepconf ウィザードを実行してマシンを構成し、次に[Advanced Configuration Options]ページで、次の図に示すように、空の PyVCP パネルを追加するためのいくつかの選択を行います。 この例では、Stepconf ウィザードの[Basic MachineInformation]ページで構成に pyvcp_xyz という名前を付けました。

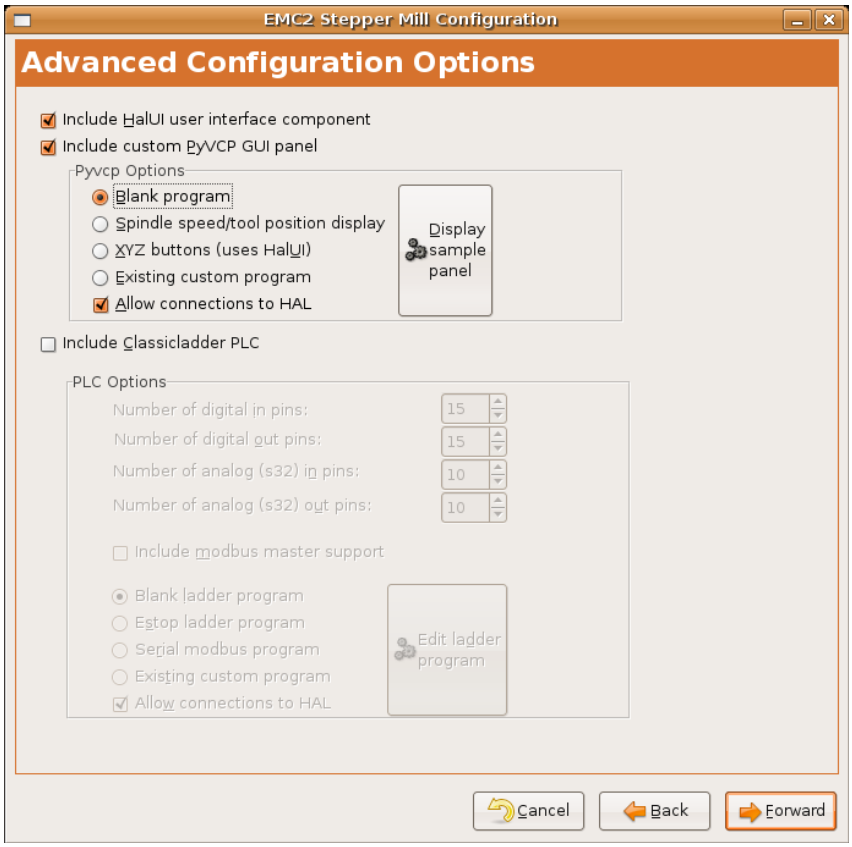


図 9-69

Stepconf ウィザードはいくつかのファイルを作成し、それらを linuxcnc / configs / pyvcp_xyz ディレクトリに配置します。 [リンクの作成]をオンのままにすると、デスクトップ上のそれらのファイルへのリンクが表示されます。

1.1.1.1 ウィジェットを作成する

custompanel.xml ファイルを右クリックし、[テキストエディタで開く]を選択して開きます。 <pyvcp> </pyvcp>タグの間に、パネルのウィジェットを追加します。

各ウィジェットの詳細については、マニュアルの「PyVCP ウィジェットリファレンス」セクションを参照してください。

custompanel.xml ファイルに、ウィジェットの説明を追加します。

```
<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>
```

上記を追加すると、AXIS の右側に次のような PyVCP パネルが取り付けられます。 見た目は良いですが、ボタンを halui に接続するまで何もしません。 ポップアップウィンドウの一番下までスクロールして実行しようとしたときにエラーが発生した場合、通常、エラーはスペルまたは構文エラーであり、そこに表示されます。

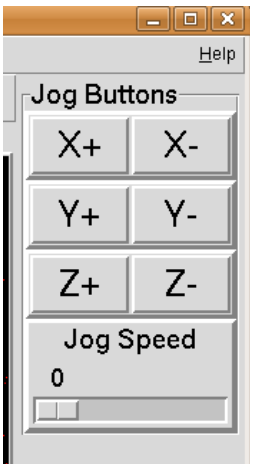


図 9-70

7.2.3.1 接続する

必要な接続を確立するには、custom_postgui.hal ファイルを開き、以下を追加します。

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

非常停止をリセットしてジョグモードにし、PyVCP パネルのジョグ速度スライダーをゼロより大きい値に移動すると、PyVCP ジョグボタンが機能するはずです。 g コードファイルの実行中、一時停止中、または[MDI]タブが選択されている間は、ジョギングできません。

7.2.4 ポートテスター

この例は、PyVCP と HAL を使用して単純なパラレルポートテスターを作成する方法を示しています。

まず、次のコードを使用して ptest.xml ファイルを作成し、パネルの説明を作成します。

```
<!-- 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>
```

これにより、2 つのインピンと 2 つのアウトピンを含む次のフローティングパネルが作成されます。

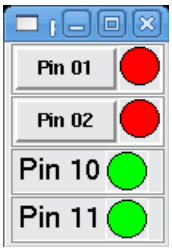


図 9-71

すべてを起動して実行するために必要な HAL コマンドを実行するには、ptest.hal ファイルに次のコマンドを入れます。

```
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
```

HAL ファイルを実行するには、ターミナルウィンドウから次のコマンドを使用します。

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

次の図は、完全なパネルがどのように見えるかを示しています。

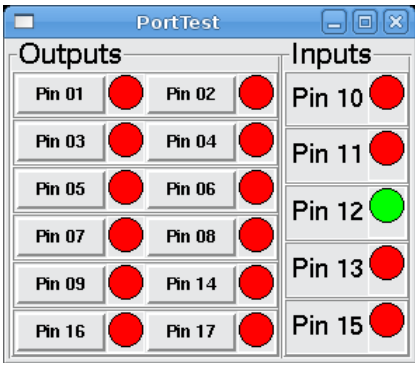


図 9-72

残りのパラレルポートピンを追加するには、.xml ファイルと.hal ファイルを変更するだけです。

HAL スクリプトの実行後にピンを表示するには、halcmd プロンプトで次のコマンドを使用します。

```
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
```

これにより、どのピンが IN で、どのピンが OUT であるか、および接続が表示されます。

7.2.5 GS2RPM メーター

次の例では、Automation Direct GS2 VDF ドライバーを使用し、RPM およびその他の情報を PyVCP パネルに表示します。 この例は、このマニュアルの「ハードウェアの例」セクションにある GS2 の例に基づいています。

1.1.1.1 パネル

パネルを作成するには、以下を.xml ファイルに追加します。

```
<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>
```

上記のように、次のような PyVCP パネルが表示されます。

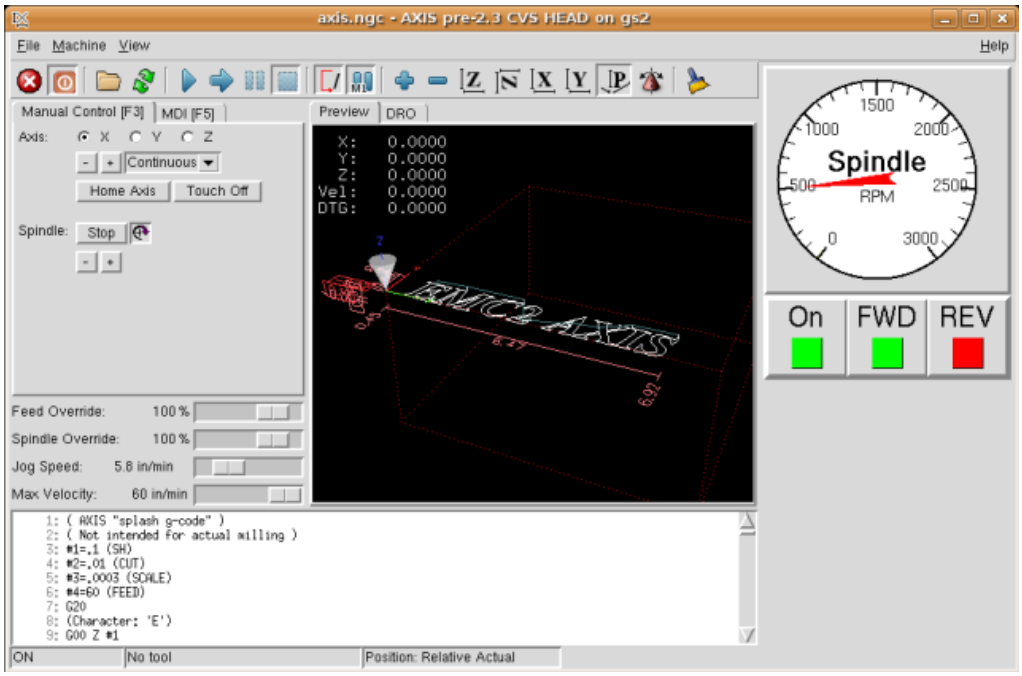


図 9-73

7.2.5.1 接続

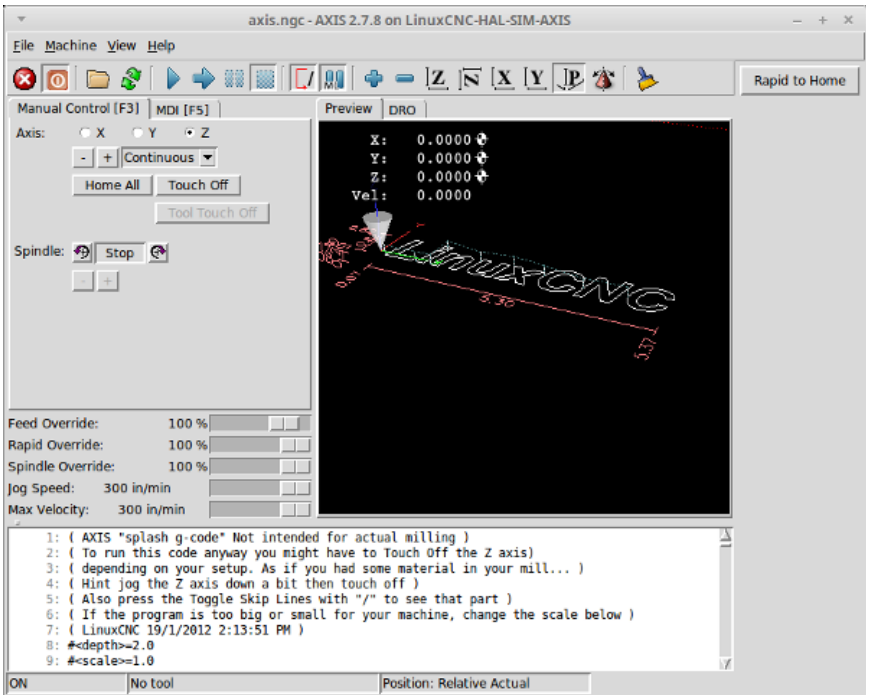
これを機能させるために、custom_postgui.hal ファイルに次のコードを追加します。

```
# 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
```

一部の行には説明が必要な場合があります。 fwd led 行は custom.hal ファイルで作成された信号を使用しますが、revled はスピンドル-rev ビットを使用する必要があります。 スピンドル fwd ビットを 2 回リンクすることはできないため、リンクされた信号を使用します。

7.2.6 ラピッドトゥホームボタン

この例では、押すと PyVCP サイドパネルにボタンが作成され、すべての軸がホームポジションに戻ります。この例では、PyVCP パネルがないことを前提としています。



構成ディレクトリに.xml ファイルを作成します。 この例では、rth.xml という名前です。 rth.xml ファイルに、次のコードを追加してボタンを作成します。

```
<pyvcp>
<!-- rapid to home button example -->
<button>
<halpin>"rth-button"</halpin>
<text>"Rapid to Home"</text>
</button>
</pyvcp>
```

テキストエディタで.ini ファイルを開き、[DISPLAY]セクションに次の行を追加します。 これが PyVCP パネルをロードするものです。

```
PYVCP = rth.xml
```

ini ファイルに[HALUI]セクションがない場合は、それを作成して、次の MDI コマンドを追加します。

```
MDI_COMMAND = G53 G0 X0 Y0 Z0
```

Note

G53 および G0G コードに関する情報

[HAL]セクションで、GUI 後のファイルがない場合は、以下を追加して、postgui.hal というファイルを作成します。

```
POSTGUI_HALFILE = postgui.hal
```

postgui.hal ファイルに次のコードを追加して、PyVCP ボタンを MDI コマンドにリンクします。

```
net rth halui.mdi-command-00 <= pyvcp.rth-button
```

note

7.3 Glade 仮想コントロールパネル

7.3.1 GladeVCP とは何ですか？

GladeVCP は LinuxCNC コンポーネントであり、次のような LinuxCNC ユーザーインターフェイスに新しいユーザーインターフェイスパネルを追加する機能を追加します。

-Axis
-Touchy
-Gscreen
-Gmoccapy

PyVCP とは異なり、GladeVCP は HAL ピンの表示と設定に限定されません。Python コードで任意のアクションを実行できるためです。実際、GladeVCP と Python を使用して完全な LinuxCNC ユーザーインターフェイスを構築できます。

GladeVCP は、Glade WYSIWYG ユーザーインターフェイスエディターを使用します。これにより、視覚的に心地よいパネルを簡単に作成できます。 これは、豊富な GTK +ウィジェットセットへの PyGTK バインディングに依存しており、実際、これらはすべて、GladeVCP アプリケーションで使用できます。HAL および LinuxCNC と対話するための専用ウィジェットだけでなく、ここに記載されています。

1.1.1.1 一目で PyVCP 対 GladeVCP

どちらも、HAL ウィジェットを使用したパネルの作成をサポートしています。LED、ボタン、スライダーなどのユーザーインターフェイス要素の値は HAL ピンにリンクされており、HAL ピンは LinuxCNC の残りの部分にインターフェイスします。

PyVCP：

- ウィジェットセット：TkInter ウィジェットを使用
- ユーザーインターフェイスの作成：「XML ファイルの編集/結果の実行/外観の評価」サイクル
- ユーザー定義のイベント処理の埋め込みはサポートされていません
- HAL ピン I / O 以外の LinuxCNC インタラクションはサポートされていません

GladeVCP：

- ウィジェットセット：GTK +ウィジェットセットに依存します。
- ユーザーインターフェイスの作成：GladeWYSIWYG ユーザーインターフェイスエディターを使用します
- HAL ピンの変更は、ユーザー定義の Python イベントハンドラーにコールバックするように指示できます。
- GTK シグナル（キー/ボタンの押下、ウィンドウ、I / O、タイマー、ネットワークイベント）は、Python のユーザー定義ハンドラーに関連付けることができます

- LinuxCNC の直接対話：G コードサブルーチンを呼び出すための MDI コマンドの開始などの任意のコマンド実行、およびアクションウィジェットによるステータス変更操作のサポート
- いくつかの独立した GladeVCP パネルを異なるタブで実行できます
- ユーザーインターフェイスの外観と機能の分離：コードに触れることなく外観を変更する

7.3.2 サンプルパネルを使用したクイックツアー

GladeVCP パネルウィンドウは、次の 3 つの異なるセットアップで実行できます。

- PyVCP パネルとまったく同じように、右側の Axis に統合されて常に表示されます
- Axis、Touchy、Gscreen、または Gmoccapy のタブとして。 Axis では、これにより、明示的に上げる必要がある[プレビュー]タブと[DRO]タブのほかに 3 番目のタブが作成されます。
- メインウィンドウとは独立してアイコン化/非アイコン化できるスタンドアロンのトップレベルウィンドウとして。

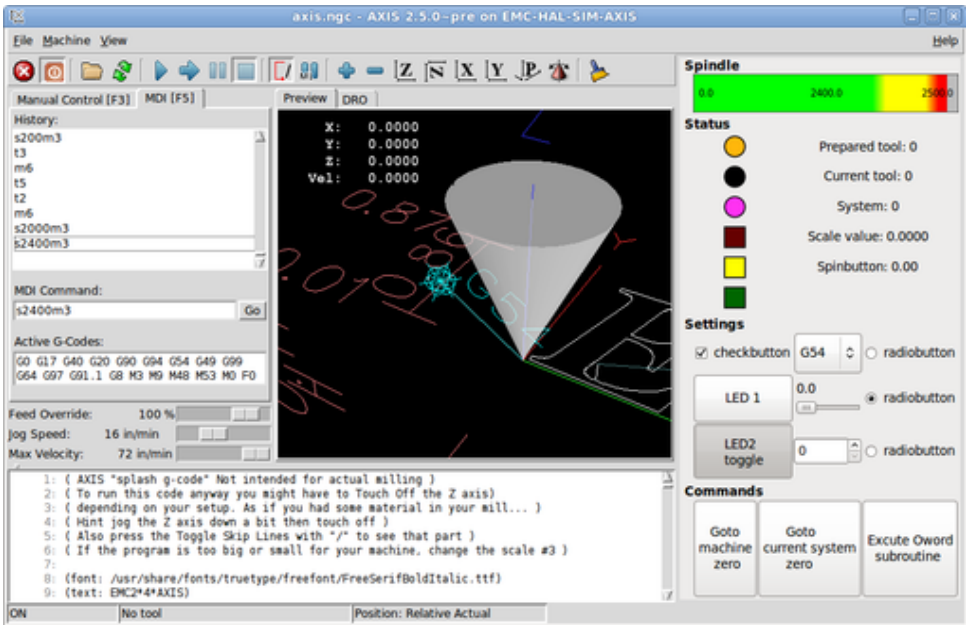
インストールされた LinuxCNC インストールされたバージョンの LinuxCNC を使用している場合、以下に示す例は、[サンプル構成]> [アプリ]> [gladevcp]ブランチの構成ピッカーにあります。

Git チェックアウト次の手順は、git チェックアウトを使用している場合にのみ適用されます。 ターミナルを開き、git によって作成されたディレクトリに移動して、図のようにコマンドを発行します。

> **Note**
>
> 次のコマンドを gitcheckout で機能させるには、最初に make を実行し、次に sudo make setuid を実行してから、を実行する必要があります。 ./scripts/rip-environment。 git チェックアウトの詳細については、linuxcncwiki ページを参照してください。

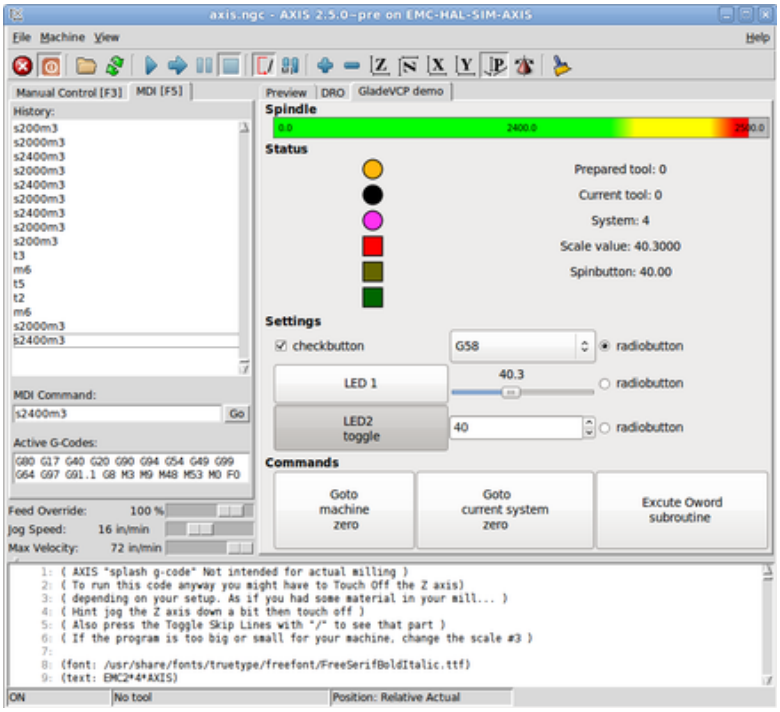
次のように、PyVCP のように Axis に統合されたサンプル GladeVCP パネルを実行します。

```
$ cd configs/sim/axis/gladevcp
$ linuxcnc gladevcp_panel.ini
```



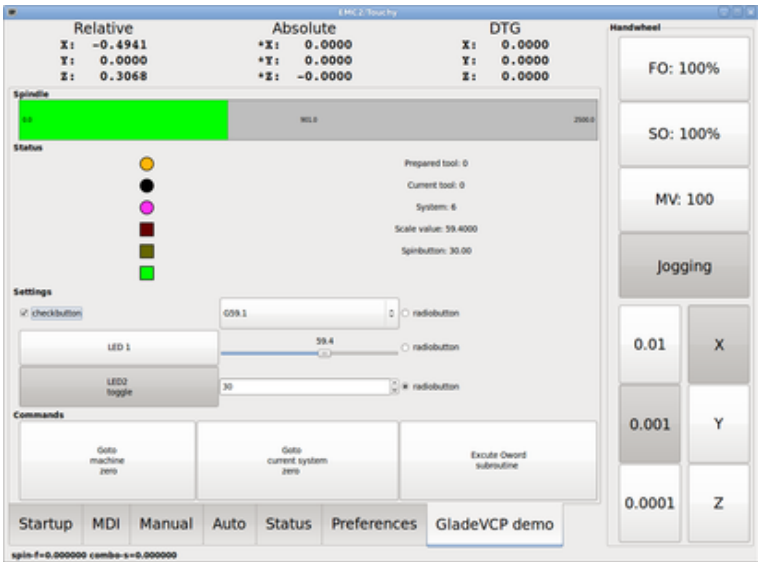
同じパネルを実行しますが、Axis 内のタブとして実行します。

```
$ cd configs/sim/axis/gladevcp
$ linuxcnc gladevcp_tab.ini
```



Touchy 内でこのパネルを実行するには：

```
$ cd configs/sim/touchy/gladevcp
$ linuxcnc gladevcp_touchy.ini
```



機能的には、これらの設定は同じです。画面のスペース要件と可視性のみが異なります。 複数の GladeVCP コンポーネントを（異なる HAL コンポーネント名で）並行して実行できるため、混合セットアップも可能です。たとえば、右側のパネルや、インターフェイスの使用頻度の低い部分用の 1 つ以上のタブなどです。 。

1.1.1.1 サンプルパネルの探索

configs / sim / axis /gladevcp_panel.ini または configs / sim / axis / gladevcp_tab.ini を実行しているときに、[HAL 構成の表示]を調べます。gladevcpHAL コンポーネントが見つかり、パネル内のウィジェットを操作しているときにピン値を確認できます。 HAL の設定は、configs / axis / gladevcp /manual-example.hal にあります。

サンプルパネルの下部には 2 つのフレームがあります。 パネルは、ESTOP をリセットすると設定フレームがアクティブになり、マシンの電源をオンにすると下部のコマンドフレームが有効になるように構成されています。 設定フレームの HAL ウィジェットは、ステータスフレームの LED とラベル、および現在準備されているツール番号にリンクされています。それらを試して効果を確認してください。 MDI ウィンドウで T <toolnumber>および M6 コマンドを実行すると、現在および準備されているツール番号フィールドが変更されます。

コマンドフレームのボタンは MDI アクションウィジェットです。ボタンを押すと、インタープリターで MDI コマンドが実行されます。 3 番目のボタン[Oword サブルーチンの実行]は高度な例です。設定フレームからいくつかの HAL ピン値を取得し、それらをパラメーターとして Oword サブルーチンに渡します。 ルーチンが受け取った実際のパラメーターは、（DEBUG、）コマンドによって表示されます。サブルーチン本体については、.. / .. / nc_files /oword.ngc を参照してください。

パネルが Axis にどのように統合されているかを確認するには、configs / sim / axis / gladevcpgladevcp_panel.ini の[DISPLAY] GLADEVCP ステートメント、configs / sim / axis / gladevcp /gladevcp_tab.ini の[DISPLAY] EMBED *ステートメントを参照してください。 configs / sim / axis / gladevcp /gladevcp_tab.ini と configs / sim / axis / gladevcp /gladevcp_panel.ini の両方の[HAL] POSTGUI_HALFILE ステートメント

7.3.2.1 ユーザーインターフェイスの説明を調べる

ユーザーインターフェイスは、glade UI エディターを使用して作成されます。これを探索するには、glade をインストールする必要があります。 ユーザーインターフェイスを編集するには、コマンドを実行します

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

（必要な glade プログラムは、最近のシステムでは glade-gtk2 という名前になる場合があります。）

中央のウィンドウには、UI の外観が表示されます。 すべてのユーザーインターフェイスオブジェクトとサポートオブジェクトは右上のウィンドウにあり、特定のウィジェットを選択できます（または中央のウィンドウでウィジェットをクリックします）。 選択したウィジェットのプロパティが右下のウィンドウに表示され、変更できます。

MDI コマンドが MDI アクションウィジェットからどのように渡されるかを確認するには、右上のウィンドウの[アクション]の下、および右下のウィンドウの[全般]タブの[MDI コマンド]プロパティにリストされているウィジェットを調べます。

7.3.2.2 Python コールバックの調査

Python コールバックが例にどのように統合されているかをご覧ください。

- 空き地では、ヒットラベルウィジェット（プレーンな GTK +ウィジェット）を参照してください
- button1 ウィジェットで、[シグナル]タブを確認し、ハンドラー on_button_press に関連付けられている押されたシグナルを見つけます。
- hitcounter.py で、メソッド on_button_press を参照し、それが hits オブジェクトの label プロパティをどのように設定するかを確認します。

これは概念に触れているだけです。コールバックメカニズムについては、GladeVCP プログラミングのセクションで詳しく説明します。

7.3.3 Glade ユーザーインターフェイスの作成と統合

1.1.1.1 前提条件：Glade のインストール

Glade UI ファイルを表示または変更するには、glade3.8.0 をインストールする必要があります。GladeVCP パネルを実行するだけでは必要ありません。 glade コマンドがない場合は、次のコマンドを使用してインストールします。

```
$ sudo apt-get install glade-gtk2
```

バージョン番号が 3.8.0 以下であることを確認します

```
$ glade-gtk2 --version
glade3 3.8.0
```

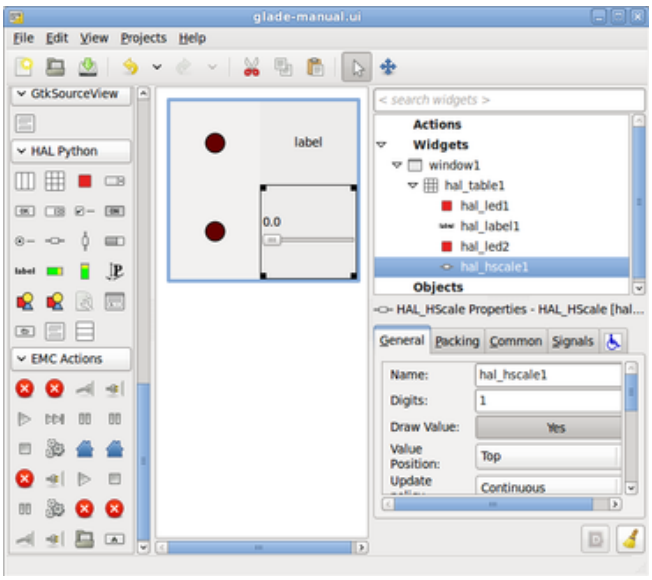
7.3.3.1 Glade を実行して新しいユーザーインターフェイスを作成する

このセクションでは、LinuxCNC 固有の最初の手順の概要を説明します。 glade の詳細とチュートリアルについては、http：//glade.gnome.org を参照してください。 いくつかの空き地のヒントとコツも YouTube で見つけることができます。

glade <file> .ui を実行して既存の UI コンポーネントを変更するか、シェルから glade コマンドを実行するだけで新しい UI コンポーネントを開始します。

- LinuxCNC がパッケージからインストールされていない場合は、LinuxCNC シェル環境をで設定する必要があります。 <linuxcncdir> / scripts /それ以外の場合、glade は LinuxCNC 固有のウィジェットを見つけられません。
- 保存されていない設定を求められたら、デフォルトを受け入れて[閉じる]をクリックします。
- 保存されていない設定を求められたら、デフォルトを受け入れて[閉じる]をクリックします。
- トップレベル（左側のペイン）から、ウィンドウ（最初のアイコン）をトップレベルウィンドウとして選択します。デフォルトでは、window1 という名前になります。 この名前は変更しないでください-GladeVCP はそれに依存しています。
- 左側のタブで、下にスクロールして[HALPython および VCP アクション]を展開します。
- HALPython の HAL_Box や HAL_Table などのコンテナをフレームに追加します
- LED、ボタンなどの要素を選択してコンテナ内に配置します

これは次のようになります。



Glade は、シェルウィンドウに多くのメッセージを書き込む傾向がありますが、ほとんどの場合無視できます。[ファイル]、[名前を付けて保存]の順に選択し、myui.ui のような名前を付けて、GtkBuilder ファイルとして保存されていることを確認します（[保存]ダイアログのラジオボタンの左下隅）。 GladeVCP も古い libglade 形式を正しく処理しますが、それを使用しても意味がありません。 GtkBuilder ファイル拡張子の規則は.ui です。

7.3.3.2 パネルのテスト

これで、次の方法で試してみる準備ができました（LinuxCNC、たとえば Axis の実行中）。

```
gladevcp myui.ui
```

GladeVCP は、-c <component name>オプションでオーバーライドされない限り、UI ファイルのベース名（この場合は myui）のような名前の HAL コンポーネントを作成します。 Axis を実行している場合は、Show HAL configuration を試して、そのピンを調べてください。

HAL_Hbox または HAL_Table を含むウィジェットがグレー表示（非アクティブ）に表示されるのはなぜか疑問に思われるかもしれません。 HAL コンテナには、デフォルトでオフになっている HAL ピンが関連付けられています。これにより、含まれているすべてのウィジェットが非アクティブになります。 一般的な使用例は、これらのコンテナ HAL ピンを halui.machine.is-on または halui.mode の 1 つに関連付けることです。 一部のウィジェットが特定の状態でのみアクティブに表示されるようにするためのシグナル。

コンテナをアクティブ化するには、HAL コマンド setpgladevcp。<container-name> 1 を実行します。

7.3.3.3 HAL コマンドファイルの準備

GladeVCP パネルで HAL ピンをリンクするための推奨される方法は、拡張子が.hal の別のファイルにそれらを収集することです。 このファイルは、ini ファイルの HAL セクションにある POSTGUI_HALFILE =オプションを介して渡されます。

警告

GladeVCPHAL コマンドファイルを Axis [HAL] HALFILE = ini セクションに追加しないでください。これにより、目的の効果が得られません。次のセクションを参照してください。

7.3.3.4 PyVCP のように Axis に統合する

ini ファイルで次のように指定して、GladeVCP パネルを右側のパネルに配置します。

```
[DISPLAY]
# add GladeVCP panel where PyVCP used to live:
GLADEVCP= -u ./hitcounter.py ./manual-example.ui
[HAL]
# HAL commands for GladeVCP components in a tab must be executed via
POSTGUI_HALFILE
POSTGUI_HALFILE = ./manual-example.hal
[RS274NGC]
# gladevcp Demo specific Oword subs live here
SUBROUTINE_PATH = ../../nc_files/gladevcp_lib
```

GLADEVCP オプションで開始された GladeVCP アプリケーションのデフォルトの HAL コンポーネント名は、gladevcp です。

上記の構成で Axis によって実際に実行されるコマンドラインは次のとおりです。

```
halcmd loadusr -Wn gladevcp gladevcp -c gladevcp -x {XID} -u ./hitcounter.py ./  -
manual-example.ui
```

上記のコマンドラインオプションと衝突しない限り、ここに任意の gladevcp オプションを追加できます。

-c オプションを追加することにより、カスタム HAL コンポーネント名を作成できます。

```
[DISPLAY]
# add GladeVCP panel where PyVCP used to live:
GLADEVCP= -c example -u ./hitcounter.py ./manual-example.ui
```

上記の Axis によって実際に実行されるコマンドラインは次のとおりです。

```
halcmd loadusr -Wn example gladevcp -c example -x {XID} -u ./hitcounter.py ./  -
manual-example.ui
```

---

note

./hitcounter.py、./manual-example.ui などのファイル指定子は、ファイルが ini ファイルと同じディレクトリにあることを示します。 それらをディレクトリにコピーする必要がある場合があります（または、ファイルへの正しい絶対パスまたは相対パスを指定します）

Note
[RS274NGC] SUBROUTINE_PATH =オプションが設定されているだけなので、サンプルパネルは MDI コマンドウィジェットの Oword サブルーチン（oword.ngc）を検索します。 セットアップでは必要ない場合があります。 相対パス指定子../../nc_files/gladevcp_lib は、構成ピッカーによってコピーされたディレクトリを処理するように構築されており、インプレースセットアップを使用します。

7.3.3.5 タブとして埋め込む

これを行うには、.ini ファイルを編集し、次のように ini ファイルの DISPLAY セクションと HAL セクションに追加します。

```
[DISPLAY]
# add GladeVCP panel as a tab next to Preview/DRO:
EMBED_TAB_NAME=GladeVCP demo
```

```
EMBED_TAB_COMMAND=halcmd loadusr -Wn gladevcp gladevcp -c gladevcp -x {XID} -u
./gladevcp/  -
hitcounter.py ./gladevcp/manual-example.ui
[HAL]
# HAL commands for GladeVCP components in a tab must be executed via
POSTGUI_HALFILE
POSTGUI_HALFILE = ./gladevcp/manual-example.hal
[RS274NGC]
# gladevcp Demo specific Oword subs live here
SUBROUTINE_PATH = ../../nc_files/gladevcp_lib
```

tab コマンドを開始する halcmdloadusr の方法に注意してください。これにより、POSTGUI_HALFILE が HAL コンポーネントの準備ができた後にのみ実行されることが保証されます。 まれに、タブを使用しているが HAL コンポーネントが関連付けられていないコマンドをここで実行する場合があります。 このようなコマンドは、halcmd loadusr なしで開始できます。これは、HAL コンポーネントがないため、HAL コンポーネントを待機する必要がないことを Axis に示します。

上記の例でコンポーネント名を変更する場合、-Wn <component>と-c <component>で使用される名前は同一でなければならないことに注意してください。

Axis を実行して試してみてください。DRO タブの近くに GladeVCP デモと呼ばれる新しいタブがあるはずです。そのタブを選択すると、サンプルパネルが Axis 内にうまく収まるはずです。

Note

UI ファイルが GLADEVCP =ステートメントと EMBED_TAB_COMMAND =ステートメントの両方で GladeVCP に渡される最後のオプションであることを確認してください。

7.3.3.6 Touchy への統合

Touchy に GladeVCP タブを追加するには、次のように.ini ファイルを編集します。

```
[DISPLAY]
# add GladeVCP panel as a tab
EMBED_TAB_NAME=GladeVCP demo
EMBED_TAB_COMMAND=gladevcp -c gladevcp -x {XID} -u ./hitcounter.py -H ./gladevcp-
touchy.hal  -
./manual-example.ui
[RS274NGC]
# gladevcp Demo specific Oword subs live here
SUBROUTINE_PATH = ../../nc_files/gladevcp_lib
```

Note

./hitcounter.py、./manual-example.ui などのファイル指定子は、ファイルが ini ファイルと同じディレクトリにあることを示します。 それらをディレクトリにコピーする必要がある場合があります（または、ファイルへの正しい絶対パスまたは相対パスを指定します）

[軸]タブの設定との次の違いに注意してください。

- Touchy は halui コンポーネントを使用しないため、HAL コマンドファイルはわずかに変更されています。そのため、その信号は使用できず、いくつかのショートカットが使用されています。
- POSTGUI_HALFILE = ini オプションはありませんが、EMBED_TAB_COMMAND =行で HAL コマンドファイルを渡すことは問題ありません。
- halcmd loaduser-Wn。 。 。 呪文は必要ありません。

7.3.4 GladeVCP コマンドラインオプション

mangladevcp も参照してください。 gladevcp コマンドラインオプションは次のとおりです。

```
Usage: gladevcp [options] myfile.ui
Options:
-h, --help
show this help message and exit
-c NAME
Set component name to NAME. Default is base name of UI file
-d
Enable debug output
-g GEOMETRY
Set geometry WIDTHxHEIGHT+XOFFSET+YOFFSET. Values are in pixel units,
XOFFSET/YOFFSET is referenced
from top left of screen. Use -g WIDTHxHEIGHT for just setting size or -g +XOFFSET+YOFFSET
for just position
-H FILE
execute hal statements from FILE with halcmd after the component is set up and ready
-m MAXIMUM
force panel window to maximize. Together with the -g geometry option one can move the
panel to a second monitor and
force it to use all of the screen
-t THEME
set gtk theme. Default is system theme. Different panels can have different themes. An
example theme can be found in the
EMC Wiki.
-x XID
Re-parent GladeVCP into an existing window XID instead of creating a new top level window
```

-u FILE Use File's as additional user defined modules with handlers -U USEROPT pass USEROPTs to Python modules

7.3.5 gladeVCP の起動プロセスを理解する

上で概説した統合手順は少し注意が必要ですが、そうです。 したがって、LinuxCNC の起動プロセスと、これが gladeVCP にどのように関連しているかを理解するのに役立ちます。

通常の LinuxCNC 起動プロセスは、次のことを行います。

- リアルタイム環境が開始されます
- すべての HAL コンポーネントがロードされます
- HAL コンポーネントは、.halcmd スクリプトを介して相互にリンクされています
- タスク、iocontrol、そして最終的にはユーザーインターフェイスが開始されます
- pre-gladeVCP の前提は、UI が開始するまでに、すべての HAL が読み込まれ、配管され、準備ができていることです。

gladeVCP の導入により、次の問題が発生しました。

- gladeVCP パネルは、マスター GUI ウィンドウのセットアップに埋め込む必要があります。 Axis、または Touchy、Gscreen、または Gmoccapy（埋め込みウィンドウまたは埋め込みタブとして）
- これには、gladeVCP ウィンドウをマスター GUI にフックする前に、マスター GUI を実行する必要があります。
- ただし、gladeVCP も HAL コンポーネントであり、独自の HAL ピンを作成します。
- 結果として、ソースまたは宛先として gladeVCP HAL ピンを含むすべての HAL 配管は、GUI がセットアップされた後に実行する必要があります。

これが POSTGUI_HALFILE の目的です。 この ini オプションは、GUI によって検査されます。 GUI がこのオプションを検出すると、埋め込まれた gladVCP パネルが設定された後、対応する HAl ファイルが実行されます。 ただし、gladeVCP パネルが実際に使用されているかどうかはチェックされません。この場合、HALcmd ファイルは正常に実行されます。 したがって、GLADEVCP や EMBED_TAB などを介して gladeVCP を開始せず、後で別のシェルウィンドウまたはその他のメカニズムで開始した場合、POSTGUI_HALFILE の HAL コマンドファイルの実行が早すぎます。 ここで gladeVCP ピンが参照されているとすると、これは失敗し、gladeVCPHAL コンポーネントが使用できないことを示すエラーメッセージが表示されます。

したがって、別のシェルウィンドウから gladeVCP を実行する場合（つまり、GUI によって埋め込まれた方法で開始されない場合）：

- POSTGUI_HALFILE ini オプションに依存して、HAL コマンドが適切な時点で実行されるようにすることはできないため、ini ファイルにコメントアウトしてください。
- -H <halcmd file>オプションを使用して、gladeVCP ピンを参照する HAL コマンドファイルを gladeVCP に明示的に渡します（前のセクションを参照）。

7.3.6 HAL ウィジェットリファレンス

GladeVcp には、HAL ウィジェットと呼ばれる HAL ピンが接続された Gtk ウィジェットのコレクションが含まれており、LinuxCNC HAL レイヤーを制御、表示、またはその他の方法で操作することを目的としています。これらは、Glade ユーザーインターフェイスエディターで使用することを目的としています。適切にインストールすると、HAL ウィジェットが Glade の HALPython ウィジェットグループに表示されます。Glade General セクションの多くの HAL 固有のフィールドには、マウスオーバーツールチップが関連付けられています。

HAL 信号には、ビットと数値の 2 つのバリエーションがあります。ビットはオフ/オン信号です。番号は「float」、「s32」、または「u32」にすることができます。HAL データ型の詳細については、HAL マニュアルを参照してください。GladeVcp ウィジェットは、インジケーターウィジェットを使用してシグナルの値を表示するか、コントロールウィジェットを使用してシグナル値を変更することができます。したがって、HAL 信号に接続できる GladeVcp ウィジェットには 4 つのクラスがあります。別のクラスのヘルパーウィジェットを使用すると、パネルを整理してラベルを付けることができます。

- 「ビット」信号を示すためのウィジェット：HAL_LED
- 「ビット」信号を制御するためのウィジェット：HAL_Button HAL_RadioButton HAL_CheckButton
- 「番号」信号を示すためのウィジェット：HAL_Label、HAL_ProgressBar、HAL_HBar および HAL_VBar、HAL_Meter
- 「数値」信号を制御するためのウィジェット：HAL_SpinButton、HAL_HScale および HAL_VScale、ジョグホイール、速度制御
- 機密性の高いコントロールウィジェット：State_Sensitive_TableHAL_Table および HAL_HBox
- ツールパスプレビュー：HAL_Gremlin
- 軸の位置を表示するウィジェット：DRO ウィジェット、CombiDRO ウィジェット
- ファイル処理用のウィジェット：IconView ファイルの選択
- すべての軸オフセットを表示/編集するためのウィジェット：OffsetPage
- すべてのツールオフセットを表示/編集するためのウィジェット：ツールオフセットエディタ
- Gcode の表示と編集用のウィジェット：HAL_Sourceview
- MDI 入力および履歴表示用のウィジェット：MDI 履歴

1.1.1.1 ウィジェットと HAL ピンの命名

ほとんどの HAL ウィジェットには、ウィジェットと同じ HAL 名（glade：General！Name）を持つ単一の HAL ピンが関連付けられています。 現在、この規則の例外はあります。

- HAL_Spinbutton と HAL_ComboBox には、<widgetname> -f（float）ピンと<widgetname> -s（s32）ピンの 2 つのピンがあります。
- HAL_ProgressBar。<widgetname>値の入力ピンと<widgetname>スケールの入力ピンがあります。

7.3.6.1 Python の属性と HAL ウィジェットのメソッド

HAL ウィジェットは GtKWidgets のインスタンスであるため、該当する GtkWidget クラスのメソッド、プロパティ、およびシグナルを継承します。 たとえば、HAL_Button が持つ GtkWidget 関連のメソッド、プロパティ、およびシグナルを特定するには、PyGtk リファレンスマニュアルで GtkButton の説明を参照してください。

特定の HAL ウィジェットの継承関係を見つける簡単な方法は、次のとおりです。glade を実行し、ウィジェットをウィンドウに配置して、それを選択します。 次に、[プロパティ]ウィンドウで[信号]タブを選択します。 たとえば、HAL_LED ウィジェットを選択すると、HAL_LED が GtkWidget から派生し、GtkObject が GtkObject から派生し、最終的には GObject から派生することが示されます。

HAL ウィジェットには、HAL 固有の Python 属性もいくつかあります。

hal_pin

ウィジェットに単一のピンタイプがある場合の基になる HAL ピン Python オブジェクト

hal_pin_s, hal_pin_f

HAL_Spinbutton ウィジェットと HAL_ComboBox ウィジェットの S32 ピンと float ピン-これらのウィジェットには hal_pin 属性がないことに注意してください。

hal_pin_scale

入力の最大絶対値を表す HAL_ProgressBar ウィジェットの float 入力ピン。

HAL ウィジェットにはいくつかの HAL 固有の方法がありますが、関連する方法は次のとおりです。

<halpin>.get()

現在の HAL ピンの値を取得します。ここで、<halpin>は上記の該当する HAL ピン名です。

7.3.6.2 ピンとウィジェットの値の設定

原則として、Python コードから HAL 出力ウィジェットの値を設定する必要がある場合は、基になる Gtk セッター（たとえば、set_active（）、set_value（））を呼び出して設定します。halcomp[によって関連付けられたピンの値を設定しようとしないでください。 pinname] =ウィジェットは変更を認識しないため、直接値を設定します。

プログラムで HAL ウィジェットの入力ピンを設定したくなるかもしれません。 これは、そもそも入力ピンの目的を損なうことに注意してください。他の HAL コンポーネントによって生成された信号にリンクし、それに反応

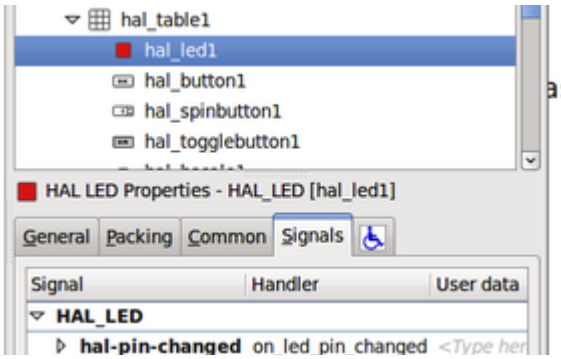
する必要があります。 現在、HAL Python の入力ピンへの書き込みには書き込み保護がありませんが、これは意味がありません。 ただし、テストには、関連するハーフイルで setppinname 値を使用できます。

この HAL ピンがウィジェットに関連付けられていない、つまり hal_glib.GPin（halcomp.newpin（<name>、<type>、<direction>）メソッド（例については、GladeVCP プログラミングを参照してください）。

7.3.6.3 hal-pin-changed 信号

イベント駆動型プログラミングとは、ボタンが押されたときのように、コールバックを介して「何かが起こった」ときに UI がコードに通知することを意味します。 LED、Bar、VBar、Meter などの出力 HAL ウィジェット（HAL ピンの値を表示するウィジェット）は、hal-pin が変更された信号をサポートします。これにより、HAL ピンが値を変更したときに Python コードにコールバックが発生する可能性があります。 。 これは、コード内の HAL ピンの変更を永続的にポーリングする必要がなくなることを意味します。ウィジェットはバックグラウンドでそれを行い、通知します。

次に、GladeUI エディターで HAL_LED の hal-pin-changed 信号を設定する方法の例を示します。



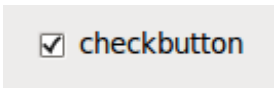
configs / apps / gladevcp / complex の例は、これが Python でどのように処理されるかを示しています。

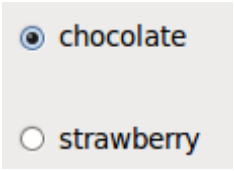
7.3.6.4 ボタン

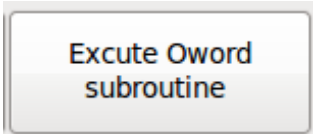
このウィジェットのグループは、さまざまな Gtk ボタンから派生し、HAL_Button、HAL_ToggleButton、HAL_RadioButton、および CheckButton ウィジェットで構成されています。それらはすべて、ウィジェットと同じ名前の単一の出力 BIT ピンを持っています。 ボタンには、基本の Gtk クラスと比較して追加のプロパティはありません。

- HAL_Button：瞬間的なアクション、状態を保持しません。 重要な信号：押された
- HAL_ToggleButton、HAL_CheckButton：オン/オフ状態を保持します。 重要な信号：切り替え
- HAL_RadioButton：1 対多のグループ。 重要な信号：切り替え（ボタンごと）。
- 重要な一般的なメソッド：set_active（）、get_active（）
- 重要なプロパティ：ラベル、画像

チェックボタン：



ラジオボタン： 

トグルボタン： 

ヒント

Glade でのラジオボタングループの定義：

-デフォルトのアクティブボタンを決定します

-他のボタンの General！Group で、[このプロジェクトのラジオボタンの選択]ダイアログでデフォルトのアクティブなボタンの名前を選択します。

ラジオボタンの操作用の GladeVCP アプリケーションおよび UI ファイルについては、configs / apps / gladevcp / by-widget /を参照してください。

7.3.6.5 スケール

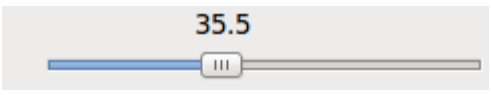
HAL_HScale と HAL_VScale は、GtkHScale と GtkVScale から派生しています。

それぞれ。

```
<widgetname>
out FLOAT pin
<widgetname>-s
out S32 pin
```

Glade でスケールを便利にするには、調整を追加します

（一般！調整！新規または既存の調整）そして調整オブジェクトを編集します。 デフォルト/最小/最大/増分値を定義します。 また、警告を回避するために、調整ページサイズとページ増分をゼロに設定します。

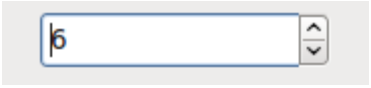
HAL_HScale の例： 

7.3.6.6 SpinButton

HAL SpinButton は GtkSpinButton から派生し、2 つのピンを保持します。

```
<widgetname>-f
out FLOAT pin
<widgetname>-s
out S32 pin
```

スピンボタンを使用するには、スケールなどの調整値が必要です。上記を参照してください。

SpinButton の例： 

7.3.6.7 Hal_Dial

hal_dial ウィジェットは、ジョグホイールまたは調整ダイヤルをシミュレートします。

マウスで操作できます。 マウスカーソルが Hal_Dial ウィジェット上にあるときにマウスホイールを使用するか、マウスの左ボタンを押したままカーソルを円方向に移動して、カウントを増減できます。

左ボタンまたは右ボタンをダブルクリックすると、倍率を増減できます。

- 反時計回り=カウントを減らす
- 時計回り=カウントを増やす
- ホイールアップ=カウントを増やす
- ホイールダウン=カウントを減らす
- 左ダブルクリック= x10 スケール
- 右ダブルクリック= / 10 スケール

```
Hal_Dial exports it’s count value as hal pins:
<widgetname>::
out S32 pin
<widgetname>-scaled::
out FLOAT pin
<widgetname>-delta-scaled::
out FLOAT pin
```

次のプロパティがあります。

Cpr

回転あたりのカウントを設定します。許可される値は 25～360 の範囲です。デフォルト= 100

show_counts

ウィジェットの中央にあるカウント表示を非表示にする場合は、これを False に設定します。 デフォルト= True

label

ラベルの内容を設定すると、カウント値の上に魔女が表示される場合があります。 指定されたラベルが 15 文字より長い場合、15 文字にカットされます。 デフォルト=空白

center_color

これにより、ホイールの色を変更できます。 GDK カラー文字列を使用します。 デフォルト=# bdefbdefbdef（灰色）

count_type_shown

利用可能な 3 つのカウントがあります 0）生の CPR カウント 1）スケーリングされたカウント 2）デルタスケーリングされたカウント。 デフォルト= 1

- カウントは、選択した CPR に基づいており、正と負のカウントになります。 S32 ピンとして利用できます。
- Scaled-count は、CPR カウントにスケールを掛けたものです。正と負のどちらでもかまいません。 スケールを変更すると、出力はすぐに変更を反映します。 FLOAT ピンとして使用できます。
- Delta-scaled-count は、cpr count CHANGE、xscale です。
- スケールを変更すると、その変更後のカウントのみが計算され、現在の値に追加されます。 FLOAT ピンとして使用できます。

scale_adjustable

ウィジェットをダブルクリックしてスケールの変更を禁止する場合は、これを False に設定します。これが false の場合、スケール係数はウィジェットに表示されません。

デフォルト= True

Scale

カウントをスケーリングするには、これを設定します。

デフォルト= 1.0

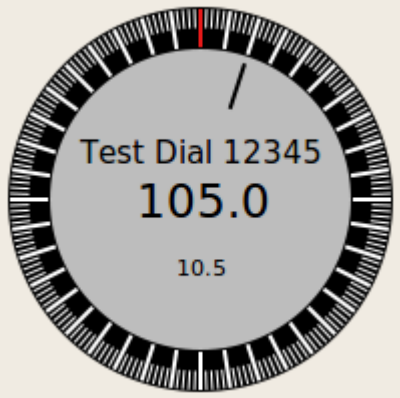
Direct program control

Python を使用してウィジェットを直接制御する方法があります。

```
Using goobject to set the above listed properties:
[widget name].set_property("cpr",int(value))
[widget name].set_property("show_counts, True)
[widget name].set_property("center_color",gtk.gdk.Color('#bdefbdefbdef'))
[widget name].set_property('label', 'Test Dial 12345')
[widget name].set_property('scale_adjustable', True)
[widget name].set_property('scale', 10.5)
[widget name].set_property('count_type_shown', 0)
There are python methods:
```

[widget name].get_value()
Will return the counts value as a s32 integer
[widget name].get_scaled_value()
Will return the counts value as a float
[widget name].get_delta_scaled_value()
Will return the counts value as a float
[widget name].set_label("string")
Sets the label content with "string"
There are two GObject signals emitted:
count_changed
emitted when the widget’s count changes eg. from being wheel scrolled.
scale_changed
emitted when the widget’s scale changes eg. from double clicking. +
connect to these like so:
[widget name].connect(’count_changed’, [count function name])
[widget name].connect(’scale_changed’, [scale function name]) +
The callback functions would use this pattern:
def [count function name](widget, count,scale,delta_scale):
This will return: the widget, the current count, scale and delta scale of -
that widget.

Hal_Dial の例：



7.3.6.8 ジョグホイール

jogwheel ウィジェットは、実際の jogwheel をシミュレートします。

マウスで操作できます。マウスカーソルが JogWheel ウィジェット上にあるときにマウスホイールを使用するか、マウスの左ボタンを押してカーソルを円方向に移動し、カウントを増減できます。

- 反時計回り=カウントを減らす

- 時計回り=カウントを増やす
- ホイールアップ=カウントを増やす
- ホイールダウン=カウントを減らす

マウスをドラッグアンドドロップで移動すると、ウィジェットがそれ自体を更新するよりも高速になる可能性があるため、カウントを失う可能性があります。マウスホイールを使用することをお勧めします。ドラッグアンドドロップの方法が非常に荒い場合にのみ使用してください。

JogWheel は、カウント値を hal ピンとしてエクスポートします。

```
<widgetname>-s
out S32 pin
```

次のプロパティがあります。

Size

ウィジェットのピクセル単位のサイズを設定します。許可される値は 100〜500 の範囲です。デフォルト= 200

cpr

回転あたりのカウントを設定します。許可される値は 25〜100 の範囲です。デフォルト= 40

show_counts

ウィジェットの中央にあるカウント表示を非表示にする場合は、これを False に設定します。

label

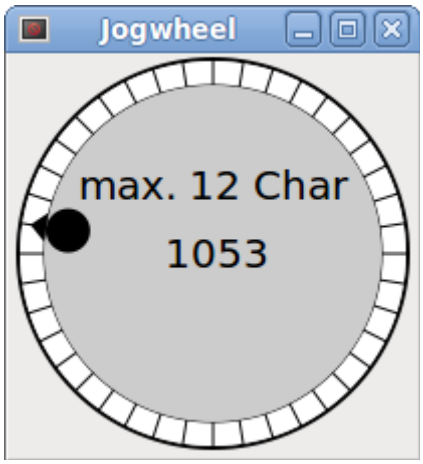
ラベルの内容を設定すると、カウント値の上に魔女が表示される場合があります。 目的は、そのジョグホイールの使用法についてユーザーにアイデアを与えることです。 指定されたラベルが 12 文字より長い場合、12 文字にカットされます。

Direct program control

Python を使用してウィジェットを直接制御する方法はいくつかあります。

```
Using gobject to set the above listed properties:
[widget name].set_property("size",int(value))
[widget name].set_property("cpr",int(value))
[widget name].set_property("show_counts, True)
There are two python methods:
[widget name].get_value()
Will return the counts value as integer
[widget name].set_label("string")
Sets the label content with "string"
```

JogWheel の例：



7.3.6.9 速度制御

SpeedControl は、タッチスクリーンで調整を制御するために特別に作成されたウィジェットです。タッチスクリーン上でスライドするのが難しい通常のスケールウィジェットの魔女に代わるものです。

値を増減するには、2 つのボタンで値を制御します。ボタンを押す限り、増分が変化します。各増分の値と 2 つの変更間の時間は、ウィジェットのプロパティを使用して設定できます。

SpeedControl はいくつかの hal ピンを提供します：

<widgetname>-value
out float pin The shown value of the widget
<widgetname>-scaled-value
out float pin The shown value divided by the scale value, this is very useful, if the velocity is shown in units / min, but
linuxcnc expects it to be in units / second
<widgetname>-scale
in float pin The scale to apply Default is 60
<widgetname>-increase
in bit pin As long as the pin is true, the value will increase Very handy with connected momentary switch
<widgetname>-decrease
in bit pin As long as the pin is true, the value will decrease Very handy with connected momentary switch

次のプロパティがあります。

height

整数ピクセル単位でのウィジェットの高さ許容値は 24～96 です。デフォルトは 36 です。

Value

float 許容値を設定するための開始値は、0.001 から 99999.0 の範囲です。デフォルトは 10.0 です。

Min

float 最小許容値の許容値は 0.0～99999.0 です。デフォルトは 0.0 です。この値を変更すると、増分がデフォルトにリセットされるため、後で新しい増分を設定する必要がある場合があります。

Max

float 許可される最大許容値は 0.001～99999.0 です。デフォルトは 100.0 です。この値を変更すると、増分がデフォルトにリセットされるため、後で新しい増分を設定する必要がある場合があります。

Increment

float は、マウスクリックごとに適用される増分を設定します。許容値は 0.001～99999.0 で、デフォルトは-1 で、最小から最大まで 100 の増分になります。

inc_speed

integer ボタンを押したままの増分速度のタイマー遅延を設定します。許可される値は 20～300 です。デフォルトは 100 です。

unit

string 任意の文字列が許可される値の後にバーに表示される単位を設定します。デフォルトは「」です。

Color

色任意の 16 進色が許可されるバーの色を設定します。デフォルトは「＃FF8116」です。

template

文字列 Python フォーマットが使用される値を表示するテキストテンプレート許可されるフォーマットのデフォルトは「%.1f」です

do_hide_button

ブール値インクリメントを表示するか非表示にするかデクリメントボタン True または False デフォルト= False

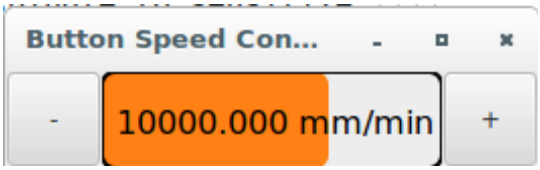
Direct program control

Python を使用してウィジェットを直接制御する方法はいくつかあります。

```
Using gobject to set the above listed properties:
[widget name].set_property("do_hide_button",bool(value))
[widget name].set_property("color","#FF00FF")
[widget name].set_property("unit", "mm/min")
etc.
There are also python methods to modify the widget:
[widget name].set_adjustment(gtk-adjustment)
```

```
You can assign a existing adjustment to the control, that way it is easy  -
to replace
existing sliders without many code changes. Be aware, that after changing  -
the adjustment
you may need to set a new increment, as it will be reseted to its default  -
(100 steps from MIN to MAX)
[widget name].get_value()
Will return the counts value as float
[widget name].set_value(float(value))
Sets the widget to the commanded value
[widget name].set_digits(int(value))
Sets the digits of the value to be used
[widget name].hide_button(bool(value))
Hide or show the button
```

速度制御の例：



7.3.6.10 Label

HAL_Label は、GtkLabel に基づく単純なウィジェットであり、ユーザー定義形式で HAL ピン値を表します。

label_pin_type

ピンの HAL タイプ（0：S32、1：float、2：U32）。「General！HAL ピンタイプ」のツールチップも参照してください（これは、タイプごとに 1 つずつ、3 つのラベルウィジェットがある PyVCP とは異なります）。

text_template

表示されるテキストを決定します-ピン値をテキストに変換する Python 形式の文字列。 デフォルトは%s（値は str（）関数によって変換されます）ですが、Python の format（）メソッドへの引数として任意の正当性を含めることができます。

例：距離：%。03f は、FLOAT ピンのテキストとピンの値をゼロで埋められた 3 桁の小数で表示します。

7.3.6.11 コンテナ

- HAL_HideTable
- HAL_Table State_Sensitive_Table
- HAL_HBox

これらのコンテナは、子供を感作（グレーアウト）または非表示にするために使用することを目的としています。

鈍感な子供は入力に応答しません。

HAL_HideTable には、子ウィジェットを非表示にするかどうかを制御する 1 つの HALBIT 入力ピンがあります。

> <Panel_basename>.<widgetname>
> If the pin is low then child widgets are visible which is the default state.

HAL_Table と HAL_Hbox には、子ウィジェットが機密であるかどうかを制御する 1 つの HALBIT 入力ピンがあります。

これらのウィジェットのピン名は、次の規則を使用しています。

> <Panel_basename>.<widgetname>
> If the pin is low then child widgets are inactive which is the default state.

State_Sensitive_table は、状態を linuxcnc のインタープリターに応答します。

オプションで、すべてのホーム、オン、アイドルに対応するために選択可能

それらを組み合わせることができます。 Estop では常に鈍感になります。

* HAL_Hbox は非推奨です-HAL_Table を使用してください。

現在のパネルがそれを使用している場合、それは失敗しません。 GLADE エディターではもう見つかりません。

gladeVCP の将来のバージョンでは、このウィジェットが完全に削除される可能性があり、パネルを更新する必要があります。

> ヒント
> GladeVCP アプリケーションの一部がグレー表示されている（感度が低い）場合は、
> HAL_Table ピンが設定されていないか接続されていないかを確認してください。

7.3.6.12 LED

hal_led は、実際のインジケーター LED をシミュレートします。

オンまたはオフの状態を制御する単一の入力 BIT ピンがあります。

LED には、ルックアンドフィールを制御するいくつかのプロパティがあります。

on_color

LED のオンカラーを定義する文字列。 任意の有効な gtk.gdk.Color 名である可能性があります。 Ubuntu8.04 では動作しません。

off_color

LED のオフカラーを定義する文字列。 有効な gtk.gdk.Color 名または特別な値 dark のいずれかです。 暗いとは、OFF カラーが ON カラーの 0.4 値に設定されることを意味します。 Ubuntu8.04 では動作しません。

pick_color_on、pick_color_off

オン状態とオフ状態の色は、＃RRRRGGGGBBBB 文字列として表すことができます。 これらは、on_color および off_color よりも優先されるオプションのプロパティです。

led_size

LED 半径（正方形の場合-LED の側面の半分）

led_shape

LED の形。 有効な値は、円形の場合は 0、楕円形の場合は 1、正方形の場合は 2 です。

led_blink_rate

設定されていて LED がオンの場合は、点滅します。 点滅時間は、ミリ秒単位で指定された「led_blink_rate」と同じです。

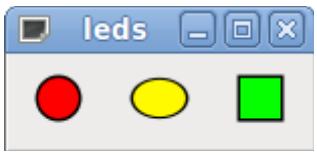
create hal pin

LED を制御するための HAL ピンの作成を選択/選択解除します。HAL ピンが作成されていない場合、LED は Python 関数で制御できます。 入力ウィジェットとして、LED は hal-pin-changed 信号もサポートします。 LED の HAL ピンが変更されたときにコードで通知を受け取りたい場合は、この信号をハンドラー（on_led_pin_changed など）に接続し、次のようにハンドラーを指定します。

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

これは、信号の任意のエッジで呼び出され、プログラムの起動時に現在の値を報告するためにも呼び出されます。

LED の例：



7.3.6.13 プログレスバー

Note

このウィジェットはなくなる可能性があります。 代わりに、HAL_HBar ウィジェットと HAL_VBar ウィジェットを使用してください。

HAL_ProgressBar は gtk.ProgressBar から派生し、2 つの floatHAL 入力ピンがあります。

```
<widgetname>
the current value to be displayed
<widgetname>-scale
the maximum absolute value of input
```

次のプロパティがあります。

Scale

値のスケール。 入力の最大絶対値を設定します。 <widgetname> .scale ピンを設定するのと同じです。 フロート、範囲は-224〜 +224 です。

green_limit

グリーンゾーン制限下限

yellow_limit

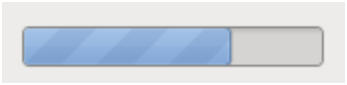
イエローゾーン制限下限

red_limit

レッドゾーン制限下限

text_template

<widgetname>ピンの現在の値を表示するテキストテンプレート。 dict {"value"：value}には Python フォーマットを使用できます

HAL_ProgressBar の例： 

7.3.6.14 コンボボックス

HAL_ComboBox は gtk.ComboBox から派生しています。 ドロップダウンリストから値を選択できます。

2 つの HAL ピンをエクスポートします。

```
<widgetname>-f
the current value, type FLOAT
<widgetname>-s
the current value, type S32
```

Glade で設定できる次のプロパティがあります。

column

列インデックス、タイプ S32、デフォルトは-1、範囲は-1..100 です。

デフォルトモードでは、このウィジェットはピンを選択されたリストエントリのインデックスに設定します。したがって、ウィジェットに 3 つのラベルがある場合、ウィジェットは値 0、1、および 2 のみを想定する可能性があります。

列モード（列> -1）では、報告される値は、Glade で定義されている ListStore 配列から選択されます。 したがって、通常、ウィジェット定義には ListStore に 2 つの列があり、1 つはドロップダウンにテキストが表示され、1 つはその選択に使用する int または float 値です。

configs / apps / by-widget / combobox。{py、ui}に例があります。これは、列モードを使用してリストストアから浮動小数点値を選択します。

ComboBox ListStores と CellRenderer の編集方法について私のように混乱している場合は、http：//www.youtube.com watch？v = Z5_FrW2cL8 を参照してください。

7.3.6.15 バー

浮動小数点値を表す水平および垂直バー用の HAL バーおよび VBar ウィジェット。 1 つの入力 FLOAThal ピンがあります。 両方のバーには次のプロパティがあります。

Invert

最小方向と最大方向を入れ替えます。 逆 HBar は右から左に成長し、逆 VBar は上から下に成長します。

min, max

希望する範囲の最小値と最大値。 現在の値がこの範囲外であれば、エラー状態ではありません。

show limits

バーの制限テキストを選択/選択解除するために使用されます。

zero

範囲のゼロ点。 最小/最大範囲内にある場合、バーはウィジェットの左側（または右側）からではなく、その値から大きくなります。 正または負の両方の値を表すのに役立ちます。

force_width, force_height

ウィジェットの強制的な幅または高さ。 設定されていない場合、サイズはパッキングまたは固定ウィジェットサイズから推定され、バーが領域全体に表示されます。

text_template

ラベルのように、最小/最大/現在の値のテキスト形式を設定します。 値の表示をオフにするために使用できます。

value

バーの表示を入力された値に設定します：GLADE エディターでのテストにのみ使用されます。 値は AHAL ピンから設定されます。

target value

ターゲット行を入力された値に設定します。GLADE エディターでのテストにのみ使用されます。 値は Python 関数で設定できます

target_width

目標値を示す線の幅。

bg_color

バーの背景（非アクティブ）の色。

target_color

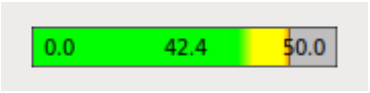
ターゲットラインの色。

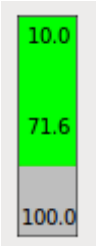
z0_color、z1_color、z2_color

さまざまなバリューゾーンの色。 デフォルトは緑、黄、赤です。 ゾーンの説明については、z * _border プロパティを参照してください。

z0_border、z1_border

カラーゾーンの上限を定義します。 デフォルトでは、1 つのゾーンのみが有効になっています。 複数のゾーンで z0_border と z1_border を目的の値に設定して、ゾーン 0 が 0 から最初の境界まで、ゾーン 1 が最初から 2 番目の境界まで、ゾーン 2 が最後の境界から 1 まで塗りつぶされるようにする場合、境界は分数として設定されます。 0 から 1 までの値。

水平バー： 

垂直バー： 

7.3.6.16 Meter

HAL メーターは、PyVCP メーターに似たウィジェットです。これは、float 値を表し、1 つの入力 FLOAThal ピンを備えています。 HAL メーターには次のプロパティがあります。

min, max

希望する範囲の最小値と最大値。 現在の値がこの範囲外であれば、エラー状態ではありません。

force_size

ウィジェットの強制直径。 設定されていない場合、サイズはパッキングまたは固定ウィジェットサイズから推定され、メーターはアスペクト比に関して使用可能なすべてのスペースを埋めます。

text_template

Label のように、現在の値のテキスト形式を設定します。 値の表示をオフにするために使用できます。

label

メーターの中心の上の大きなラベル。

Sublabel

メーターの中心の下にある小さなラベル。

bg_color

メーターの背景色。

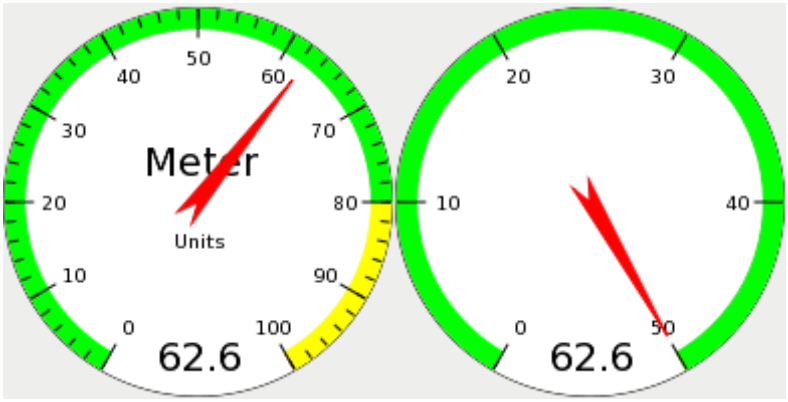
z0_color、z1_color、z2_color

さまざまなバリューゾーンの色。 デフォルトは緑、黄、赤です。 ゾーンの説明については、z * _border プロパティを参照してください。

z0_border、z1_border

カラーゾーンの上限を定義します。 デフォルトでは、1 つのゾーンのみが有効になっています。 複数のゾーンが必要な場合は、z0_border と z1_border を目的の値に設定して、ゾーン 0 が最小から最初の境界まで、ゾーン 1 が最初から 2 番目の境界まで、ゾーン 2 が最後の境界から最大まで塗りつぶされるようにします。 境界線は、最小から最大の範囲の値として設定されます。

HAL メーターの例：



7.3.6.17 HAL_Graph

このウィジェットは、時間の経過に伴う値をプロットするためのものです。

7.3.6.18 .ngc ファイルの Gremlin ツールパスプレビュー

Gremlin は、Axis プレビューウィンドウに似たプロットプレビューウィジェットです。 Axis や Touchy などの実行中の LinuxCNC 環境を想定しています。 接続するには、INI_FILE_NAME 環境変数を調べます。 Gremlin は現在の.ngc ファイルを表示します-変更を監視し、Axis / Touchy のファイル名が変更された場合は ngc ファイルをリロードします。 LinuxCNC が実行されていないときに GladeVCP アプリケーションで実行すると、Gremlin ウィジェットが現在のファイル名などの LinuxCNC ステータスを検出できないため、トレースバックが発生する可能性があります。

グレムリンは HAL ピンをエクスポートしません。 次のプロパティがあります。

show tool speed

工具速度が表示されます。 デフォルトは true

show commanded

これにより、コマンド値または実際の値を使用する DRO が選択されます。 デフォルトは true

use metric units

これにより、メートル法またはインペリアル単位を使用する DRO が選択されます。 デフォルトは true

show rapids

これにより、プロッタは急速な動きを示すようになります。 デフォルトは true

show DTG

これにより、DRO が選択され、移動距離の値が表示されます。 デフォルトは true

show relative

これにより、DRO が選択され、ユーザーシステムまたはマシンの座標に関連する値が表示されます。 デフォルトは true

show live plot

これは、プロッタに描画するかどうかを指示します。 デフォルトは true

show limits

これは、プロッタにマシンの制限を表示するように指示します。 デフォルトは true

show lathe radius

これにより、旋盤モードの場合に X 軸を半径または直径で表示する DRO が選択されます（LATHE = 1 の INI ファイルで選択可能）。 デフォルトは false

show extents

これは、範囲を表示するようにプロッタに指示します。 デフォルトは true

show tool

これは、プロッタにツールを描画するように指示します。 デフォルトは true

show program

TODO

use joints mode

トリビアルキンス以外のマシン（ロボットなど）で使用されます。 デフォルトは false

grid size

グリッドのサイズを設定します。 これは、X、Y、および Z ビューでのみ表示されます。 デフォルトは 0

use default mouse controls

これにより、デフォルトのマウスコントロールが無効になります。 これは、デフォルトのコントロールがうまく機能しないため、タッチスクリーンを使用する場合に最も役立ちます。 Python とハンドラーファイルの手法を使用して、プログラムでコントロールを追加できます。 デフォルトは True です

View

x、y、y2、z、z2、p（パースペクティブ）のいずれかです。 デフォルトは z ビューです。

enable_dro

ブール値; プロットに DRO を描画するかどうか。 デフォルトは True です

mouse_btn_mode

整数; マウスボタンの処理、ボタンのさまざまな機能につながります 0 =デフォルト：左回転、中央移動、右ズーム

1 =左ズーム、中回転、右回転 2 =左移動、中回転、右ズーム 3 =左ズーム、中回転、右移動

4 =左移動、中ズーム、右回転 5 =左回転、中ズーム、右移動 6 =左移動、中ズーム、右ズーム

mode 6 is reccomended for plasmas and lathes, as rotation is not needed for such machines

Direct program control

Python を使用してウィジェットを直接制御する方法はいくつかあります。

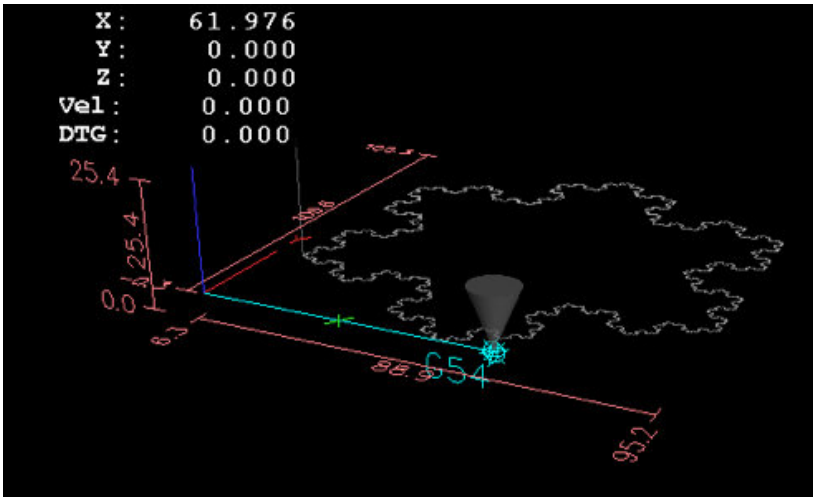
Using goobject to set the above listed properties: [widget name].set_property('view','P') [widget name].set_property('metric_units',False) [widget name].set_property('use_default_controls',False) [widget name].set_property('enable_dro' False)) [widget name].set_property('show_program', False) [widget name].set_property('show_limits', False) [widget name].set_property('show_extents_option', False) [widget name].set_property('show_live_plot', False) [widget name].set_property('show_tool', False) [widget name].set_property('show_lathe_radius',True) [widget name].set_property('show_dtg',True) [widget name].set_property('show_velocity',False) [widget name].set_property('mouse_btn_mode', 4) There are python methods: [widget name].show_offsets = True [widget name].grid_size = .75 [widget name].select_fire(event.x,event.y) [widget name].select_prime(event.x,event.y) [widget name].start_continuous_zoom(event.y) [widget name].set_mouse_start(0,0) [widget name].gremlin.zoom_in()

```
[widget name].gremlin.zoom_out()
[widget name].get_zoom_distance()
[widget name].set_zoom_distance(dist)
[widget name].clear_live_plotter()
[widget name].rotate_view(x,y)
[widget name].pan(x,y)
```

ヒント

- すべてのプロットオプションを false に設定し、show_offsets を true に設定すると、グラフィックプロットではなくオフセットページが表示されます。
- ビューを変更する前にズーム距離を取得してからズーム距離をリセットすると、はるかにユーザーフレンドリーになります。
- プレビューで要素を選択すると、選択した要素が回転の中心点として使用されます

例：



7.3.6.19 HAL_Offset

HAL_Offset ウィジェットは、単一軸のオフセットを表示するために使用されます。 次のプロパティがあります。

ジョイント番号

表示する軸（技術的にはどのジョイント）を選択するために使用されます。 トリビアルキンスマシン（ミル、旋盤、ルーター）の軸 vrs ジョイント番号は次のとおりです。

```
0:X 1:Y 2:Z 3:A 4:B 5:C 6:U 7:V 8:W
Text template for metric units::
You can use python formatting to display the position with different  -
precision.
Text template for imperial units::
You can use python formatting to display the position with different  -
```

```
precision.
Reference Type::
0:G5x 1:tool 2:G92 3:Rotation around Z
```

7.3.6.20 DRO ウィジェット

DRO ウィジェットは、現在の軸の位置を表示するために使用されます。 次のプロパティがあります。

Actual Position

実際の（フィードバック）位置またはコマンド位置を選択します。

Text template for metric units

Python フォーマットを使用して、さまざまな精度で位置を表示できます。

Text template for imperial units

Python フォーマットを使用して、さまざまな精度で位置を表示できます。

Reference Type

絶対（マシンの原点）、相対（現在のユーザー座標の原点に対して-G5x）、または移動距離（現在のユーザー座標の原点に対して）

Joint Number

表示する軸（技術的にはどのジョイント）を選択するために使用されます。 トリビアルキンスマシン（ミル、旋盤、ルーター）の軸 vrs ジョイント番号は次のとおりです。

```
0:X 1:Y 2:Z 3:A 4:B 5:C 6:U 7:V 8:W
```

Display units

表示単位をメートル法とインペリアルの間で切り替えるために使用されます。

ヒント

- 表示を右揃えにする場合は、X 位置合わせを 1.0 に設定します
- 異なる色、サイズ、またはテキストが必要な場合は、空き地エディタで属性を変更します（たとえば、スケールはテキストのサイズを変更するための良い方法です）
- ウィジェットの背景は実際には透けて見えます。そのため、画像の上に配置すると、DRO 番号が背景なしでその上に表示されます。 これを行うための特別なテクニックがあります。 以下のアニメーション機能図を参照してください。
- DRO ウィジェットは、変更された gtk ラベルウィジェットです。 そのように、または gtk ラベルに対して実行できることは、DRO ウィジェットに対して実行できます。

Python を使用してウィジェットを直接制御する方法はいくつかあります。

```
Using goobject to set the above listed properties:
[widget name].set_property("display_units_mm",True)
[widget name].set_property("actual",True)
[widget name].set_property("mm_text_template","%f")
[widget name].set_property("imperial_text_template","%f")
[widget name].set_property("Joint_number",3)
[widget name].set_property("reference_type",3)
There are two python methods:
[widget name].set_dro_inch()
[widget name].set_dro_metric()
```

7.3.6.21 Combi_DRO ウィジェット

Combi_DRO ウィジェットは、現在、相対軸位置、および 1 つの DRO 内を移動する距離を表示するために使用されます。

DRO をクリックすると、DRO の順序が切り替わります。

相対モードでは、実際の座標系が表示されます。

次のプロパティがあります。

joint_number

表示する軸（技術的にはどのジョイント）を選択するために使用されます。

トリビアルキンスマシン（ミル、旋盤、ルーター）軸 vrs。 ジョイント番号は次のとおりです。

0：X 1：Y 2：Z など

actual

実際の（フィードバック）またはコマンドされた位置を選択します。

metric_units

表示単位をメートル法とインペリアルの間で切り替えるために使用されます。

auto_units

単位は、アクティブな gcode が G20 または G21 であることに応じて、メートル法とインペリアル法の間で切り替わります。

デフォルトは TRUE です

diameter

位置を直径として表示するか半径として表示するかにかかわらず、直径モードでは、DRO はジョイント値に 2 を掛けた値を表示します。

mm_text_template

Python フォーマットを使用して、さまざまな精度で位置を表示できます。

デフォルトは「%10.3f」です

imperial_text_template

Python フォーマットを使用して、さまざまな精度で位置を表示できます。

デフォルトは「%9.4f」です

homed_color

ジョイントがホームになっている場合の DRO 番号の前景色

デフォルトは緑です

unhomed_color

ジョイントがホームになっていない場合の DRO 番号の前景色

デフォルトは赤です

abs_color

メイン DRO が絶対座標を示している場合、DRO の背景色

デフォルトは青です

rel_color

メイン DRO が相対座標を示している場合、DRO の背景色

デフォルトは黒です

dtg_color

メイン DRO が移動距離を示している場合、DRO の背景色

デフォルトは黄色です

font_size

大きい数字、小さい数字のフォントサイズは 2.5 倍小さくなり、値は 8〜96 の範囲の整数である必要があります。

デフォルトは 25 です

toggle_readout

マウスを左クリックすると、DRO の読み取り値がさまざまなモード["Rel"、 "Abs"、 "DTG"]に切り替わります。

チェックボックスをオフにすると、その動作を無効にできます。 トグルは[ウィジェット名]で引き続き実行できます。toggle_readout（）値は bool である必要があります

デフォルトは TRUE です

cycle_time

DRO が 2 つのポーリング間で待機する時間、値は 100～1000 の範囲の整数である必要があり、デフォルトは 150 です。この設定は、同時に 5 つ以上の DRO を使用する場合、つまり 6 軸でのみ変更する必要があります。 回避するために、DRO がメインアプリケーションの速度を大幅に低下させるように構成します。

Direct program control

gobject を使用して上記のプロパティを設定します。

```
[widget name].set_property(property, value)
```

ウィジェットを制御するための Python メソッドはいくつかあります。

```
[widget name].set_to_inch(state)
sets the DRO to show imperial units
state = boolean (True or False)
[widget name].set_auto_units(state)
if True the DRO will change units according to active gcode (G20 / G21)
state = boolean (True or False)
Default is True
[widget name].set_to_diameter(state)
if True the DRO will show the diameter not the radius, specially needed for -
lathes
the DRO will display the axis value multiplied by 2
state = boolean (True or False)
Default is False
[widget name].toggle_readout()
toggles the order of the DRO in the widget
[widget name].change_axisletter(letter)
changes the automatically given axis letter
very useful to change an lathe DRO from X to R or D
letter = string
[widget name].get_order()
returns the order of the DRO in the widget mainly used to maintain them -
consistent
```

```
the order will also be transmitted with the clicked signal
returns a list containing the order
[widget name].set_order(order)
sets the order of the DRO, mainly used to maintain them consistent
order = list object, must be one of
["Rel", "Abs", "DTG"]
["DTG", "Rel", "Abs"]
["Abs", "DTG", "Rel"]
Default = ["Rel", "Abs", "DTG"]
[widget name].get_position()
returns the position of the DRO as a list of floats
the order is independent of the order shown on the DRO
and will be given as [Absolute , relative , DTG]
Absolute = the machine coordinates, depends on the actual property
will give actual or commanded position
Relative = will be the coordinates of the actual coordinate system
DTG = the distance to go, will mostly be 0, as this function should not be -
used
while the machine is moving, because of time delays
```

ウィジェットは次のシグナルを発します：

```
clicked
This signal is emitted, when the user has clicked on the Combi_DRO widget,
it will send the following data:
widget = widget object = The widget object that sends the signal
joint_number = integer = The joint number of the DRO, where '0:X 1:Y 2:Z -
etc'
order = list object = the order of the DRO in that widget
the order may be used to set other Combi_DRO widgets to -
the same order with [widget name].set_order(order)
units_changed
This signal is emitted, if the DRO units are changed, it will send the -
following data:
widget = widget object = The widget object that sends the signal
metric_units = boolean = True if the DRO does display metric units, False in -
case of imperial display
system_changed
This signal is emitted, if the DRO units are changed, it will send the -
following data:
widget = widget object = The widget object that sends the signal
system = string = The actual coordinate system. Will be one of
```

G54 G55 G56 G57 G58 G59 G59.1 G59.2 G59.3 or Rel if non has been selected at all, what will only - happen in Glade with no linuxcnc running

あなたがコマンドを通して得ることができるいくつかの情報があります、魔女はあなたにとって興味があるかもしれません：

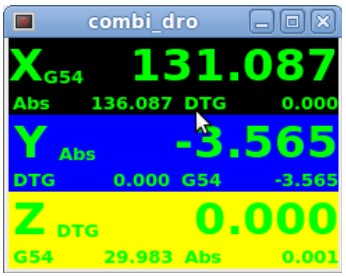
[widget name].system The actual system, as mentioned in the system_changed signal [widget name].homed True if the joint is homed [widget name].machine_units 0 if Imperial, 1 if Metric

例、ウィンドウ内の 3 つの Combi_DRO

X =相対モード

Y =絶対モード

Z = DTG モード



7.3.6.22 IconView（ファイル選択）

これは、ファイルを選択したり、ディレクトリを変更したりするためのタッチスクリーン対応のウィジェットです。

ウィジェットには次のプロパティがあります。

icon_size

表示されるアイコンのサイズを設定します。

許可される値は、12～96 の範囲の整数です。

デフォルトは 48 です

start_dir

ウィジェットが最初に表示されたときに開始するディレクトリを設定します。

有効なディレクトリパスを含む文字列である必要があります。

デフォルトは「/」です

jump_to_dir

ディレクトリを「ジャンプ先」ディレクトリに設定します。魔女は下のボタンリストの対応するボタンで選択され、左から 5 番目のボタンが数えられます。

有効なディレクトリパスを含む文字列である必要があります。

デフォルトは「～」です

filetypes

表示するオブジェクトのファイルフィルターを設定します

表示する拡張子のコンマ区切りリストを含む文字列である必要があります

デフォルトは「ngc、py」です

sortorder

sortorder

表示されるアイコンの並べ替え順序を設定します。0 から 3 までの整数値である必要があります。

0 =昇順（ファイル名に従ってソート）

1 =降順（ファイル名に従ってソート）

2 = FOLDERFIRST（最初にフォルダーを表示し、次にファイルを表示します）

3 = FILEFIRST（最初にファイルを表示し、次にフォルダーを表示します）、

デフォルト= 2 = FOLDERFIRST

Direct program control

goobject を使用して上記のプロパティを設定します。

```
[widget name].set_property(property,Value)
```

ウィジェットを制御するための Python メソッドがあります。

```
[widget name].show_buttonbox(state)
if False the bottom button box will be hidden, this is helpful in custom  -
screens,
with special buttons layouts to not alter the layout of the GUI, good example
for that is gmoccapy
state = boolean (True or False)
Default is True
```

```
[widget name].show_filelabel(state)
if True the file label (between the IconView window and the bottom button box -
will be shown.
Hiding this label may save place, but showing it is very useful for debugging -
reasons,
state = boolean (True or False)
Default is True
[widget name].set_icon_size(iconsize)
sets the icon size
must be an integer in the range from 12 to 96
Default = 48
[widget name].set_directory(directory)
Allows to set an directory to be shown
directory = string (a valid file path)
[widget name].set_filetypes(filetypes)
sets the file filter to be used, only files with the given extensions will be -
shown
filetypes = string containing a comma separated list of extensions
Default = "ngc,py"
[widget name].get_selected()
Returns the path of the selected file, or None if an directory has been -
selected
[widget name].refresh_filelist()
Refreshes the filelist, needed if you add a file without changing the -
directory
```

ボタンボックスが非表示になっている場合は、次のようにクリックされた信号を介してこのボタンの機能にアクセスできます。

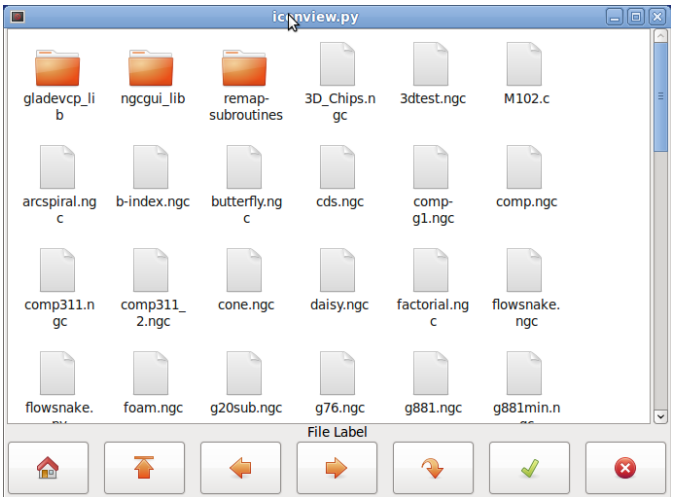
```
[widget name].btn_home.emit("clicked")
[widget name].btn_jump_to.emit("clicked")
[widget name].btn_sel_prev.emit("clicked")
[widget name].btn_sel_next.emit("clicked")
[widget name].btn_get_selected.emit("clicked")
[widget name].btn_dir_up.emit("clicked")
[widget name].btn_exit.emit("clicked")
```

ウィジェットは次のシグナルを発します：

```
selected
This signal is emitted, when the user selects an icon, it will return a string -
containing a
file path if a file has been selected, or None if an directory has been -
```

selected
sensitive
This signal is emitted, when the buttons change there state from sensitive to not sensitive or vice versa.
This signal is useful to maintain surrounding GUI synchronized with the button of the widget. See gmoccapy as example.
It will return the buttonname and the new state. Buttonname is one of "btn_home", "btn_dir_up", "btn_sel_prev", "btn_sel_next", "btn_jump_to" or "btn_select". State is a boolean and will be True or False.
exit
This signal is Emmit, when the exit button has been pressed to close the IconView
mostly needed if the application is started as stand alone.

例：



7.3.6.23 電卓ウィジェット

これは、数値入力に使用できる単純な電卓ウィジェットです。

表示をプリセットして、結果またはそのプリセット値を取得できます。

次のプロパティがあります。

Is editable

これにより、キーボードからエントリ表示を入力できます。

Set Font

これにより、ディスプレイのフォントを設定できます

Direct program control

Python を使用してウィジェットを直接制御する方法はいくつかあります。

```
Using goobject to set the above listed properties:
[widget name].set_property("is_editable",True)
[widget name].set_property("font","sans 25")
There are python methods:
[widget name].set_value(2.5)
This presets the display and is recorded.
[widget name].set_font("sans 25")
[widget name].set_editable(True)
[widget name].get_value()
Returns the calculated value - a float.
[widget name].set_editable(True)
[widget name].get_preset_value()
Returns the recorded value: a float.
```

7.3.6.24 **Tooleditor ウィジェット**

これは、ツールエディタファイルを表示および変更するためのツールエディタウィジェットです。

旋盤モードの場合、摩耗オフセットと工具オフセットが別々に表示されます。

摩耗オフセットは、10000 を超える工具番号で指定されます（ファナックスタイル）

linuxcnc では、摩耗オフセットを実際に使用するためにツール呼び出しを再マッピングする必要があることに注意してください

現在のファイルを 1 秒に 1 回チェックして、linuxcnc がファイルを更新したかどうかを確認します。

次のプロパティがあります。

Hidden Columns

これにより、指定された列が非表示になります。列は次のように（順番に）指定されます。

s、t、p、x、y、z、a、b、c、u、v、w、d、i、j、q、;

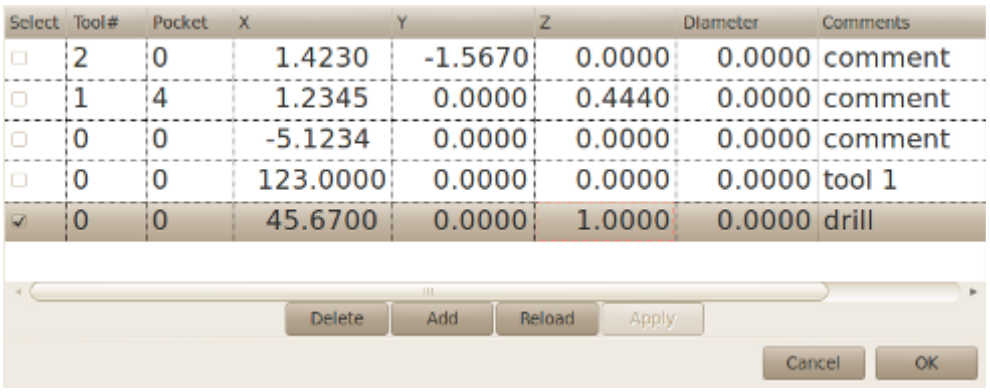
選択とコメントを含む任意の数の列を非表示にできます

Direct program control

Python を使用してウィジェットを直接制御する方法はいくつかあります。

```
using goobject to set the above listed properties:
[widget name].set_properties('hide_columns','uvwijq')
This would hide the uvwij and q columns and show all others.
There are python methods:
```

[widget name].set_visible("ijq",False)
Would hide ij and Q columns and leave the rest as they were.
[widget name].set_filename(path_to_file)
Sets and loads the tool file.
[widget name].reload(None)
Reloads the current toolfile
[widget name].set_font('sans 16,tab='1')
Sets the (Pango) font on the Tab, column title, and tool data.
The all_offsets, wear_offsets, tool_offsets can be set at the same -
time by
adding 1,2 and/or 3 to the tab string. Default is all the tabs
set.
[widget name].set_title_font('sans 16,tab='1')
Sets the (Pango) font on the column titles only.
The all_offsets, wear_offsets, tool_offsets can be set at the same -
time by
adding 1,2 and/or 3 to the tab string. Default is all the tabs
set.
[widget name].set_tab_font('sans 16,tab='1')
Sets the (Pango) font on the tabs only.
The all_offsets, wear_offsets, tool_offsets can be set at the same -
time by
adding 1,2 and/or 3 to the tab string. Default is all the tabs
set.
[widget name].set_col_visible("abcUVW", False, tab='1')
This would hide (False) the abcuvw columns on tab 1 (all_offsets)
[widget name].set_lathe_display(value)
hides or shows the wear and tool offset tabs used for lathes
[widget name].get_toolinfo(toolnum)
Returns the tool information array of the requested toolnumber
or current tool if no tool number is specified
returns None if tool not found in table or if there is no current tool
[widget name].hide_buttonbox(self, True)
'convenience' method to hide buttons
you must call this after show_all()
[widget name].get_selected_tool()
return the user selected (highlighted) tool number
[widget name].set_selected_tool(toolnumber)
Selects (highlights) the requested tool



7.3.6.25 オフセットページ

オフセットページウィジェットは、すべての軸のオフセットを表示/編集するために使用されます。

G92 および Rotation-Around-Z オフセットをゼロにするための便利なボタンがあります。

マシンがオンでアイドル状態のときにのみ編集モードを選択できます。

この時点で、テーブルのオフセットを直接編集できます。 編集の選択を解除します

ボタンをクリックして、OffsetPage に変更を反映させます。

次のプロパティがあります。

Hidden Columns

非表示にする列のスペースなしリスト：列は次のように（順番に）指定されます。

xyzabcuvwt

任意の列を非表示にできます。

Hidden Rows

非表示にする行のスペースなしリスト：行は（順番に）そのように指定されます

0123456789abc

任意の行を非表示にできます。

Pango Font

テキストのフォントタイプとサイズを設定します

HighLight color

編集するとき、これはハイライトカラーです

Active color

OffsetPage がアクティブなユーザー座標系を検出すると、テキストにこの色を使用します

Text template for metric units

Python フォーマットを使用して、さまざまな精度で位置を表示できます。

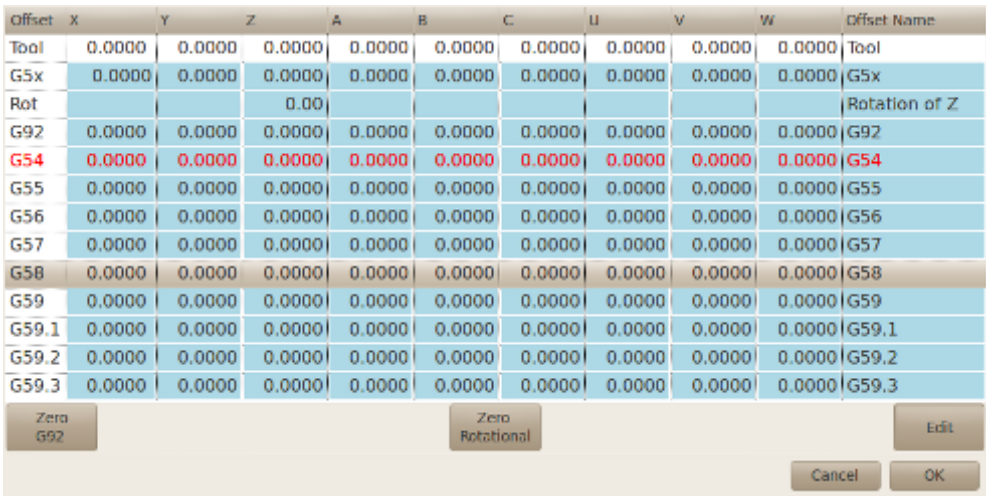
Text template for imperial units

Python フォーマットを使用して、さまざまな精度で位置を表示できます。

Direct program control

Python を使用してウィジェットを直接制御する方法はいくつかあります。

Using goobject to set the above listed properties:
[widget name].set_property("highlight_color",gtk.gdk.Color('blue'))
[widget name].set_property("foreground_color",gtk.gdk.Color('black'))
[widget name].set_property("hide_columns","xyzabcuvwt")
[widget name].set_property("hide_rows","123456789abc")
[widget name].set_property("font","sans 25")
There are python methods to control the widget:
[widget name].set_filename("../../../configs/sim/gscreen/gscreen_custom/sim. -
var")
[widget name].set_col_visible("Yabuvw",False)
[widget name].set_row_visible("456789abc",False)
[widget name].set_to_mm()
[widget name].set_to_inch()
[widget name].hide_button_box(True)
[widget name].set_font("sans 20")
[widget name].set_highlight_color("violet")
[widget name].set_foreground_color("yellow")
[widget name].mark_active("G55")
Allows you to directly set a row to highlight.
(eg in case you wish to use your own navigation controls.
See <<cha:gmoccapy,Gmoccapy Chapter>>
[widget name].selection_mask = ("Tool","Rot","G5x")
These rows are NOT selectable in edit mode.
[widget name].set_names([['G54','Default'],["G55","Vice1"],['Rot','Rotational -
']])
This allows you to set the text of the 'T' column of each/any row.
This is a list of a list of offset-name/user-name pairs.
The default text is the same as the offset name.
[widget name].get_names()
This returns a list of a list of row-keyword/user-name pairs.
The user name column is editable, so saving this list is user friendly.
see set_names above.



7.3.6.26 HAL_sourceview ウィジェット

これは、Gcode の表示と簡単な編集用です。 〜/ share / gtksourceview-2.0 / language-specs /で.ngc ハイライト仕様を検索します。現在実行中の行がハイライトされます。 外部の Python グルーコードを使用する場合：

*テキストの検索、変更の取り消し、やり直しが可能です。

※プログラム行選択に使用できます。

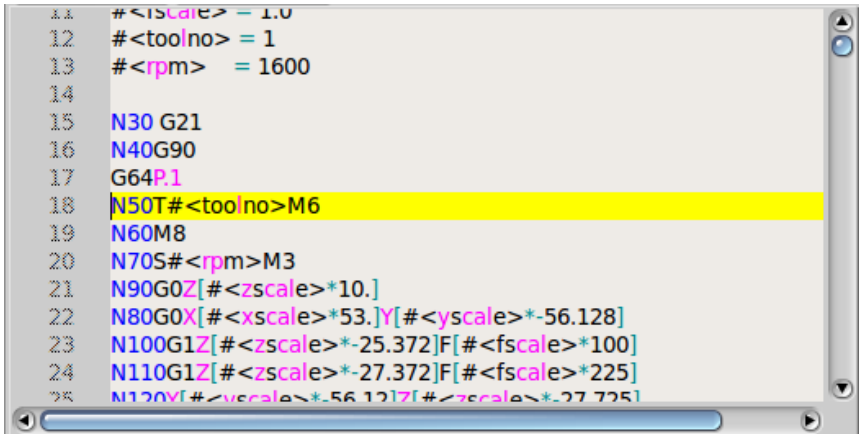
Direct program control

Python を使用してウィジェットを直接制御する方法はいくつかあります。

[widget name].redo()

redo one level of changes.
[widget name].undo()
undo one level of changes
[widget name].text_search(direction=True,mixed_case=True,text='G92')
Searches forward (direction = True) or back, +
Searches with mixed case (mixed_case = True) or exact match
[widget name].set_line_number(linenumber)
Sets the line to high light. Uses the sourceview line numbers.
[widget name].get_line_number()
returns the currently high lighted line.
[widget name].line_up()
Moves the High lighted line up one line
[widget name].line_down()
Moves the High lighted line down one line
[widget name].load_file('filename')
loads a file. Using None (not a filename string) will reload the same -
program.

[widget name].get_filename()



7.3.6.27 MDI の履歴

これは、MDI コードを表示および入力するためのものです。

MDI が使用できない場合は、自動的にグレー表示されます。

たとえば、Estop およびプログラムの実行中。

font_size_tree

8〜96 +の整数値は、ツリービューのデフォルトのフォントサイズを選択した値に変更します

font_size_entry

8〜96 +の整数値は、エントリのデフォルトのフォントサイズを選択した値に変更します

use_double_click

True または False、これを True に設定すると、マウスのダブルクリック機能が有効になり、エントリをダブルクリックするとそのコマンドが送信されます。間違ったエントリをダブルクリックすると危険な状況が発生する可能性があるため、この機能を実際のマシンで使用することはお勧めしません。

Using goobject to set the above listed properties

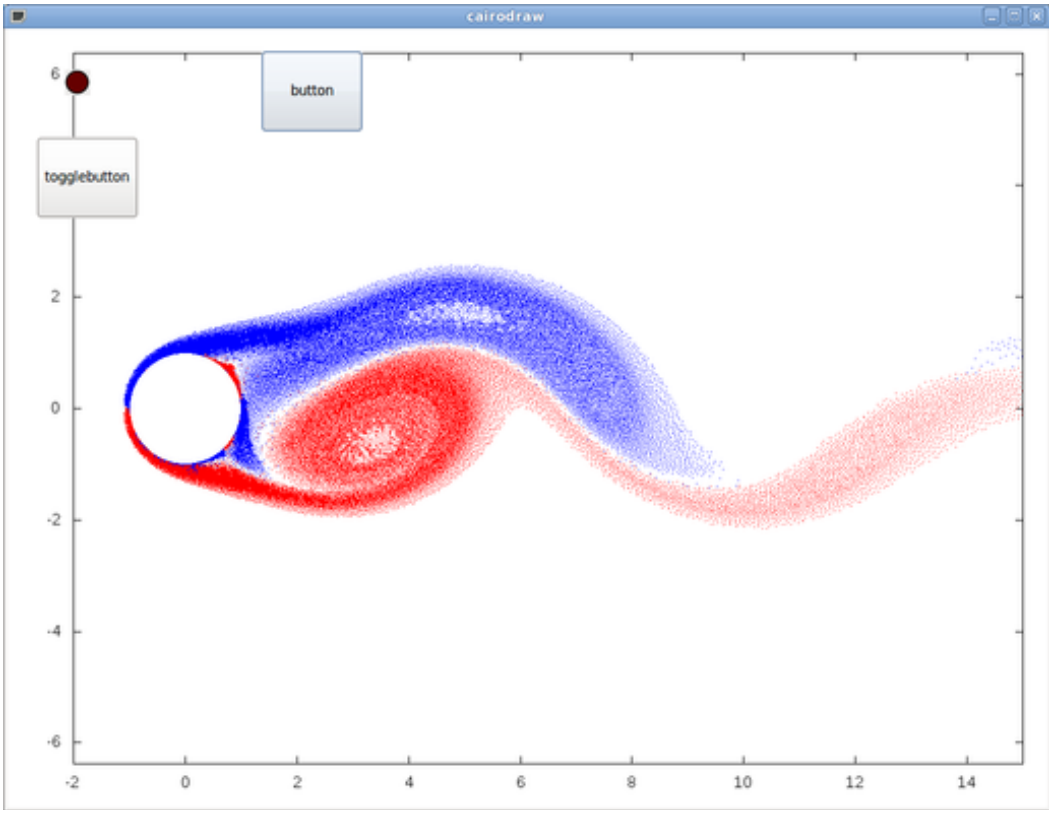
goobject を使用してリストされたプロパティを設定します：[widget name] .set_property（ "font_size_tree"、10）[widget ame] .set_property（ "font_size_20）[widget ame] .set_property（" use_double_click "、False）

7.3.6.28 アニメーション化された関数図：ビットマップ内の HAL ウィジェット

一部のアプリケーションでは、機能図のような背景画像を用意し、その図の適切な場所にウィジェットを配置することが望ましい場合があります。 適切な組み合わせは、.png ファイルなどのビットマップ背景画像を設定し、

gladevcp ウィンドウを固定サイズにし、gladeFixed ウィジェットを使用してこの画像にウィジェットを配置することです。

以下の例のコードは、configs / apps / gladevcp / animated-backdrop にあります。



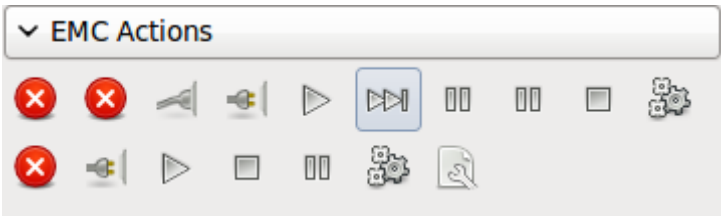
7.3.7 アクションウィジェットリファレンス

GladeVcp には、Glade ユーザーインターフェイスエディター用の VCP アクションウィジェットと呼ばれる「既定のアクション」のコレクションが含まれています。 HAL ピンと相互作用する HAL ウィジェット以外に、VCP アクションは LinuxCNC および G コードインタープリターと相互作用します。

VCP アクションウィジェットは Gtk.Action ウィジェットから派生しています。 一言で言えば、アクションウィジェット：

- Glade で利用可能なオブジェクトです
- それ自体には視覚的な外観はありません
- 目的：メニュー、ツールボタン、ボタンなどの表示された機密性の高い UI コンポーネントをコマンドに関連付けます。 これらのウィジェットの General！RelatedAction プロパティを参照してください。
- 「固定アクション」は、関連する UI コンポーネントがトリガーされたときに実行されます（ボタンを押す、メニューをクリックするなど）。
- Python プログラミングに頼らずにコマンドを実行する簡単な方法を提供します

Glade での VCP アクションの外観は、おおまかに次のとおりです。



ツールチップホバーが説明を提供します。

1.1.1.1 VCP アクションウィジェット

VCP アクションウィジェットはワンショットタイプのウィジェットです。 これらは単一のアクションを実装し、単純なボタン、メニューエントリ、またはラジオ/チェックグループで使用します。

7.3.7.1 VCP アクション python

このウィジェットは、小さな任意の Python コードを実行するために使用されます。

コマンド文字列は、重要な機能にアクセスするために特別なキーワードを使用する場合があります。

- linuxcnc ステータスに使用される Gstat ライブラリにアクセスするための GSTAT
- linuxcncpython モジュールを介して linuxcnc のステータスにアクセスするための STAT
- linuxcncpython モジュールを介して linuxcnc のコマンドにアクセスするための CMD
- ハンドラーファイル機能にアクセスするための EXT（使用可能な場合）
- linuxcncpython モジュールにアクセスするための linuxcnc
- ウィジェットインスタンスにアクセスするための自己

ウィジェットをいつアクティブにするかを選択するオプションがあります。

コマンドを実行する前にモードを設定するオプションがあります。

ターミナルにメッセージを出力するコマンドの例：

```
print('action activated')
```

マシンをオフ状態に設定するコマンドの例：

```
CMD.state(linuxcnc.STATE_OFF)
```

データを渡すハンドラー関数を呼び出すコマンドの例：

```
EXT.on_button_press(self, 100)
```

コロンを使用して、複数のコマンドを区切ることができます。

```
print('Set Machine Off');CMD.state(linuxcnc.STATE_OFF)
```

7.3.7.2 VCPToggleAction ウィジェット

これらはバイモーダルウィジェットです。 これらは 2 つのアクションを実装するか、2 番目の（通常は押された）状態を使用して、現在アクションが実行されていることを示します。 トグルアクションは、

ToggleButtons、ToggleToolButtons、またはトグルメニュー項目での使用を目的としています。 シンプレックスの例は、ESTOP トグルボタンです。

現在、次のウィジェットを使用できます。

- ESTOP トグルは、状態に応じて ESTOP または ESTOP_RESET コマンドを LinuxCNC に送信します。
- ON / OFF トグルは、STATE_ON および STATE_OFF コマンドを送信します。
- Pause / Resume は、AUTO_PAUSE または AUTO_RESUME コマンドを送信します。

次のトグルアクションにはコマンドが 1 つだけ関連付けられており、押された状態を使用して、要求された操作が実行中であることを示します。

- 実行トグルは AUTO_RUN コマンドを送信し、インタプリタが再びアイドル状態になるまで押された状態で待機します。
- インタープリターがアクティブ状態になり（G コードを実行している）、ユーザーが AUTO_ABORT コマンドを送信できるようになるまで、停止トグルは非アクティブです。
- MDI トグルは、指定された MDI コマンドを送信し、押された非アクティブ状態で完了するのを待ちます。

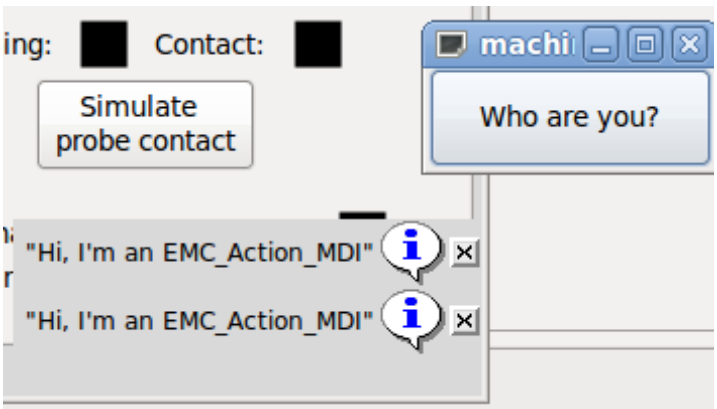
7.3.7.3 Action_MDI トグルウィジェットと Action_MDI ウィジェット

これらのウィジェットは、任意の MDI コマンドを実行する手段を提供します。 Action_MDI ウィジェットは、Action_MDI Toggle のようにコマンドの完了を待機しません。これは、コマンドが完了するまで無効のままです。

7.3.7.4 簡単な例：ボタンを押したときに MDI コマンドを実行する

configs/apps/gladevcp/mdi-command-example/whoareyou.ui is a Glade UI file which conveys the basics:

Glade で開き、どのように行われるかを調べます。 Axis を起動し、gladevcpwhoareyou.ui を使用してターミナルウィンドウから起動します。 hal_action_mdi1 アクションとその MDI コマンドプロパティを参照してください。これは実行されるだけなので（MSG、「こんにちは、私は VCP_Action_MDI です」）、Axis に次のようなメッセージポップアップが表示されます。



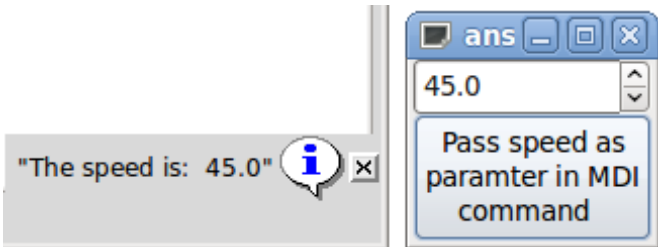
マシンがオフの場合、非常停止中、またはインタプリタが実行中の場合、Action_MDI アクションに関連付けられたボタンがグレー表示されます。 マシンの電源がオンまたはオフになり、プログラムがアイドル状態になると、自動的にアクティブになります。

7.3.7.5 Action_MDI および ToggleAction_MDI ウィジェットを使用したパラメーターの受け渡し

オプションで、MDI コマンド文字列は、インタプリタに渡される前にパラメータが置換される場合があります。パラメータは現在、GladeVCP コンポーネントの HAL ピンの名前である可能性があります。 仕組みは次のとおりです。

- speed という名前の HALSpinBox があり、その現在の値を MDI コマンドのパラメーターとして渡したいとします。
- HAL SpinBox には、speed-f という名前の float タイプの HAL ピンがあります（HalWidgets の説明を参照）。
- MDI コマンドでこの値を置き換えるには、次のように囲まれた HAL ピン名を挿入します。$ {pin-name}
- 上記の HALSpinBox の場合、何が起こっているかを示すためだけに（MSG、「速度は：$ {speed-f}」）を使用できます。

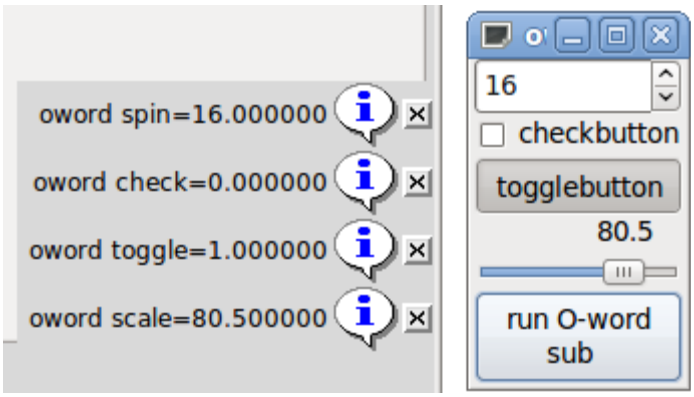
UI ファイルの例は configs / apps / gladevcp / mdi-command-example /speed.ui です。 実行すると次のようになります。



7.3.7.6 高度な例：O-word サブルーチンへのパラメーターのフィード

MDI コマンドで O-word サブルーチンを呼び出し、HAL ピンの値を実際のパラメーターとして渡すことはまったく問題ありません。 UI ファイルの例は configs / apps / gladevcp / mdi-command-example /owordsub.ui にあります。

Axis が見つけられるように nc_files / gladevcp_lib / oword.ngc を配置し、ターミナルウィンドウから gladevcpowordsub.ui を実行します。 これは次のようになります。



7.3.7.7 MDI アクションの準備、およびその後のクリーンアップ

MDI アクションの準備と、その後のクリーンアップ LinuxCNC G コードインタープリターには、フィード、スピンドル速度、相対/絶対モードなどの単一のグローバル変数セットがあります。 G コードコマンドまたは O ワードサブルーチンを使用する場合、これらの変数の一部はコマンドまたはサブルーチンによって変更される可能性があります。たとえば、プロービングサブルーチンは、フィード値を非常に低く設定する可能性があります。 これ以上の予防措置がない場合、以前のフィード設定はプローブサブルーチンの値によって上書きされます。

LinuxCNC ToggleAction_MDI で実行される特定の O-word サブルーチンまたは G コードステートメントのこの驚くべき望ましくない副作用に対処するには、MDI 前および MDI 後のハンドラーを特定の LinuxCNCToggleAction_MDI に関連付けることができます。 これらのハンドラーはオプションであり、MDI アクションを実行する前に状態を保存し、後でそれを以前の値に復元する方法を提供します。 信号名は mdi-command-start と mdi-command-stop です。 ハンドラー名は、他のハンドラーと同じように Glade で設定できます。

このようなハンドラーによってフィード値が保存および復元される方法の例を次に示します（LinuxCNC コマンドおよびステータスチャネルは、VCP_ActionBase クラスを介して self.linuxcnc および self.stat として使用できることに注意してください。

```
def on_mdi_command_start(self, action, userdata=None):
action.stat.poll()
self.start_feed = action.stat.settings[1]
def on_mdi_command_stop(self, action, userdata=None):
action.linuxcnc.mdi('F%.1f' % (self.start_feed))
while action.linuxcnc.wait_complete() == -1:
pass
```

Action_MDIToggle ウィジェットのみがこれらのシグナルをサポートします。

Note

LinuxCNC の今後のリリースでは、新しい M コード M70～M72 が使用可能になり、サブルーチン呼び出しの前に状態を保存し、戻り時に状態を復元するのがはるかに簡単になります。

7.3.7.8 LinuxCNCStat オブジェクトを使用してステータスの変更を処理する

多くのアクションは LinuxCNC ステータスに依存します-それは手動、MDI、または自動モードですか？ プログラムは実行中、一時停止中、またはアイドル状態ですか？ G コードプログラムの実行中は MDI コマンドを開始できないため、これに注意する必要があります。 多くの LinuxCNC アクションはこれを自動的に処理し、操作が現在不可能な場合、関連するボタンとメニューエントリは非アクティブ化されます。

アクションよりも低いレベルの Python イベントハンドラーを使用する場合は、ステータスの依存関係を自分で処理する必要があります。 この目的のために、LinuxCNCStat ウィジェットがあります。LinuxCNC ステータスの変更をイベントハンドラーに関連付けるためです。

LinuxCNC Stat には、表示されるコンポーネントはありません。Glade を使用して UI に追加するだけです。 追加すると、ハンドラーを次のシグナルに関連付けることができます。

- 状態関連：非常停止状態が発生したとき、リセットされたとき、マシンの電源がオンになったとき、またはオフになったときに発行されます
  - state-estop
  - state-estop-reset
  - state-on,
  - state-off
- モード関連：LinuxCNC がその特定のモードに入ったときに発行されます

– mode-manual

– mode-mdi

– mode-auto

- インタプリタ関連：G コードインタプリタがそのモードに変更されたときに発行されます

– interp-run

– interp-idle

– interp-paused

– interp-reading

– interp-waiting

– file-loaded

– line-changed

ホーミング関連：linuxcnc がホームに接続されているかどうかに関係なく発行されます

– all-homed

7.3.8 GladeVCP プログラミング

1.1.1.1 ユーザー定義のアクション

ほとんどのウィジェットセットとそれに関連するユーザーインターフェイスエディターは、コールバックの概念をサポートしています-UI で何かが発生したときに実行されるユーザー作成コードの関数-マウスクリック、入力された文字、マウスの動き、タイマーイベント、ウィンドウの非表示、 露出など。

HAL 出力ウィジェットは通常、ボタンを押すなどの入力タイプのイベントを、事前定義されたコールバックなどを使用して、関連付けられた HAL ピンの値の変更にマップします。 PyVCP 内では、これが実際にサポートされている唯一のタイプのイベント処理です。MDI コマンドを実行して G コードサブルーチンを呼び出すなど、より複雑な処理はサポートされていません。

GladeVCP 内では、HAL ピンの変更は、GTK +のイベント（信号と呼ばれる）の一般的なクラスの 1 つのタイプにすぎません。 ほとんどのウィジェットはそのようなシグナルを生成する可能性があり、Glade エディターはそのようなシグナルを Python メソッドまたは関数名に関連付けることをサポートしています。

ユーザー定義のアクションを使用することにした場合、あなたの仕事は、クラスメソッド（または単純な場合は関数のみ）を Glade でイベントハンドラーとして参照できる Python モジュールを作成することです。 GladeVCP は、起動時にモジュールをインポートする方法を提供し、GladeUI の説明で設定されているウィジェットシグナルにイベントハンドラーを自動的にリンクします。

7.3.8.1 例：Python でカスタムユーザーコールバックを追加する

これは、アイデアを伝えるための最小限の例にすぎません。詳細は、このセクションの残りの部分で説明されています。

GladeVCP は、HAL ピンを操作または表示できるだけでなく、Python で通常のイベントハンドラーを作成することもできます。 これは、とりわけ、MDI コマンドを実行するために使用できます。 方法は次のとおりです。

そのような Python モジュールを作成し、たとえば次のように保存します。 handlers.py：

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

Glade で、ボタンまたは HAL ボタンを定義し、[信号]タブを選択し、GtkButton プロパティで押された線を選択します。 そこで on_button_press と入力し、Glade ファイルを保存します。

次に、オプション-uhandlers.py を gladevcp コマンドラインに追加します。 イベントハンドラーが複数のファイルに分散している場合は、複数の-u <pyfilename>オプションを追加するだけです。

これで、ボタンを押すと、コールバック関数で設定されているため、ラベルが変更されます。

-u フラグの機能：このファイル内のすべての Python 関数が収集され、Gtk ウィジェットの潜在的なコールバックハンドラーとして設定されます。これらは、GladeSignals タブから参照できます。 コールバックハンドラーは、上記の GtkButton インスタンスのように、特定のオブジェクトインスタンスをパラメーターとして使用して呼び出されるため、そこから任意の GtkButton メソッドを適用できます。

または、MDI コマンドを呼び出すなど、もっと便利なことをしてください。

7.3.8.2 HAL 値変更イベント

LED のような HAL 入力ウィジェットは、HAL ピンの状態（オン/オフ）をウィジェットの光学的外観（LED の点灯/暗）に自動的に関連付けます。

この組み込み機能に加えて、変更コールバックを、事前定義された HAL ウィジェットのピンを含む任意の HAL ピンに関連付けることができます。 これは、一般的なウィジェットアプリケーションのイベント駆動型構造にうまく適合します。マウスクリック、キー、タイマーの期限切れ、HAL ピンの値の変更など、すべてのアクティビティがコールバックを生成し、同じ直交メカニズムによって処理されます。

特定の HAL ウィジェットに関連付けられていないユーザー定義の HAL ピンの場合、信号名は値が変更されます。 詳細については、以下の「HAL ピンの追加」セクションを参照してください。

HAL ウィジェットには、hal-pin-changed と呼ばれる事前定義された信号が付属しています。 詳細については、Hal ウィジェットのセクションを参照してください。

7.3.8.3 プログラミングモデル

全体的なアプローチは次のとおりです。

- Glade を使用して UI を設計し、ウィジェットに関連付けられたアクションが必要な場所にシグナルハンドラーを設定します
- 呼び出し可能なオブジェクトを含む Python モジュールを作成します（以下のハンドラーモデルを参照）
- -u <module>オプションを使用して、モジュールのパス名を gladevcp に渡します
- gladevcp はモジュールをインポートし、シグナルハンドラーを検査して、ウィジェットツリーに接続します
- メインイベントループが実行されます。

単純なハンドラーモデル単純なタスクの場合、Glade シグナルハンドラーにちなんで名付けられた関数を定義するだけで十分です。 これらは、対応するイベントがウィジェットツリーで発生したときに呼び出されます。 これは簡単な例です。Gtk ボタンまたは HAL ボタンの押されたシグナルが on_button_press と呼ばれるコールバックにリンクされていることを前提としています。

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

この関数を Python ファイルに追加し、次のように実行します。

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

ハンドラー間の通信はグローバル変数を経由する必要があることに注意してください。グローバル変数は適切にスケーリングされず、積極的に非 Python です。 これが、クラスベースのハンドラーモデルを思いついた理由です。

クラスベースのハンドラーモデルここでの考え方は、ハンドラーがクラスメソッドにリンクされているということです。 基盤となるクラスは、GladeVCP の起動時にインスタンス化および検査され、シグナルハンドラーとしてウィジェットツリーにリンクされます。 したがって、今のタスクは次のように書くことです。

- 1 つのモジュール内または複数のモジュールに分割された 1 つまたは複数のメソッドを持つ 1 つ以上の複数のクラス定義
- クラスインスタンスのリストを GladeVCP に返す各モジュールの関数 get_handlers-それらのメソッド名はシグナルハンドラーにリンクされます

最小限のユーザー定義ハンドラーのサンプルモジュールを次に示します。

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

これで、on_this_signal がウィジェットツリーのシグナルハンドラーとして使用できるようになります。

GladeVCP 固有の信号 HAL 入力に応答する GladevCP パネルの場合、ハンドラーコードが GladeVCP パネルが現在アクティブで表示されていることを通知できることが重要な場合があります。 たとえば、Touchy インターフェイス内のパネルは、touchy.cycle-start に接続されたスイッチが操作されたときにアクションを実行する必要がある場合があります（ネイティブタブが同じボタンに対して異なる応答をするのと同じ方法で）。これを可能にするために信号 GUI（執筆時点では Touchy のみ）から埋め込みタブに送信されます。 信号のタイプは「Gladevcp」で、送信される 2 つのメッセージは「Visible」と「Hidden」です。 （シグナルの長さは 20 文字に固定されているため、比較では最初の文字のみを使用する必要があることに注意してください。したがって、以下の[：7]です。）これらのシグナルのサンプルハンドラーは次のとおりです。

```
# This catches our messages from another program
def event(self,w,event):
print event.message_type,event.data
if event.message_type == 'Gladevcp':
if event.data[:7] == 'Visible':
self.active = True
else:
self.active = False
```

```
# connect to client-events from the host GUI
def on_map_event(self, widget, data=None):
top = widget.get_toplevel()
print "map event"
top.connect('client-event', self.event)
```

get_handlers プロトコルモジュールの検査中に GladeVCP が関数 get_handlers を検出すると、次のように呼び出します。

```
get_handlers（halcomp、builder、useropts）
```

引数は次のとおりです。

- halcomp-構築中の HAL コンポーネントを指します
- ビルダー-ウィジェットツリー-UI 定義を読み取った結果（GtkBuilder または libglade タイプのオブジェクトを参照）
- useropts-gladevcp コマンドラインから収集された文字列のリスト-U <useropts>オプション

次に、GladeVCP はクラスインスタンスのリストを検査し、それらのメソッド名を取得します。 修飾メソッド名は、シグナルハンドラーとしてウィジェットツリーに接続されます。 _（アンダースコア）で始まらないメソッド名のみが考慮されます。

Glade UI に libglade または新しい GtkBuilder 形式のどちらを使用しているかに関係なく、ウィジェットは常に builder.get_object（<widgetname>）として参照できることに注意してください。 また、ウィジェットの完全なリストは、UI 形式に関係なく、builder.get_objects（）として利用できます。

7.3.8.4 初期化シーケンス

get_handlers（）関数がどの状態で呼び出されるかを知ることは重要です。そうすれば、そこで何が安全で何が安全でないかを知ることができます。 まず、モジュールがインポートされ、コマンドライン順に初期化されます。 インポートが成功すると、get_handlers（）が次の状態で呼び出されます。

- ウィジェットツリーは作成されますが、まだ実現されていません（トップレベルの window.show（）はまだ実行されていません）
- halcomp HAL コンポーネントがセットアップされ、すべての HAL ウィジェットのピンがすでに追加されています
- この時点では halcomp.ready（）がまだ呼び出されていないため、HAL ピンを追加しても安全です。そのため、たとえばクラス__init __（）メソッドで独自のピンを追加できます。

すべてのモジュールがインポートされ、メソッド名が抽出されると、次の手順が実行されます。

- すべての修飾メソッド名は、connect_signals（）/ signal_autoconnect（）を使用してウィジェットツリーに接続されます（インポートされた UI のタイプ（GtkBuilder と古い libglade 形式）によって異なります）。
- HAL コンポーネントは halcomp.ready（）で終了します

- ウィンドウ ID が引数として渡された場合、ウィジェットツリーはこのウィンドウで実行されるように親が変更され、Glade のトップレベルの window1 は破棄されます（FAQ を参照）。
- HAL コマンドファイルが-Hhalfile で渡された場合、halcmd で実行されます。
- Gtk メインループが実行されます。

したがって、ハンドラークラスが初期化されると、すべてのウィジェットは存在しますが、まだ実現されていません（画面に表示されます）。 また、HAL コンポーネントも準備ができていないため、__ init __（）メソッドでピンの値にアクセスするのは安全ではありません。

HAL ピンに安全にアクセスした後、プログラムの開始時にコールバックを実行する場合は、ハンドラーをトップレベルウィンドウ 1 の実現シグナルに接続します（これが唯一の本当の目的である可能性があります）。 この時点で、GladeVCP はすべてのセットアップタスクで完了し、ハーフイルが実行され、GladeVCP が Gtk メインループに入るところです。

7.3.8.5 同じ名前の複数のコールバック

クラス内では、メソッド名は一意である必要があります。 ただし、同じ名前のメソッドを使用して get_handlers（）によって複数のクラスインスタンスを GladeVCP に渡してもかまいません。 対応するシグナルが発生すると、これらのメソッドは定義順に呼び出されます-モジュールごとに、モジュール内では、クラスインスタンスが get_handlers（）によって返される順序で呼び出されます。

7.3.8.6 GladeVCP -U <useropts>フラグ

ハンドラークラスに役立つ可能性のある考えられるオプションに対して GladeVCP を拡張する代わりに、-U <useroption>フラグを使用できます（必要に応じて繰り返し）。 このフラグは、<useroption>文字列のリストを収集します。 このリストは get_handlers（）関数（useropts 引数）に渡されます。 コードは、これらの文字列を適切と思われるように自由に解釈できます。 考えられる使用法は、次のように get_handlers（）の Pythonexec 関数にそれらを渡すことです。

```
debug = 0
...
def get_handlers(halcomp,builder,useropts):
...
global debug # assuming there's a global var
for cmd in useropts:
exec cmd in globals()
```

このようにして、gladevcp -U オプションを使用して、任意の Python ステートメントをモジュールに渡すことができます。次に例を示します。

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

これにより、デバッグが 2 に設定され、モジュールが実際に実行したことを確認する必要があります。

7.3.8.7 GladeVCP の永続変数

以前の形式と pyvcp の GladeVCP の厄介な側面は、テキスト入力、スライダー、スピンボックス、トグルボタンなどを使用して値と HAL ピンを変更できるという事実ですが、それらの設定は LinuxCNC の次回の実行時に保存および復元されません- パネルまたはウィジェット定義で設定されたデフォルト値から開始します。

GladeVCP には、HAL ウィジェットの状態とプログラム変数（実際には、int、float、bool、または string 型のインスタンス属性）を保存および復元するための使いやすいメカニズムがあります。

このメカニズムは、一般的な.ini ファイル形式を使用して、永続属性を保存および再ロードします。

永続性、プログラムバージョン、および署名チェック Glade でウィジェットの名前を変更、追加、または削除することを想像してみてください。以前のプログラムバージョンからの.ini ファイル、またはまったく異なるユーザーインターフェイスでは、変数と タイプが変更された可能性があります。

GladeVCP は、保存および復元されるすべてのオブジェクト名とタイプに依存するシグニチャによってこの状況を検出します。 シグニチャが一致しない場合は、デフォルト設定の新しい.ini ファイルが生成されます。

7.3.8.8 永続変数の使用

Gtk ウィジェットの状態、HAL ウィジェットの出力ピンの値やハンドラークラスのクラス属性のいずれかを呼び出し間で保持する場合は、次の手順に従います。

- gladevcp.persistence モジュールをインポートします
- どのインスタンス属性と、保持したいデフォルト値（ある場合）を決定します
- どのウィジェットの状態を保持するかを決定します
- ネストされたディクショナリを介して、ハンドラクラスの init（）メソッドでこれらの決定を次のように記述します。

```
def __init__(self, halcomp,builder,useropts):
self.halcomp = halcomp
self.builder = builder
self.useropts = useropts
self.defaults = {
# the following names will be saved/restored as method attributes
# the save/restore mechanism is strongly typed - the variables type will be derived  -
from the type of the
# initialization value. Currently supported types are: int, float, bool, string
IniFile.vars : { 'nhits' : 0, 'a': 1.67, 'd': True ,'c' : "a string"},
# to save/restore all widget's state which might remotely make sense, add this:
IniFile.widgets : widget_defaults(builder.get_objects())
# a sensible alternative might be to retain only all HAL output widgets' state:
# IniFile.widgets: widget_defaults(select_widgets(self.builder.get_objects(),  -
```

```
    hal_only=True,output_only = True)),
    }
```

次に、.ini ファイルをこの記述子に関連付けます。

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

restore_state（）の後、以下を実行している場合、self には属性が設定されます。

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

タイプは復元時に保存および保持されることに注意してください。 この例では、ini ファイルが存在しないか、self.defaults のデフォルト値が設定されていることを前提としています。

この呪文の後、次の IniFil メソッドを使用できます。

ini.save_state（obj）

Self.defaults の IniFile.widgets で説明されているように、IniFil.vars ディクショナリに従って obj の属性とウィジェットの状態を保存します

ini.create_default_ini（）

デフォルト値で.ini ファイルを作成する

ini.restore_state（obj）

上記のようにデフォルトに保存/初期化された HAL 出力ピンと obj の属性を復元します

7.3.8.9 Gladvcp シャットダウン時に状態を保存する

終了時にウィジェットや変数の状態を保存するには、次の手順に従います。

- いくつかの内部ウィジェットを選択します（たとえば、テーブルなど、タイプは重要ではありません）。
- [信号]タブで、[GtkObject]を選択します。 最初の列に破棄信号が表示されます。
- ハンドラー名を追加します。例： on_destroy を 2 番目の列に移動します。
- 以下のような Python ハンドラーを追加します。

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

これにより、パネルが Axis に埋め込まれているか、スタンドアロンウィンドウに埋め込まれているかに関係なく、状態が保存され、GladeVCP が適切にシャットダウンされます。

警告

破棄イベントを接続するために window1（トップレベルウィンドウ）を使用しないでください。 パネルが Axis 内に埋め込まれている場合、GladeVCP パネルが Axis と対話する方法が原因で、window1 は destroy イベントを適切に受信しません。 ただし、シャットダウンするとすべてのウィジェットが破棄されるため、誰でも破棄できます。 推奨：第 2 レベルのウィジェットを使用します。たとえば、パネルにテーブルコンテナーがある場合は、それを使用します。

次回 GladeVCP アプリケーションを起動すると、ウィジェットはアプリケーションが閉じられたときの状態で起動するはずです。

警告

GtkWidget 行には同様に聞こえる destroy-event があります-on_destroy ハンドラーに接続するためにそれを使用しないでください、それは機能しません-GtkObject 行からの destroy イベントを使用することを確認してください。

7.3.8.10 Ctrl-C を押したときの状態の保存

デフォルトでは、Ctrl-C イベントに対する GladeVCP の反応は、状態を保存せずに終了することです。 このケースを確実にカバーするために、Ctrl-C で自動的に呼び出されるハンドラー呼び出し on_unix_signal を追加します（実際には SIGINT および SIGTERM シグナルで呼び出されます）。 例

```
def on_unix_signal(self,signum,stack_frame):
print "on_unix_signal(): signal %d received, saving state" % (signum)
self.ini.save_state(self)
```

7.3.8.11 .ini ファイルを手動で編集する

それは可能ですが、編集に構文エラーまたはタイプエラーがある場合、self.defaults の値が編集を上書きすることに注意してください。 エラーが検出され、コンソールメッセージがそのことを示唆し、不正な inifile の名前が.BAD サフィックスに変更されます。 後続の不良 ini ファイルは、以前の.BAD ファイルを上書きします。

7.3.8.12 HAL ピンの追加

特定の HAL ウィジェットに関連付けられていない HAL ピンが必要な場合は、次のように追加します。

```
import hal_glib
...
# in your handler class __init__():
self.example_trigger = hal_glib.GPin(halcomp.newpin('example-trigger', hal.HAL_BIT, hal.
-
```

```
HAL_IN))
```

このピンの値が変更されたときにコールバックを取得するには、値変更コールバックをこのピンに関連付け、次を追加します。

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

コールバックメソッド（または関数、この場合は自己パラメーターを省略）を定義します。

```
# note '_' - this method will not be visible to the widget tree
def _on_example_trigger_change(self,pin,userdata=None):
print "pin value changed to:" % (pin.get())
```

7.3.8.13 タイマーの追加

GladeVCP は GObject 基本クラスに依存する Gtk ウィジェットを使用するため、完全な glib が機能的に利用可能です。 タイマーコールバックの例を次に示します。

```
def _on_timer_tick(self,userdata=None):
...
return True # to restart the timer; return False for on-shot
...
# demonstrate a slow background timer - granularity is one second
# for a faster timer (granularity 1 ms), use this:
# glib.timeout_add(100, self._on_timer_tick,userdata) # 10Hz
glib.timeout_add_seconds(1, self._on_timer_tick)
```

7.3.8.14 プログラムによる HAL ウィジェットのプロパティの設定

空き地では、ウィジェットのプロパティは通常、編集中に固定されて設定されます。 ただし、実行時にウィジェットのプロパティを設定することはできます。たとえば、ini ファイルの値から設定できます。これは通常、ハンドラーの初期化コードで行われます。 HAL ピン値からプロパティを設定することも可能です。

次の例（meter と呼ばれる HAL Meter ウィジェットを想定）では、メーターの最小値は起動時に INI ファイルパラメーターから設定され、最大値は HAL ピンを介して設定されます。これにより、ウィジェットのスケールが動的に再調整されます。

```
import linuxcnc
import os
import hal
import hal_glib
class HandlerClass:
def _on_max_value_change(self,hal_pin,data=None):
self.meter.max = float(hal_pin.get())
self.meter.queue_draw() # force a widget redraw
```

```
def __init__(self, halcomp,builder,useropts):
self.builder = builder
# hal pin with change callback.
# When the pin's value changes the callback is executed.
self.max_value = hal_glib.GPin(halcomp.newpin('max-value', hal.HAL_FLOAT, hal.  -
HAL_IN))
self.max_value.connect('value-changed', self._on_max_value_change)
inifile = linuxcnc.ini(os.getenv("INI_FILE_NAME"))
mmin = float(inifile.find("METER", "MIN") or 0.0)
self.meter = self.builder.get_object('meter')
self.meter.min = mmin
def get_handlers(halcomp,builder,useropts):
return [HandlerClass(halcomp,builder,useropts)]
```

7.3.8.15 例、および独自の GladeVCP アプリケーションのローリング

独自のプロジェクトの例とスターターを実行するには、linuxcnc_root_directory / configs / apps / gladevcp にアクセスしてください。

7.3.9 よくある質問

1.起動直後にハンドラー関数で予期しないマップ解除イベントが発生します。 これは何ですか？

これは、GladeUI ファイルで window1Visible プロパティが True に設定され、GladeVCP ウィンドウの親が Axis または touchy に変更された結果です。 トップレベルウィンドウを含む GladeVCP ウィジェットツリーが作成され、Axis に再ペアレント化され、そのトップレベルウィンドウが孤立したままになります。 この役に立たない空のウィンドウがぶら下がるのを避けるために、マップされていない（非表示になっている）ため、マップ解除信号が表示されます。 推奨される修正：window1.visible を False に設定し、最初のマップ解除イベントを無視します。

2. GladeVCP プログラムが起動しますが、期待する場所にウィンドウが表示されませんか？

Axis が GladeVCP に割り当てるウィンドウは、結合されたすべての子ウィジェットの自然なサイズを取得します。 サイズ（幅や高さ）をリクエストするのは子ウィジェットの仕事です。 ただし、すべてのウィジェットが 0 より大きい幅を要求するわけではありません。たとえば、現在の形式のグラフウィジェットです。 Glade ファイルにそのようなウィジェットがあり、それがレイアウトを定義するウィジェットである場合は、その幅を明示的に設定することをお勧めします。 Glade で window1 の width プロパティと height プロパティを設定しても意味がないことに注意してください。これは、このウィンドウが再ペアレント化中に孤立し、そのジオメトリがレイアウトに影響を与えないためです（上記を参照）。 一般的なルールは次のとおりです。 gladevcp<uifile>を使用して UI ファイルを手動で実行し、そのウィンドウに適切なジオメトリがある場合は、Axis でも正しく表示されるはずです。

3. LED を点滅させたいのですが、点滅しません

チェックボタンをチェックして、100 ミリ秒間隔で点滅させました。 点滅せず、起動時の警告が表示されます。警告：タイプ「gint」の値「0」が無効であるか、タイプ「gint」のプロパティ「led-blink-rate」の範囲外ですか？ これは空き地のバグのようです。 まばたき率のフィールドに入力して、もう一度保存するだけです。これでうまくいきます。

4.ウィンドウの破棄信号にリンクされた on_destroy ハンドラーを定義しましたが、Axis の gladevcp パネルが Axis を閉じたときに状態を保存しません

このハンドラーは window1 にリンクされている可能性が非常に高く、親の変更のため、この目的には使用できません。 on_destroy ハンドラーを内部ウィンドウの destroy シグナルにリンクしてください。 たとえば、window1 内にノートブックがあり、on_destroy をノートブックの destroy シグナルにリンクすると、正常に機能します。 window1 では機能しません。

5. HAL ピン値に応じて HAL_Label ウィジェットの背景色またはテキストを設定したい

configs / apps / gladevcp / colored-label の例を参照してください。 GtkLabel ウィジェット（および HAL_Label は GtkLabel から派生）の背景色を設定するのは少し注意が必要です。 GtkLabel ウィジェットには、パフォーマンス上の理由から独自のウィンドウオブジェクトはなく、ウィンドウオブジェクトのみが背景色を持つことができます。 解決策は、ラベルを EventBox コンテナで囲むことです。このコンテナにはウィンドウがありますが、それ以外の場合は表示されません。coloredlabel.ui ファイルを参照してください。

6. glade で hal_spinbutton ウィジェットを定義し、対応する調整でデフォルト値のプロパティを設定しました。それはゼロを思い付く？

これは、Ubuntu 8.04 および 10.04 で配布された古い Gtk バージョンのバグが原因であり、調整を使用するすべてのウィジェットに当てはまる可能性があります。 たとえば http://osdir.com/ml/gtk-app-devel-list/2010-04/-msg00129.html に記載されている回避策では、HAL ピン値が確実に設定されないため、明示的に設定することをお勧めします。 ウィジェット作成中の on_realize シグナルハンドラ。 configs / apps / gladevcp / by-widget / spinbutton。{ui、py}の例を参照してください。

7.3.10 トラブルシューティング

- LinuxCNC の開発バージョンがインストールされていることを確認してください。 axisrc ファイルはもう必要ありません。これは古い GladeVcpwiki ページで言及されていました。
- ターミナルウィンドウから GladeVCP または Axis を実行します。 Python エラーが発生した場合は、新しい/usr/lib/python2.6/dist-packages/_hal.so 以外に/usr/lib/python2.6/dist-file が残っているかどうかを確認してください（アンダースコアに注意してください）。 はいの場合、hal.so ファイルを削除します。 同じディレクトリ内の hal.py に置き換えられ、インポートメカニズムが混乱します。
- ランインプレースを使用している場合は、make clean を実行して、誤って残った hal.so ファイルをすべて削除してから、make を実行します。

- HAL_table または HAL_HBox ウィジェットを使用している場合は、HAL ピンが関連付けられていることに注意してください。これはデフォルトでオフになっています。 このピンは、これらのコンテナの子がアクティブであるかどうかを制御します。

7.3.11 実装上の注意：Axis でのキー処理

キーの処理は問題なく機能すると考えていますが、これは新しいコードであるため、問題に注意できるようにお知らせします。 エラーや奇妙な動作をお知らせください。 これが物語です：

Axis は TkInter ウィジェットセットを使用します。 GladeVCP アプリケーションは Gtk ウィジェットを使用し、別のプロセスコンテキストで実行されます。 それらは Xembed プロトコルで Axis に接続されています。 これにより、GladeVCP のような子アプリケーションが親のウィンドウに適切に収まり、理論的にはイベント処理が統合されます。

ただし、これは、親アプリケーションと子アプリケーションの両方が Xembed プロトコルを適切にサポートしていることを前提としています。これは Gtk がサポートしていますが、TkInter はサポートしていません。 この結果、特定のキーがすべての状況で GladeVCP パネルから Axis に適切に転送されないことになります。 これらの状況の 1 つは、エントリまたは SpinButton ウィジェットにフォーカスがあった場合です。この場合、たとえば、Esc キーが Axis に転送されず、本来のように中止され、悲惨な結果を招く可能性があります。

したがって、GladeVCP の主要なイベントは明示的に処理され、選択的に Axis に転送されて、そのような状況が発生しないようにします。 詳細については、lib / python gladevcp / xembed.py の keyboard_forward（）関数を参照してください。

7.3.12 カスタムウィジェットの追加

LinuxCNC Wiki には、GladeVCP にカスタムウィジェットを追加するための情報があります。 GladeVCP カスタムウィジェット

7.3.13 補助 Gladevcp アプリケーション

スクリプト linuxcnc_var によって報告される LINUXCNC_AUX_GLADEVCP および LINUXCNC_AUX_EXAMPLES 項目によって定義されるシステムディレクトリの配置に準拠する、独立してインストールされた gladevcp アプリケーションのサポートが提供されます。

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

LINUXCNC_AUX_GLADEVCP（/ usr / share / linuxcnc / aux_gladevcp）によって定義されたシステムディレクトリは、gladevcp 互換の Python ファイルおよび関連するサブディレクトリの場所を指定します。 Python ファイルは gladevcp の起動時にインポートされ、GUI のサポートでの組み込みの使用法を含む後続の gladevcp アプリケーションで利用できるようになります。

LINUXCNC_AUX_EXAMPLES（/ usr / share / linuxcnc / aux_examples）で定義されたシステムディレクトリは、補助アプリケーションに使用される設定サブディレクトリの例の場所を指定します。 構成選択項目の追加については、getting-started / running-linuxcnc セクションを参照してください。

テストの場合、補助アプリケーションのランタイム仕様は、エクスポートされた環境変数 GLADEVCP_EXTRAS を使用して指定できます。 この変数は、（:)で区切られた 1 つ以上の構成ディレクトリのパスリストである必要があります。 通常、この変数は、linuxcnc を起動するシェル、またはユーザーの〜/ .profile 起動スクリプトで設定されます。 例：

```
export GLADEVCP_EXTRAS=~/mygladevcp:/opt/othergladevcp
```

環境変数 GLADEVCP_EXTRAS で指定されたディレクトリで見つかったファイルは、LINUXNC_AUX_GLADEVCP で指定されたシステムディレクトリのサブディレクトリ内の同じ名前のファイル（例：/ usr / share / linuxcnc / aux_gladevcp）に優先します。 このプロビジョニングにより、開発者は、システムにインストールされたアプリケーションディレクトリを削除せずに、GLADEVCP_EXTRAS をエクスポートしてプライベートアプリケーションディレクトリを指定することにより、アプリケーションをテストできます。 拒否された重複を示すメッセージが stdout に出力されます。

Note

補助 gladevcp アプリケーションのサポートには、importlib という名前の Python モジュールが必要です。このモジュールは、Ubuntu-Lucid などの古いインストールでは使用できない場合があります。

8章 ユーザーインターフェース

8.1 Panelui

8.1.1 序章

Panelui は、ボタンを linuxcnc または HAL にインターフェースするためのユーザースペースコンポーネントです。

MESA 7I73 スタイルのキースキャンコードをデコードし、適切なルーチンを呼び出します。

リアルタイムコンポーネント（サンプラー）から入力を取得します。

サンプラーは、MESA7i73 または sim_matrix_kb コンポーネントから入力を取得します。

Panelui は、INI スタイルのテキストファイルを使用して構成可能で、ボタンタイプ、HAL ピンタイプ、および/またはコマンドを定義します。

Python ベースのハンドラーファイルを使用して拡張し、関数を追加できます。

実際の入力ボタンは瞬間的である必要がありますが、Panelui はこの入力を使用して、トグル、ラジオ、または瞬間的なボタン出力を行います。

8.1.2 コマンドの読み込み

panelui をロードするために使用されるコマンド（オプションの-d debug スイッチを使用）：

```
loadusr -W panelui -d
```

これにより、panelui が初期化され、config フォルダーまたは user フォルダーで INI ファイル panelui.ini が検索されます。 次のコマンドで INI ファイルを検証できます。

```
loadusr pyui
```

これにより、panelui.ini ファイルが読み取られ、修正が試みられ、保存されます。 見つかった場合、端末にエラーを出力します。 通常の HAL ファイルには、次のコマンドが追加されます。

```
# commands needed for panelui loading
#
# sampler is needed for panelui
# cfg= must always be u for panelui. depth sets the available buffer
loadrt sampler cfg=u depth=1025
#uncomment to validate the panelui INI file
#loadusr pyui
# -d = debug, -v = verbose debug
# -d will show you keypress identification and commands called
```

```
# -v is for develeper info
loadusr -W panelui -d
# using simulated buttons instead of the MESA 7I73 card
# so we load the sim_matrix_kb component to convert HAL pins to keyscan codes
loadrt sim_matrix_kb
# connect the compnents together.
# sampler talks to panelui internally
net key-scan sim-matrix-kb.0.out
net key-scan sampler.0.pin.0
# add panelui components to a thread
addf sim-matrix-kb.0 servo-thread
addf sampler.0 servo-thread
```

8.1.3 panelui.ini ファイルリファレンス

KEY WORDS

- KEY =これはボタンが応答するキーを指定するために使用されます。 これは、NONE または ROW 番号と列番号（R1C2 など）にすることができます。 行と列は 1 回だけ使用できます。
- OUTPUT =これは、ボタンの出力タイプ（S32、U32、FLOAT、BIT、NONE、COMMAND など）を設定します。
- DEFAULT =これはグループまたはボタンの開始出力を設定します。
- GROUP =ラジオボタンで、ボタンが対話するグループを指定します。
- GROUP_OUTPUT =は、このボタンがアクティブな場合にグループピンが出力される出力を設定します。
- STATUS_PIN = TRUE の場合、ボタンの現在の状態を反映する HAL ピンが追加されます。
- TRUE_STATE =ボタンが TRUE の場合、HAL ピンの出力を設定します
- FALSE_STATE =ボタンが FALSE の場合、HAL ピンの出力を設定します
- TRUE_COMMAND =は、ボタンが TRUE のときに呼び出されるコマンドと引数を設定します
- FALSE_COMMAND =は、ボタンが FALSE のときに呼び出されるコマンドと引数を設定します。

HAL Prefix

```
[HAL_PREFIX]
NAME= Yourname
```

これにより、HAL ピンのプレフィックスを panelui から任意の名前に変更できます。

ラジオボタンラジオボタンを使用すると、グループ内の 1 つのボタンのみを一度にアクティブにすることができます。 各グループには、グループ内の各ボタンとは別に、独自の出力ピンがあります。

ラジオボタンの定義は、単一の角かっこで囲まれたテキスト RADIO_BUTTON で始まります。

```
[RADIO_BUTTONS]
# The double bracket section(s) define the group(s) of radio buttons.
# The group name must be unique and is case sensitive.
# Groups output is controlled by what button is active not directly by keycode.
# DEFAULT references a button in the group by name and is case sensitive.
[[group1_name]]
KEY = NONE
OUTPUT = FLOAT
DEFAULT = small
# The triple bracket sections define the buttons in this group.
# button names must be unique and are case sensitive.
# There must be at least two buttons in a group.
#
# This button, named 'small'is controller by the row 0 column 1 key.
# It will cause the group output to be .0001 when it is pressed.
# It has no output of it's own, but has a status
# pin which will follow it's current state.
# since this button is in a group, DEFAULT has no bearing.
# since OUTPUT in not 'COMMAND' _COMMAND entries are ignored.
[[[small]]]
KEY = R0C1
GROUP = group1_name
GROUP_OUTPUT = .0001
OUTPUT = NONE
STATUS_PIN = True
TRUE_STATE = TRUE
FALSE_STATE = FALSE
TRUE_COMMAND = NONE, NONE
FALSE_COMMAND = NONE, NONE
DEFAULT = false
# This button, named 'large' is controller by the row 0 column 2 key.
# It will cause the group output to be 1000 when it is pressed.
# It has a S32 output of it's own, will be 20 on true and 0 on false.
# It also has a status pin which will follow it's current state.
# since this button is in a group, DEFAULT has no bearing.
# since OUTPUT in not 'COMMAND' _COMMAND entries are ignored.
[[[large]]]
KEY = R0C2
GROUP = group1_name
GROUP_OUTPUT = 1000
```

```
OUTPUT = S32
STATUS_PIN = True
TRUE_STATE = 20
TRUE_COMMAND = NONE, NONE
FALSE_COMMAND = NONE, NONE
FALSE_STATE = 0
DEFAULT = false
```

トグルボタントグルボタンは、ボタンを押すたびに状態を変更するだけです。 トグルボタンの定義は、単一の角かっこで囲まれたテキスト TOGGLE_BUTTON で始まります。

```
[TOGGLE_BUTTONS]
# Each button name inside double brackets, must be unique and is case sensitive.
# This button, named 'tool_change'is controller by the row 2 column 5 key.
# It has a BIT output, will output 1 on true state and 0 on false state.
# It also has a status pin which will follow it's current state.
# DEFAULT sets this to true when first initialized.
# The _COMMAND are not used since OUTPUT is not set to COMMAND but validation will
# add the lines regardless
[[tool_change]]
KEY = R2C5
OUTPUT = BIT
TRUE_COMMAND = NONE, NONE
FALSE_COMMAND = NONE, NONE
STATUS_PIN = True
DEFAULT = TRUE
TRUE_STATE = 1
FALSE_STATE = 0
```

モーメンタリボタンモーメンタリボタンは、押すと true になり、離すと false になります。 モーメンタリボタンの定義は、単一の角かっこで囲まれたテキスト MOMENTARY_BUTTON で始まります。

```
[MOMENTARY_BUTTONS]
# Each button name inside double brackets, must be unique and is case sensitive.
# This button, named 'spindle_rev'is controller by the row 2 column 3 key.
# It has a COMMAND output, so will use TRUE_COMMAND and FALSE_COMMAND.
# It also has a status pin which will follow it's current state.
# COMMANDs will have a command name and then any required arguments
# This TRUE_COMMAND calls an internal command to start the spindle in reverse at 200
-
rpm
```

```
# If the spindle is already started, it will increase the rpm.
# DEFAULT is not used with Momentary buttons.
# The _STATE are not used since OUTPUT is set to COMMAND but validation will
# add the lines regardless
[[spindle_rev]]
KEY = R2C3
OUTPUT = COMMAND
TRUE_COMMAND = SPINDLE_REVERSE_INCREASE, 200
FALSE_COMMAND = None, NONE
STATUS_PIN = True
DEFAULT = FALSE
TRUE_STATE = 1
FALSE_STATE = 0
```

8.1.4 内部コマンドリファレンス

使用できる内部コマンドは多数あります。

mist_on
mist_off
flood_on
flood_off
estop
estop_reset
machine_off
machine_on
home_all
unhome_all

HOME_SELECTED

- 必須の引数：軸番号（int）

UNHOME_SELECTED

- 必須の引数：軸番号（int）

SPINDLE_FORWARD_ADJUST

- オプションの引数：開始 RPM（int）-デフォルト 100
- 説明：スピンドルが停止すると、順方向に始動します。 すでに稼働している場合は、スピンドルが稼働している方向に応じて回転数を増減します。

SPINDLE_FORWARD

- オプションの引数：開始 RPM（int）-デフォルト 100

spindle_stop

SPINDLE_REVERSE

- オプションの引数：開始 RPM（int）-デフォルト 100

SPINDLE_REVERSE_ADJUST

- オプションの引数：開始 RPM（int）-デフォルト 100
- 説明：スピンドルが停止すると、逆方向に始動します。 すでに稼働している場合は、スピンドルが稼働している方向に応じて回転数を増減します。

SPINDLE_FASTER

- 説明：スピンドル速度を 100RPM 増加させます

SPINDLE_SLOWER

- 説明：RPM が 100 になるまで、スピンドル速度を 100RPM 下げます。

SET_LINEAR_JOG_VELOCITY

- 必要な引数：インチ/分単位の速度（フロート）
- 説明：軸 0、1、2、6、7、8（X、Y、Z、U、V、W）のジョグ速度を設定します

SET_ANGULAR_JOG_VELOCITY

- 必要な引数：毎分度単位の速度（フロート）
- 説明：軸 3,4,5（A.B.C）のジョグ速度を設定します

CONTINUOUS_JOG

- 必要な引数：軸番号（int）、方向（int）

QUILL_UP

- オプションの引数：マシン Z 軸の絶対位置（フロート）
- 説明：Z 軸を指定された機械位置に移動します

FEED_HOLD

- 必須の引数：状態（ブール 0 または 1）

FEED_OVERRIDE

- 必要な引数：レート（フロート）

1. RAPID_OVERRIDE

- 必要な引数：rate（float 0-1）

SPINDLE_OVERRIDE

- 必要な引数：レート（フロート）

MAX_VELOCITY

- 必要な引数：レート（フロート）

reload_tooltable

OPTIONAL_STOP

- 必須の引数：状態（ブール 0 または 1）

BLOCK_DELETE

- 必須の引数：状態（ブール 0 または 1）

Abort

Pause

Resume

SINGLE_BLOCK

- 必須の引数：状態（ブール 0 または 1）

SMART_CYCLE_START

- 説明：アイドル状態の場合、一時停止すると 1 行実行すると、g コードプログラムが開始されます。

RE_START LINE

- 必須の引数：行番号（int）

MDI_AND_RETURN

- 必須の引数：Gcode コマンド
- 説明：現在のモードを記録し、コマンドを呼び出してからモードに戻ります。

MDI

- 必須の引数：Gcode コマンド
- 説明：モードを MDI に設定し、コマンドを呼び出します。

8.1.5 ハンドラーファイル拡張子

特別なファイルを使用して、コマンドとして使用できるカスタム Python コードを追加できます。 panelui_handler.py は Python で記述し、構成フォルダーに配置する必要があります。 panelui がそこでファイルを見つけると、使用可能なコマンドにその関数呼び出しを追加します。 hello_world と cycle_mode の 2 つの関数を追加するハンドラーファイルの例を次に示します。

```
# standard handler call - This will always be required
```

```
def get_handlers(linuxcnc_stat, linucnc_cmd, commands, master):
return [HandlerClass(linuxcnc_stat, linucnc_cmd, commands, master)]
# Also required - handler class
class HandlerClass:
# This will be pretty standard to gain access to everything
# linuxcnc_stat: is the python status instance of linuxcnc
# linuxcnc_cmd: is the python command instance of linuxcnc
# commands: is the command instance so one can call the internal routines
# master: give access to the master functions/data
def __init__(self, linuxcnc_stat, linuxcnc_cmd, commands, master):
self.parent = commands
self.current_mode = 0
# command functions are expected to have this layout:
# def some_name(self, widget_instance, arguments from widget):
# widget_instance gives access to the calling widget's function/data
# arguments can be a list of arguments, a single argument, or None
# depending on what was given in panelui's INI file.
def hello_world(self, wname, m):
# print to terminal so we know it worked
print '\nHello world\n'
print m # print the argument(s)
print wname.metadata # Print the calling widgets internal metadata (from config -
file)
# call a mdi command to print a msg in linuxcnc
# This requires linuxcnc to be homed, but does not check for that.
# parent commands expect a widget_instance - None is substituted
self.parent.mdi(None,'(MSG, Hello Linuxcnc World!)')
# Each call to this function will cycle the mode of linuxcnc
def cycle_mode(self, wname, m):
if self.current_mode == 0:
self.current_mode = 1
self.parent.set_mdi_mode()
elif self.current_mode == 1:
self.current_mode = 2
self.parent.set_auto_mode()
else:
self.current_mode = 0
self.parent.set_manual_mode()
print self.current_mode
# Boiler code, often required
```

```
def __getitem__(self, item):
return getattr(self, item)
def __setitem__(self, item, value):
return setattr(self, item, value)
```

8.2 HAL ユーザーインターフェース

8.2.1 序章

Halui は LinuxCNC 用の HAL ベースのユーザーインターフェイスであり、HAL ピンを NML コマンドに接続します。 一般的な GUI（軸、gscreen など）によって提供される機能（ボタン、インジケーターなど）のほとんどは、Halui の HAL ピンによって提供されます。

halui を追加する最も簡単な方法は、ini ファイルの[HAL]セクションに以下を追加することです。

```
[HAL]
HALUI = halui
...
```

それを呼び出す別の方法は、.hal ファイルに以下を含めることです。 ini ファイルへの正しいパスを使用していることを確認してください。

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

8.2.2 MDI

ユーザーは、HAL ピンのアクティブ化によって実行されるより複雑なタスクを追加したい場合があります。これは、次の MDI コマンドメソッドを使用して可能です。

MDI_COMMAND は、[HALUI]セクションの ini ファイルに追加されます。例：

[HALUI]

```
MDI_COMMAND = G0 G53 Z0
MDI_COMMAND = G28
MDI_COMMAND = o<mysub>call
...
```

halui が起動すると、ini の MDI_COMMAND フィールドが読み取られ、ピンが 00 から ini で見つかった MDI_COMMAND の数まで最大 64 コマンドまでエクスポートされます。 これらのピンは、他の hal ピンと同じように接続できます。 一般的な方法は、仮想コントロールパネルによって提供されるボタンを使用することです。 ハーフィル接続の例：

```
net quill-up <= pyvcp.quillup
net quill-up => halui.mdi-command-00
net reference-pos <= pyvcp.referencepos
```

```
net reference-pos => halui.mdi-command-01
net call-mysub <= pyvcp.callmysub
net call-mysub => halui.mdi-command-02
```

これらのネットは、halui が提供する halui.mdi-command-NN ピンを接続します。

```
$ halcmd show pin halui.mdi
Component Pins:
Owner Type Dir Value Name
10 bit IN FALSE halui.mdi-command-00 <== quill-up
10 bit IN FALSE halui.mdi-command-01 <== reference-pos
10 bit IN FALSE halui.mdi-command-02 <== call-mysub
...
```

halui MDI ピンが true に設定（パルス）されると、halui は ini で定義された MDI コマンドを送信します。 これは、現在の動作モードによっては常に成功するとは限りません（たとえば、AUTO の場合、halui は MDI コマンドを正常に送信できません）。

8.2.3 Halui ピンリファレンス

すべての halui ピンは、halui の man ページに記載されています。

```
$ man halui
```

8.2.4 構成例

sim config の例（configs / sim / axis / halui_pyvcp / halui.ini）がディストリビューションに含まれています。

8.3 Halui の例

Halui の例を機能させるには、ini ファイルの[HAL]セクションに次の行を追加する必要があります。

```
HALUI = halui
```

8.3.1 リモートスタート

リモートプログラムのスタートボタンを LinuxCNC に接続するには、halui.program.run ピンと halui.mode.auto ピンを使用します。 halui.mode.is-auto ピンを使用して、最初に実行しても問題がないことを確認する必要があります。 これは、and2 コンポーネントを使用して行います。 次の図は、これがどのように行われるかを示しています。 リモート実行ボタンを押すと、halui.mode.auto と and2.0.in0 の両方に接続されます。 自動モードで問題がない場合、ピン halui.mode.is-auto がオンになります。 and2.0 コンポーネントへの両方の入力がオンの場合、and2.0.out がオンになり、プログラムが開始されます。

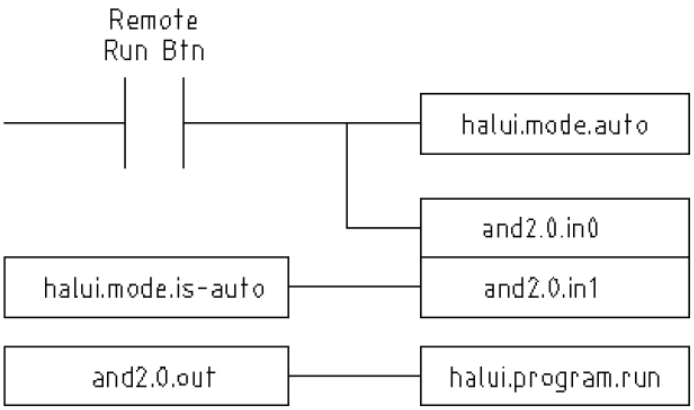


図 10-74

上記を実行するために必要な hal コマンドは次のとおりです。

```
net program-start-btn halui.mode.auto and2.0.in0 <= <your input pin>
net program-run-ok and2.0.in1 <= halui.mode.is-auto
net remote-program-run halui.program.run <= and2.0.out
```

1 行目には、リーダーピンが 2 つあることに注意してください。これは、次のように 2 行に分割することもできます。

```
net program-start-btn halui.mode.auto <= <your input pin>
net program-start-btn and2.0.in0
```

8.3.2 一時停止と再開

この例は、LinuxCNC が外部マシンからの信号で回転軸を移動できるようにするために開発されました。 2 つのシステム間の調整は、2 つの Halui コンポーネントによって提供されます。

- halui.program.is-paused
- halui.program.resume

カスタマイズした hal ファイルに、I / O に接続する次の 2 行を追加して、プログラムの一時停止をオンにするか、外部システムが LinuxCNC の続行を希望するときに再開します。

```
net ispaused halui.program.is paused => "your output pin"
net resume halui.program.resume <= "your input pin"
```

入力ピンと出力ピンは、他のコントローラーに配線されているピンに接続されています。 それらは、パラレルポートピンまたはアクセス可能なその他の I / O ピンである可能性があります。

このシステムは次のように機能します。 G コードで M0 が検出されると、halui.program.is-paused 信号が真になります。 これにより、出力ピンがオンになり、外部コントローラーは LinuxCNC が一時停止していることを認識します。

LinuxCNC gcode プログラムを再開するには、外部コントローラーの準備ができると、出力が true になります。 これにより、LinuxCNC に Gcode の実行を再開する必要があることが通知されます。

タイミングの難しさ

- 「再開」入力リターン信号は、g コードを再度実行するのに必要な時間より長くてはなりません。
- 「一時停止」出力は、「再開」信号が終了するまでにアクティブではなくなります。

これらのタイミングの問題は、ClassicLadder を使用して、単安定タイマーを介して「一時停止」出力をアクティブにし、1 つの狭い出力パルスを配信することで回避できます。「再開」パルスは、単安定タイマーを介して受信することもできます。

8.4 Python インターフェース

このドキュメントでは、LinuxCNC と通信するための PythonAPI を提供する linuxcncpython モジュールについて説明します。

8.4.1 linuxcncPython モジュール

ユーザーインターフェイスは、LinuxCNC タスクコントローラーに NML メッセージを送信することによって LinuxCNC アクティビティを制御し、LinuxCNC ステータス構造とエラー報告チャネルを監視することによって結果を監視します。

NML へのプログラムによるアクセスは、C ++ API を介して行われます。 ただし、LinuxCNC への NML インターフェイスの最も重要な部分は、linuxcnc モジュールを介して Python プログラムでも利用できます。

コマンド、ステータス、およびエラーチャネルへの NML インターフェイスの他に、linuxcnc モジュールには次のものも含まれています。

- ini ファイルからの値の読み取りのサポート

8.4.2 LinuxCNCNML インターフェースの使用パターン

linuxcnc の使用法の一般的なパターンは、おおまかに次のようになります。

- linuxcnc モジュールをインポートします
- 必要に応じて、コマンド、ステータス、およびエラー NML チャネルへの接続を確立します
- 定期的または必要に応じて、ステータスチャネルをポーリングします
- コマンドを送信する前に、ステータスから、実際に送信しても問題がないかどうかを判断します（たとえば、タスクが ESTOP 状態の場合、またはインタープリターがアイドル状態でない場合、実行コマンドを送信しても意味がありません）。
- linuxcnc コマンドチャネルメソッドの 1 つを使用してコマンドを送信します

エラーチャネルからメッセージを取得するには、エラーチャネルを定期的にポーリングし、取得したメッセージを処理します。

- 定期的または必要に応じて、ステータスチャネルをポーリングします
- エラーメッセージを出力し、例外コードを調べます

linuxcnc は、エラー報告をサポートするためにエラー Python 例外タイプも定義します。

8.4.3 LinuxCNC ステータスの読み取り

これは、いくつかの 80 以上の値を含む linuxcnc.stat オブジェクトのコンテンツを探索するための Python フラグメントです（通常の値に対して linuxcnc が実行されている間に実行されます）。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys
import linuxcnc
try:
s = linuxcnc.stat() # create a connection to the status channel
s.poll() # get current values
except linuxcnc.error, detail:
print "error", detail
sys.exit(1)
for x in dir(s):
if not x.startswith("_"):
print x, getattr(s,x)
```

Linuxcnc は、オーバーライドされない限り、NML 構成ファイルへのデフォルトのコンパイル済みパスを使用します。例については、ini ファイル値の読み取りを参照してください。

1.1.1.1 linuxcnc.stat 属性

acceleration

（float を返します）-デフォルトのアクセラレーション。ini エントリ[TRAJ] DEFAULT_ACCELERATION を反映します。

active_queue

（整数を返します）-ブレンドするモーションの数。

actual_position

（フロートのタプルを返します）-マシン単位での現在の軌道位置（x y z a b c u v w）。

adaptive_feed_enabled

（ブール値を返します）-適応送り速度オーバーライドのステータス（0/1）。

ain

（フロートのタプルを返します）-アナログ入力ピンの現在の値。

angular_units

（float を返します）-度あたりの機械角度単位。[TRAJ] ANGULAR_UNITSini 値を反映します。

aout

（フロートのタプルを返します）-アナログ出力ピンの現在の値。

axes

（整数を返します）-軸の数。 [TRAJ] COORDINATESini 値から派生。

axis

（dict のタプルを返します）-現在の軸の値を反映します。 軸辞書を参照してください。

axis_mask

（整数を返します）-ini ファイルの[TRAJ] COORDINATES で定義されている使用可能な軸のマスク。軸 X = 1、Y = 2、Z = 4、A = 8、B = 16、C = 32、U = 64、V = 128、W = 256 の合計を返します。

block_delete

（ブール値を返します）-ブロックは現在のステータスを削除します。

command

（文字列を返します）-現在実行中のコマンド。

current_line

（整数を返します）-現在実行中の行、int。

current_vel

（float を返します）-1 秒あたりのユーザー単位での現在の速度。

cycle_time

（float を返します）-スレッド期間

debug

（整数を返します）-ini ファイルからのデバッグフラグ。

delay_left

（float を返します）-ドウェル（G4）コマンドの残り時間、秒。

din

（整数のタプルを返します）-デジタル入力ピンの現在の値。

distance_to_go

（フロートを返します）-軌道プランナーによって報告された、現在の移動の残りの距離。

dout

（整数のタプルを返します）-デジタル出力ピンの現在の値。

dtg

（フロートのタプルを返します）-軌道プランナーによって報告された、各軸の現在の移動の残りの距離

echo_serial_number

（整数を返します）-UI からタスクに送信された最後に完了したコマンドのシリアル番号。 すべてのコマンドにはシリアル番号が付いています。 コマンドが実行されると、そのシリアル番号は echo_serial_number に反映されます。

enabled

（ブール値を返します）-軌道プランナー有効フラグ。

estop

（整数を返します）-STATE_ESTOP を返すかどうか。

exec_state

（整数を返します）-タスクの実行状態。 EXEC_ERROR、EXEC_DONE、EXEC_WAITING_FOR_MOTION、EXEC_WAITING_FOR_MOTION_QUEUE、EXEC_WAITING_FOR_IO、EXEC_WAITING_FOR_MOTION_AND_IO、EXEC_WAITING_FOR_DELAY、EXEC_WAITING_FOR_SYSTEM_CMD、XEC_WAITING_FOR_SPIND のいずれか

feed_hold_enabled

（ブール値を返します）-フィード保留のフラグを有効にします。

feed_override_enabled

（ブール値を返します）-フィードオーバーライドのフラグを有効にします。

feedrate

（float を返します）-現在の送り速度オーバーライド、1.0 = 100%。

file

（文字列を返します）-現在ロードされている gcode ファイル名とパス。

flood

（整数を返します）-フラッドステータス、FLOOD_OFF または FLOOD_ON のいずれか

g5x_index

（整数を返します）-現在アクティブな座標系、G54 = 1、G55 = 2 など。

g5x_offset

（float のタプルを返します）-現在アクティブな座標系のオフセット。

g92_offset

（フロートのタプルを返します）-現在の g92 オフセットのポーズ。

gcodes

（整数のタプルを返します）-各モーダルグループのアクティブな G コード。 G コード定数 G_0、G_1、G_2、G_3、G_4、G_5、G_5_1、G_5_2、G_5_3、G_7、G_8、G_100、G_17、G_17_1、G_18、G_18_1、G_19、G_19_1、G_20、G_21、G_28、G_28_1、G 、G_33、G_33_1、G_38_2、G_38_3、G_38_4、G_38_5、G_40、G_41、G_41_1、G_42、G_42_1、G_43、G_43_1、G_43_2、G_49、G_50、G_51、G_53、G_54、G_55、G_56 、G_59_2、G_59_3、G_61、G_61_1、G_64、G_73、G_76、G_80、G_81、G_82、G_83、G_84、G_85、G_86、G_87、G_88、G_89、G_90、G_90_1、G_91、G_91_1、G_92、G_92 、G_93、G_94、G_95、G_96、G_97、G_98、G_99

homed

（整数のタプルを返します）-現在ホームになっているジョイント、0 =ホームになっていない、1 =ホームになっている。

id

（整数を返します）-現在実行中のモーション ID。

inpos

（ブール値を返します）-machine-in-position フラグ。

input_timeout

（ブール値を返します）-進行中の M66 タイマーのフラグ。

interp_state

（整数を返します）-RS274NGC インタープリターの現在の状態。 INTERP_IDLE、INTERP_READING、INTERP_PAUSED、INTERP_WAITING のいずれか。

interpreter_errcode

（整数を返します）-現在の RS274NGC インタープリターの戻りコード。 INTERP_OK、INTERP_EXIT、INTERP_EXECUTE_FINISH、INTERP_ENDFILE、INTERP_FILE_NOT_OPEN、INTERP_ERROR のいずれか。 src / emc / nml_intf /interp_return.hh を参照してください

joint

（dict のタプルを返します）-現在のジョイント値を反映します。 共同辞書を参照してください。

joint_actual_position

（フロートのタプルを返します）-実際のジョイント位置。

joint_position

（フロートのタプルを返します）-必要なジョイント位置。

joints

（整数を返します）-ジョイントの数。 [KINS] JOINTSini 値を反映します。

kinematics_type

kinematics_type （整数を返します）-キネマティクスのタイプ。 の一つ：

- KINEMATICS_IDENTITY
- KINEMATICS_FORWARD_ONLY
- KINEMATICS_INVERSE_ONLY
- KINEMATICS_BOTH

limit

（整数のタプルを返します）-軸制限マスク。 minHardLimit = 1、maxHardLimit = 2、minSoftLimit = 4、maxSoftLimit = 8。

linear_units

（float を返します）-mm あたりの機械線形単位、[TRAJ] LINEAR_UNITSini 値を反映します。

lube

潤滑油 （整数を返します）-フラグの潤滑油。

lube_level

（整数を返します）-iocontrol.0.lube_level を反映します。

max_acceleration

float を返します）-最大加速度。 [TRAJ] MAX_ACCELERATION を反映します

max_velocity

（float を返します）-最大速度。 [TRAJ] MAX_VELOCITY を反映します。

mcodes

（10 個の整数のタプルを返します）-現在アクティブな M コード。

mist

（整数を返します）-ミストステータス、MIST_OFF または MIST_ON のいずれか

motion_line

（整数を返します）-ソース行番号のモーションは現在実行中です。 id との関係は不明です。

motion_mode

（整数を返します）-これはモーションコントローラのモードです。

TRAJ_MODE_COORD、TRAJ_MODE_FREE、TRAJ_MODE_TELEOP のいずれか。

motion_type

（整数を返します）-現在実行中のモーションのタイプ。 の一つ：

- MOTION_TYPE_TRAVERSE
- MOTION_TYPE_FEED
- MOTION_TYPE_ARC
- MOTION_TYPE_TOOLCHANGE
- MOTION_TYPE_PROBING
- MOTION_TYPE_INDEXROTARY
- Or 0 if no motion is currently taking place.

optional_stop

（整数を返します）-オプション停止フラグ

paused

（ブール値を返します）-モーション一時停止フラグ。

pocket_prepped

（整数を返します）-Tx コマンドが完了し、このポケットが準備されます。 -準備されたポケットがない場合は-1。

poll()

（組み込み関数）現在のステータス属性を更新するメソッド。

position

（フロートのタプルを返します）-軌道位置

probe_tripped

（ブール値を返します）-フラグ、プローブがトリップした場合は True（ラッチ）

probe_val

（整数を返します）-motion.probe 入力ピンの値を反映します。

probed_position

（フロートのタプルを返します）-プローブがトリップした位置。

probing

（ブール値を返します）-フラグ、プローブ操作が進行中の場合は True。

program_units

（整数を返します）-CANON_UNITS_INCHES = 1、CANON_UNITS_MM = 2、CANON_UNITS_CM = 3 のいずれか

queue

（整数を返します）-軌道プランナーキューの現在のサイズ。

queue_full

（ブール値を返します）-軌道プランナーキューがいっぱいです。

rapidrate

（float を返します）-高速オーバーライドスケール。

read_line

（整数を返します）-RS274NGC インタープリターが現在読み取っている行。

rotation_xy

（フロートを返します）-Z 軸を中心とした現在の XY 回転角。

settings

（float のタプルを返します）-現在のインタープリター設定。 settings [0] =シーケンス番号、settings [1] =送り速度、settings [2] =速度。

spindle

（dict のタプルを返します）’-現在のスピンドルステータスを返します。<sec：the-spindle-dictionary、スピンドル辞書>>を参照してください。

spindles

（整数を返します）-スピンドルの数。 [TRAJ] SPINDLESini 値を反映します。

state

（整数を返します）-現在のコマンド実行ステータス。 RCS_DONE、RCS_EXEC、RCS_ERROR のいずれか。

task_mode

（整数を返します）-現在のタスクモード。 MODE_MDI、MODE_AUTO、MODE_MANUAL のいずれか。

task_paused

（整数を返します）-タスク一時停止フラグ。

task_state

（整数を返します）-現在のタスクの状態。

STATE_ESTOP、STATE_ESTOP_RESET、STATE_ON、STATE_OFF のいずれか。

tool_in_spindle

（整数を返します）-現在のツール番号。

tool_offset

（float のタプルを返します）-現在のツールのオフセット値。

tool_table

（tool_results のタプルを返します）-ツールエントリのリスト。 各エントリは、id、xoffset、yoffset、zoffset、aoffset、boffset、coffset、uoffset、voffset、woffset、diameter、frontangle、backangle、orientation の一連のフィールドです。 id と orientation は整数で、残りは float です。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import linuxcnc
s = linuxcnc.stat()
s.poll()
# to find the loaded tool information it is in tool table index 0
if s.tool_table[0].id != 0: # a tool is loaded
print s.tool_table[0].zoffset
else:
print "no tool loaded"
```

velocity

（float を返します）-このプロパティは定義されていますが、有用な解釈がありません。

8.4.3.1 axis 辞書

軸の構成とステータスの値は、軸ごとの辞書のリストから入手できます。 特定の軸の属性にアクセスする方法の例を次に示します。

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

```
import linuxcnc
s = linuxcnc.stat()
s.poll()
print "Joint 1 homed: ", s.joint[1]["homed"]
```

以前は軸ディクショナリにあった多くのプロパティが現在はジョイントディクショナリにあることに注意してください。これは、重要なキネマティクスマシンでは、これらのアイテム（バックラッシュなど）が軸のプロパティではないためです。

max_position_limit

（float を返します）-マシン units.configuration パラメーターでの軸モーションの最大制限（ソフト制限）は、[JOINT_n] MAX_LIMIT を反映します。

min_position_limit

（float を返します）-マシン units.configuration パラメーターでの軸モーションの最小制限（ソフト制限）は、[JOINT_n] MIN_LIMIT を反映します。

velocity

（float を返します）-現在の速度。

8.4.3.2 joint 辞書

ジョイントごとに、次の辞書キーを使用できます。

backlash

（フロートを返します）-マシンユニットのバックラッシュ。 構成パラメーター。[JOINT_n] BACKLASH を反映します。

enabled

（整数を返します）-ゼロ以外は有効であることを意味します。

fault

（整数を返します）-ゼロ以外は、軸アンプの障害を意味します。

ferror_current

float を返します）-現在の次のエラー。

ferror_highmark

（float を返します）-エラー後の最大の大きさ。

homed

（整数を返します）-ゼロ以外の意味はホームになっています。

homing

（整数を返します）-ゼロ以外は、ホーミングが進行中であることを意味します。

inpos

（整数を返します）-ゼロ以外は位置にあることを意味します。

input

（float を返します）-現在の入力位置。

jointType

（整数を返します）-軸構成パラメーターのタイプ。[JOINT_n] TYPE を反映します。 LINEAR = 1、ANGULAR = 2。 詳細については、Jointini 構成を参照してください。

max_ferror

（float を返します）-最大次のエラー。 構成パラメーターは、[JOINT_n] FERROR を反映します。

max_hard_limit

（整数を返します）-ゼロ以外は、最大ハード制限を超えたことを意味します。

max_position_limit

（浮動小数点を返します）-機械単位での関節運動の最大制限（ソフト制限）。 構成パラメーター。[JOINT_n] MAX_LIMIT を反映します。

max_soft_limit

ゼロ以外は、max_position_limit を超えたことを意味します、int

min_ferror

（float を返します）-構成パラメーター。[JOINT_n] MIN_FERROR を反映します。

min_hard_limit

（整数を返します）-ゼロ以外は、最小ハード制限を超えたことを意味します。

min_position_limit

（浮動小数点を返します）-機械単位での関節運動の最小制限（ソフト制限）。 構成パラメーター。[JOINT_n] MIN_LIMIT を反映します。

min_soft_limit

（整数を返します）-ゼロ以外は、min_position_limit を超えたことを意味します

output

（float を返します）-コマンドされた出力位置。

override_limits

（整数を返します）-ゼロ以外は、制限がオーバーライドされることを意味します。

units

（フロートを返します）-mm あたり、または角度ジョイントの場合は 1 度あたりのジョイント単位。（ジョイントユニットは、構成パラメーター[JOINT_n] UNITS で特に設定されていない限り、マシンユニットと同じです）

velocity

（float を返します）-現在の速度。

8.4.4 スピンドル辞書

brake

（整数を返します）-スピンドルブレーキフラグの値。

direction

（整数を返します）-スピンドルの回転方向。 フォワード= 1、リバース= -1。

enabled

（整数を返します）-スピンドル有効フラグの値。

homed

（現在実装されていません）

increasing

（整数を返します）-不明。

orient_fault

（整数を返します）

orient_state

（整数を返します）

override

（フロートを返します）-スピンドル速度オーバーライドスケール。

override_enabled

（ブール値を返します）-スピンドルオーバーライド有効フラグの値。

speed

（フロートを返します）-スピンドル速度値、rpm、> 0：時計回り、<0：反時計回り。

8.4.5 コマンド送信の準備

一部のコマンドは、モードや状態に関係なく、常に送信できます。 たとえば、linuxcnc.command.abort（）メソッドはいつでも呼び出すことができます。

他のコマンドは適切な状態でのみ送信される可能性があり、それらのテストは少し注意が必要です。 たとえば、MDI コマンドは次の場合にのみ送信できます。

- ESTOP がトリガーされておらず、
- マシンの電源がオンになり、
- 軸がホームになり、
- インタプリタが実行されておらず、
- モードは MDI モードに設定されます

したがって、linuxcnc.command.mdi（）を介して MDI コマンドを送信する前の適切なテストは、次のようになります。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import linuxcnc
s = linuxcnc.stat()
c = linuxcnc.command()
def ok_for_mdi():
s.poll()
return not s.estop and s.enabled and (s.homed.count(1) == s.joints) and (s.interp_state  -
== linuxcnc.INTERP_IDLE)
if ok_for_mdi():
c.mode(linuxcnc.MODE_MDI)
c.wait_complete() # wait until mode switch executed
c.mdi("G0 X10 Y20 Z30")
```

8.4.6 linuxcnc.command を介したコマンドの送信

コマンドを送信する前に、次のようにコマンドチャネルを初期化します。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import linuxcnc
c = linuxcnc.command()
# Usage examples for some of the commands listed below:
c.abort()
c.auto(linuxcnc.AUTO_RUN, program_start_line)
c.auto(linuxcnc.AUTO_STEP)
```

```
c.auto(linuxcnc.AUTO_PAUSE)
c.auto(linuxcnc.AUTO_RESUME)
c.brake(linuxcnc.BRAKE_ENGAGE)
c.brake(linuxcnc.BRAKE_RELEASE)
c.flood(linuxcnc.FLOOD_ON)
c.flood(linuxcnc.FLOOD_OFF)
c.home(2)
c.jog(linuxcnc.JOG_STOP, jjogmode, joint_num_or_axis_index)
c.jog(linuxcnc.JOG_CONTINUOUS, jjogmode, joint_num_or_axis_index, velocity)
c.jog(linuxcnc.JOG_INCREMENT, jjogmode, joint_num_or_axis_index, velocity, increment)
c.load_tool_table()
c.maxvel(200.0)
c.mdi("G0 X10 Y20 Z30")
c.mist(linuxcnc.MIST_ON)
c.mist(linuxcnc.MIST_OFF)
c.mode(linuxcnc.MODE_MDI)
c.mode(linuxcnc.MODE_AUTO)
c.mode(linuxcnc.MODE_MANUAL)
c.override_limits()
c.program_open("foo.ngc")
c.reset_interpreter()
c.tool_offset(toolno, z_offset, x_offset, diameter, frontangle, backangle, orientation)
```

1.1.1.1 linuxcnc.command 属性

serial

現在のコマンドのシリアル番号

8.4.6.1 linuxcnc.command メソッド：

abort()

EMC_TASK_ABORT メッセージを送信します。

auto(int[, int])

プログラムを実行、ステップ、一時停止、または再開します。

brake(int)

スピンドルブレーキをオンまたはリリースします。

debug(int)

EMC_SET_DEBUG メッセージを介してデバッグレベルを設定します。

feedrate(float)

送り速度を設定します。

flood(int)

Syntax:

flood(command)

flood(linuxcnc.FLOOD_ON)

flood(linuxcnc.FLOOD_OFF)

Constants:

FLOOD_ON

FLOOD_OFF

home(int)

与えられた関節を家に帰します。

jog(command-constant, bool, int[, float[, float]])

構文：

jog（command、jjogmode、joint_num_or_axis_index、velocity [、distance]]）

jog（linuxcnc.JOG_STOP、jjogmode、joint_num_or_axis_index）

jog（linuxcnc.JOG_CONTINUOUS、jjogmode、joint_num_or_axis_index、velocity）

jog（linuxcnc.JOG_INCREMENT、jjogmode、joint_num_or_axis_index、velocity、distance）

コマンド定数

linuxcnc.JOG_STOP

linuxcnc.JOG_CONTINUOUS

linuxcnc.JOG_INCREMENT

jjogmode

True

個別のジョイントジョグを要求します（teleop_enable（0）が必要です）

Fales

要求軸デカルト座標ジョグ（teleop_enable（1）が必要）

Joint_num_or_axis_index

For joint jog (jjogmode=1)

joint_number

For axis Cartesian coordinate jog (jjogmode=0)

zero-based index of the axis coordinate with respect to the known coordinate letters XYZABCUVW (x=>0,y=>1,z=>2,a=>

load_tool_table()

ツールテーブルをリロードします。

maxvel(float)

最大速度を設定する

mdi(string)

MDI コマンドを送信します。 最大 255 文字。

mist(int)

turn on/off mist.

Syntax:

mist(command)

mist(linuxcnc.MIST_ON)

mist(linuxcnc.MIST_OFF)

Constants:

MIST_ON

MIST_OFF

mode(int)

モードを設定します（MODE_MDI、MODE_MANUAL、MODE_AUTO）。

override_limits()

オーバーライド軸制限フラグを設定します。

program_open(string)

NGC ファイルを開きます。

rapidrate()

set rapid override factor

reset_interpreter()

RS274NGC インタープリターをリセットする

set_adaptive_feed(int)

アダプティブフィードフラグを設定する

set_analog_output(int, float)

アナログ出力ピンを値に設定します

set_block_delete(int)

ブロック削除フラグを設定します

set_digital_output(int, int)

デジタル出力ピンを値に設定します

set_feed_hold(int)

フィードホールドのオン/オフを設定します

set_feed_override(int)

フィードオーバーライドのオン/オフを設定します

set_max_limit(int, float)

特定の軸の最大位置制限を設定します

set_min_limit()

特定の軸の最小位置制限を設定します

set_optional_stop(int)

オプションの停止のオン/オフを設定します

set_spindle_override(int [, int])

スピンドルオーバーライドを有効に設定します。 デフォルトはスピンドル 0 です。

spindle(int [[float] [int] [float,int]])

スピンドル方向を設定します。

SPINDLE_FORWARD、SPINDLE_REVERSE、SPINDLE_OFF、SPINDLE_INCREASE、SPINDLE_DECREASE、または SPINDLE_CONSTANT のいずれかの引数。

```
#!/usr/bin/env python
import linuxcnc
c = linuxcnc.command()
#Increase speed of spindle 0 by 100rpm. Spindle must be on first
c.spindle(linuxcnc.INCREASE)
#Increase speed of spindle 2 by 100rpm. Spindle must be on first
c.spindle(linuxcnc.SPINDLE_INCREASE, 2)
#Set speed of spindle 0 to 1024 rpm
c.spindle.(linuxcnc.SPINDLE_FORWARD, 1024)
#Set speed of spindle 1 to -666 rpm
```

```
c.spindle.(linuxcnc.SPINDLE_REVERSE, 666, 1)
#Stop spindle 0
c.spindle.(linuxcnc.SPINDLE_OFF)
#Stop spindle 0 explicitly
c.spindle.(linuxcnc.SPINDLE_OFF, 0)
```

spindleoverride(float [, int])

主軸オーバーライド係数を設定します。 デフォルトはスピンドル 0 です。

state(int)

マシンの状態を設定します。 マシンの状態は、
STATE_ESTOP、STATE_ESTOP_RESET、STATE_ON、または STATE_OFF である必要があります

task_plan_sync()

この呼び出しが完了すると、ディスク上の var ファイルがインタープリターからのライブ値で更新されます。

teleop_enable(int)

テロープモードを有効/無効にします（ジョイントジョギングを無効にします）。

tool_offset(int, float, float, float, float, float, int)

ツールオフセットを設定します。 上記の使用例を参照してください。

traj_mode(int)

軌道モードを設定します。 モードは、MODE_FREE、MODE_COORD、または MODE_TELEOP のいずれかです。

unhome(int)

特定の関節のホームを解除します。

wait_complete([float])

最後に送信されたコマンドの完了を待ちます。 秒単位のタイムアウトが指定されていない場合、デフォルトは 5 秒です。 タイムアウトした場合は-1 を返し、コマンドの実行状態に応じて RCS_DONE または RCS_ERROR を返します。

8.4.7 エラーチャネルの読み取り

エラーメッセージを処理するには、エラーチャネルに接続し、定期的にポーリング（）します。

エラーメッセージの NML チャネルには（コマンドチャネルとステータスチャネル以外の）キューがあることに注意してください。これは、エラーメッセージの最初のコンシューマーがそのメッセージをキューから

削除することを意味します。 別のエラーメッセージコンシューマー（Axis など）にメッセージが表示されるかどうかは、タイミングによって異なります。 セットアップには、エラーチャネルリーダータスクを 1 つだけ含めることをお勧めします。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import linuxcnc
e = linuxcnc.error_channel()
error = e.poll()
if error:
kind, text = error
if kind in (linuxcnc.NML_ERROR, linuxcnc.OPERATOR_ERROR):
typus = "error"
else:
typus = "info"
print typus, text
```

8.4.8 ini ファイル値の読み取り

linuxcnc.ini オブジェクトを介して ini ファイルから値を読み取る例を次に示します。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# run as:
# python ini-example.py ~/emc2-dev/configs/sim/axis/axis_mm.ini
import sys
import linuxcnc
inifile = linuxcnc.ini(sys.argv[1])
# inifile.find() returns None if the key wasnt found - the
# following idiom is useful for setting a default value:
machine_name = inifile.find("EMC", "MACHINE") or "unknown"
print "machine name: ", machine_name
# inifile.findall() returns a list of matches, or an empty list
# if the key wasnt found:
extensions = inifile.findall("FILTER", "PROGRAM_EXTENSION")
print "extensions: ", extensions
# override default NML file by ini parameter if given
nmlfile = inifile.find("EMC", "NML_FILE")
if nmlfile:
linuxcnc.nmlfile = os.path.join(os.path.dirname(sys.argv[1]), nmlfile)
```

8.4.9 linuxcnc.positionlogger タイプ

いくつかの使用上のヒントは、src / emc / usr_intf / gremlin /gremlin.py から収集できます。

1.1.1.1 members

npts

ポイントの数。

8.4.9.1 メソッド

start(float)

位置ロガーを開始し、ARG 秒ごとに実行します

clear()

ポジションロガーをクリアする

stop()

位置ロガーを停止します

call()

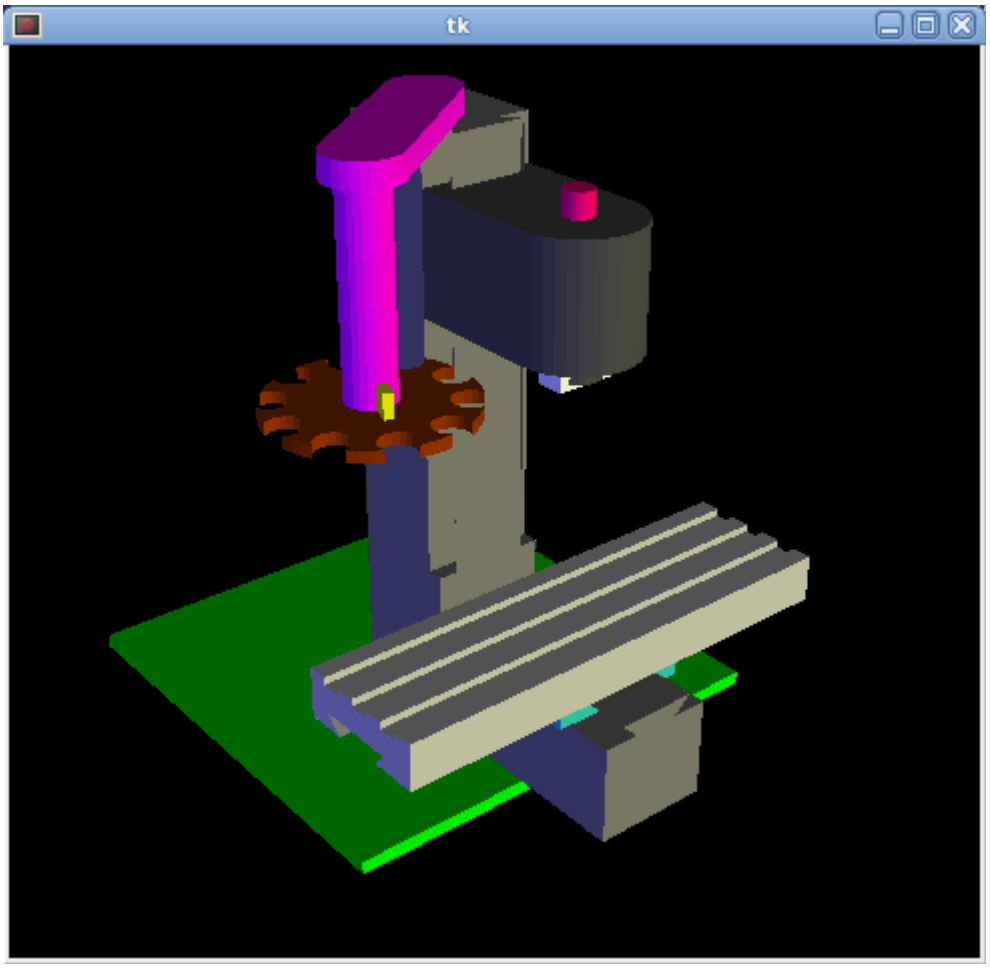
ここでバックプロットをプロットします。

last([int])

プロットの最新のポイントを返すか、なし、

8.5 Vismach

Vismach は、マシンのモデルを作成およびアニメーション化するために使用できる Python 関数のセットです。 Vismach はモデルを 3D ビューポートで表示し、関連する HAL ピンの値が変化するとモデルパーツがアニメーション化されます。



Vismach ビューポートビューは次のように操作できます

- スクロールホイールまたは右ボタンドラッグでズーム
- 左ボタンでパンドラッグ
- 中ボタンのドラッグまたは Shift キーを押しながらドラッグして回転します。

Vismach モデルは Python スクリプトの形式を取り、標準の Python 構文を使用できます。 これは、スクリプトをレイアウトする方法が複数あることを意味しますが、このドキュメントに記載されている例では、最も単純で最も基本的な方法を使用します。

Vismach モデルを作成する際の基本的な手順は次のとおりです。

- モーションを制御する HAL ピンを作成します
- パーツを作成する
- それらがどのように動くかを定義する
- 運動グループに集まる

8.5.1 スクリプトを開始します

＃！/ usr / bin / env python を含めて、ファイルをスクリプトとして実行できるようにすることは、テストに役立ちます。 最初に行うことは、必要なライブラリをインポートすることです。

```
#!/usr/bin/env python
from vismach import *
import hal
import math
import sys
```

8.5.2 HAL ピンを作成します。

Hal ピンは通常の Python「hal」ライブラリで作成され、Vismach に固有のものではありません。 詳細については、「Python での Userpace コンポーネントの作成」セクションを参照してください。 スクリプトファイル名と一致する名前でコンポーネントを作成してから、HAL ピンをそのコンポーネントに追加する必要があります。 これらは、Vismach モデルのアニメーション化に使用されるときに、コンポーネントハンドルと短い名前で参照されます。

```
c = hal.component("samplegui")
c.newpin("joint0", hal.HAL_FLOAT, hal.HAL_IN)
c.newpin("joint1", hal.HAL_FLOAT, hal.HAL_IN)
c.ready()
```

HAL ピン samplegui.joint0 および samplegui.joint1 を作成します。 loadusr -W samplegui を使用して Vismach モデルをロードすると、c.ready（）関数は loadusr に準備ができていることを通知します。

8.5.3 パーツの作成

CAD パッケージでジオメトリを作成し、AsciiSTL（）または AsciiOBJ（）関数を使用してモデルスクリプトにインポートするのがおそらく最も簡単です。 どちらの関数も、ファイル名または生データの 2 つの名前付き引数のいずれかを取ることができます

```
part = AsciiSTL(filename="path/to/file.stl)
part = AsciiSTL(data="solid part1 facet normal ....")
part = AsciiOBJ(filename="path/to/file.obj)
```

part = AsciiOBJ(data="v 0.123 0.234 0.345 1.0 ...")

パーツは、STL または OBJ スペースと同じ場所の Vismach スペースに作成されます。 これは、CAD パッケージでモデルを組み立てることができる可能性があることを意味します。

または、さまざまな形状プリミティブからモデルスクリプト内にパーツを作成することもできます。 多くの形状は原点で作成され、作成後に必要な場所に移動する必要があります。

```
cylinder = CylinderX(x1, r1, x2, r2)
cylinder = CylinderY(y1, r1, y2, r2)
cylinder = CylinderZ(z1, r1, z2, r2)
```

軸上の指定されたポイントに指定された半径を使用して、指定された軸上に（オプションでテーパーが付けられた）円柱を作成します。

```
sphere = Sphere(x, y, z, r)
```

（x、y、z）に半径 r の球を作成します

```
triangle = TriangleXY(x1, y1, x2, y2, x3, y3, z1, z2)
triangle = TriangleXZ(x1, z1, x2, z2, x3, z3, y1, y2)
triangle = TriangleYZ(y1, z1, y2, z2, y3, z3, x1, x2)
```

指定された平面に平行な最後の 2 つの値によって定義された平面間に三角形のプレートを作成し、頂点は 3 つの座標ペアによって指定されます。

```
arc = ArcX(x1, x2, r1, r2, a1, a2)
```

円弧形状を作成します。

```
box = Box(x1, y1, z1, x2, y2, z2)
```

指定された位置に反対側のコーナーがあり、XYZ 軸に平行なエッジを持つ直角プリズムを作成します。

```
box = BoxCentered(xw, yw, zw)
```

原点を中心とした xwby yw byzw ボックスを作成します。

```
box = BoxCenteredXY(xw, yw, z)
```

幅 xw / yw および高さ z のボックスを作成します。

複合パーツは、作成時またはその後にこれらのプリミティブを組み立てることによって作成できます。

```
part1 = Collection([Sphere(100,100,100,50), CylinderX(100,40,150,30)])
part2 = Box(50,40,75,100,75,100)
part3 = Collection([part2, TriangleXY(10,10,20,10,15,20,100,101)])
part4 = Collection([part1, part2])
```

8.5.4 可動部品

モデルを組み立てるには、Vismach スペースでパーツを移動する必要がある場合があります。 また、アニメーションの回転軸が原点で作成されるため、アニメーションを作成するために移動する必要がある場合もあります（ただし、パーツとともに移動します）。

```
part1 = Translate([part1], x, y, z)
```

part1 を x、y、z で指定された距離だけ移動します。

```
part1 = Rotate([part1], theta, x, y, z)
```

原点と x、y、z の間の軸を中心に角度シータでパーツを回転させます。

8.5.5 パーツのアニメーション

モデルをアニメーション化するために（HAL ピンの値によって制御されます）、HalTranslate と HalRotate の 2 つの関数があります。 パーツをアセンブリ内で移動するには、「コレクション」コマンドでアセンブリする前に、HAL モーションを定義する必要があります。 回転軸と平行移動ベクトルは、モデルの

組み立て中に vismach スクリプトによって移動されるとき、またはモデルがアニメーション化されるときに HAL ピンに応答して移動するときに、パーツとともに移動します。

```
part = HalTranslate([part], comp, "hal_pin", xs, ys, zs)
```

関数の引数は、最初はスクリプトの前半で事前に作成できるコレクション/パーツです。または、必要に応じてこの時点で作成することもできます。例：part1 = HalTranslate（[Box（。。。。）]、。。。）。 HAL コンポーネントは次の引数です。つまり、comp = hal.component（。。。）コマンドによって返されるオブジェクトです。 その後、モーションをアニメーション化する HAL の名前になります。これは、スクリプトの前半で作成された HAL コンポーネントの一部である既存の HAL ピンと一致する必要があります。

次に、X、Y、Z のスケールに従います。 1：1 スケールで作成されたデカルトマシンの場合、正の X 方向のモーションの場合、これは通常 1,0,0 になります。 ただし、STL ファイルがたまたま cm で、マシンがインチであった場合、0.3937（1cm /2.54in）をスケールとして使用することで、この時点でこれを修正できます。

```
part = HalRotate([part], comp, "hal_pin", angle_scale, x, y, z)
```

このコマンドの操作は HalTranslate と似ていますが、軸を定義するために最初にパーツを原点に移動する必要がある点が異なります。 回転軸は、原点から（x、y、z）で定義される点までです。 回転角度は度単位であるため、0 から 1 のスケーリングの回転ジョイントの場合、360 の角度スケールを使用する必要があります。パーツを原点から正しい位置に戻すと、回転軸は次のように見なされます。 パーツに「埋め込まれた」ままです。

8.5.6 モデルの組み立て。

パーツを一緒に移動するには、Collection（）コマンドを使用してパーツを組み立てる必要があります。パーツを組み立てて、正しい順序でそれらのモーションを定義することが重要です。 たとえば、回転スピンドルとアニメーション化されたドローバーを備えた移動ヘッドフライス盤を作成するには、次のようにします。

- ヘッド本体を作成します。
- 原点にスピンドルを作成します。
- 回転を定義します。
- ヘッドをスピンドルに移動するか、スピンドルをヘッドに移動します。
- ドローバーを作成する
- ドローバーの動きを定義する
- 3 つのパーツをヘッドアセンブリに組み立てます
- ヘッドアセンブリのモーションを定義します。

この例では、スピンドルの回転は、ドライブドッグのセットの回転によって示されます。

```
#Drive dogs
dogs = Box(-6,-3,94,6,3,100)
```

```
dogs = Color([1,1,1,1],[dogs])
dogs = HalRotate([dogs],c,"spindle",360,0,0,1)
dogs = Translate([dogs],-1,49,0)
#Drawbar
draw = CylinderZ(120,3,125,3)
draw = Color([1,0,.5,1],[draw])
draw = Translate([draw],-1,49,0)
draw = HalTranslate([draw],c,"drawbar",0,0,1)
# head/spindle
head = AsciiSTL(filename="./head.stl")
head = Color([0.3,0.3,0.3,1],[head])
head = Translate([head],0,0,4)
head = Collection([head, tool, dogs, draw])
head = HalTranslate([head],c,"Z",0,0,0.1)
# base
base = AsciiSTL(filename="./base.stl")
base = Color([0.5,0.5,0.5,1],[base])
# mount head on it
base = Collection([head, base])
```

最後に、すべての機械部品、床、および作業（存在する場合）の単一のコレクションを作成する必要があります。 シリアルマシンの場合、新しいパーツはそれぞれ前のパーツのコレクションに追加されます。 並列マシンの場合、いくつかの「ベース」パーツが存在する場合があります。 したがって、たとえば、scaragui.py では、link3 が link2 に、link2 が link1 に、link1 が link0 に追加されるため、最終的なモデルは次のように作成されます。

```
model = Collection([link0, floor, table])
```

一方、ベース上を移動する個別のパーツを備えた VMC モデルには

```
model = Collection([base, saddle, head, carousel])
```

8.5.7 その他の機能

```
part = Color([colorspec], [part])
```

パーツの表示色を設定します。 この場合、他の関数とは異なり、パーツ定義が 2 番目になることに注意してください。 colorspec は、3 つの RGB 値と不透明度で構成されます。 たとえば、不透明度 50%の赤の場合は[1,0,0,0.5]です。

```
myhud = Hud()
```

Vismach GUI にヘッドアップディスプレイを作成して、軸の位置などの項目を表示します。

```
part = Capture()
```

これが何をするのかわかりませんが、ツールチップの視覚化には重要なようです。

```
main(model, tooltip, work, size=10, hud=0, rotation_vectors=None, lat=0, lon=0)
```

これは、すべてを実現し、表示などを作成するコマンドです。「モデル」は、すべての機械部品を含むコレクションである必要があります。 「ツールチップ」と「作業」は、バックプロットでそれらの動きを視覚化するために Capture（）によって作成される必要があります。 ツールチップをツールに接続し、ツールをモデルに接続する方法の例については、scaragui.py を参照してください。

回転ベクトルまたは緯度/経度のいずれかを使用して元の視点を設定できます。デフォルトの初期視点はすぐ頭上からは役に立たないため、これを行うことをお勧めします。

サイズは、初期ビューで視覚化されるボリュームの範囲を設定します。 hud は、軸位置のヘッドアップディスプレイを指します。

8.5.8 Vismach スクリプトの基本構造。

```
#imports
from vismach import *
import hal
#create the HAL component and pins
comp = hal.component("compname")
comp.newpin("pin_name", hal.HAL_FLOAT, hal.HAL_IN)
...
#create the floor, tool and work
floor = Box(-50, -50, -3, 50, 50, 0)
work = Capture()
tooltip = Capture()
...
#Build and assemble the model
part1 = Collection([Box(-6,-3,94,6,3,100)])
part1 = Color([1,1,1,1],[part1])
part1 = HalRotate([part1],comp,"pin_name",360,0,0,1)
part1 = Translate([dogs],-1,49,0)
...
#create a top-level model
model = Collection([base, saddle, head, carousel])
#Start the visualization
main(model, tooltip, work, 100, lat=-75, lon=215)
```

9章 ドライバー

9.1 パラレルポートドライバ

hal_parport コンポーネントは、従来の PC パラレルポートのドライバです。 ポートには合計 17 個の物理ピンがあります。 元のパラレルポートは、これらのピンをデータ、制御、およびステータスの 3 つのグループに分割しました。 データグループは 8 つの出力ピンで構成され、制御グループは 4 つのピンで構成され、ステータスグループは 5 つの入力ピンで構成されます。

1990 年代初頭に、双方向パラレルポートが導入されました。これにより、データグループを出力または入力に使用できます。 HAL ドライバーは双方向ポートをサポートし、ユーザーがデータグループを入力または出力として設定できるようにします。 出力として構成されている場合、ポートは合計 12 の出力と 5 つの入力を提供します。 のように構成されている場合、4 つの出力と 13 の入力を提供します。

一部のパラレルポートでは、制御グループのピンはオープンコレクタであり、外部ゲートによってローに駆動される場合もあります。 オープンコレクタ制御ピンを備えたボード上。 x として構成されている場合、8 つの出力と 9 つの入力を提供します。

一部のパラレルポートでは、制御グループにプッシュプルドライバがあり、入力として使用できません。

HAL とオープンコレクター

HAL は、x モードの双方向ピンが実際にオープンコレクタ（OC）であるかどうかを自動的に判断できません。 そうでない場合、それらを入力として使用することはできず、外部ソースからそれらを LOW に駆動しようとすると、ハードウェアが損傷する可能性があります。

ポートのコレクタピンが開いているかどうかを確認するには、hal_parport を x モードでロードします。 デバイスが接続されていない場合、HAL はピンを TRUE として読み取る必要があります。 次に、制御ピンの 1 つから GND に 470 オームの抵抗を挿入します。 制御ピンに結果として生じる電圧が 0V に近く、HAL がピンを FALSE として読み取る場合、OC ポートがあります。 結果の電圧が 0V から遠く離れている場合、または HAL がピンを FALSE として読み取らない場合、ポートを x モードで使用することはできません。

制御ピンを駆動する外部ハードウェアも、オープンコレクタゲート（74LS05 など）を使用する必要があります。 一部のコンピューターでは、BIOS 設定が x モードを使用できるかどうかに影響する場合があります。 SPP モードが機能する可能性が最も高いです。

他の組み合わせはサポートされておらず、ドライバがインストールされると、ポートを入力から出力に変更することはできません。 パーポートドライバーは、最大 8 つのポート（hal_parport.c の MAX_PORTS で定義）を制御できます。 ポートにはゼロから始まる番号が付けられています。

9.1.1 読み込み中

hal_parport ドライバーはリアルタイムコンポーネントであるため、loadrt を使用してリアルタイムスレッドにロードする必要があります。 構成文字列は、使用するパラレルポートと（オプションで）それらのタイプを記述します。 構成文字列に少なくとも 1 つのポートが記述されていない場合は、エラーです。

```
loadrt hal_parport cfg="port [type] [port [type] ...]"
```

16 未満のポート番号の指定は、システムによって検出されたパラレルポートを指します。 これは、hal_parport ドライバーを構成する最も簡単な方法であり、ロードされている場合は Linuxparport_pc ドライバーと連携します。 0 のポートはシステムで検出された最初のパラレルポートであり、1 は次のポートです。

基本構成これは、Linux が検出した最初のパラレルポートを使用します。

```
loadrt hal_parport cfg="0"
```

ポートアドレスの使用代わりに、ポートアドレスは 16 進表記 0x、次にアドレスを使用して指定できます。

```
loadrt hal_parport cfg="0x378"
```

タイプ hal_parport ドライバによって処理されるパラレルポートごとに、オプションでタイプを指定できます。 タイプは、in、out、epp、または x のいずれかです。

Table 11.1: Parallel Port Direction

Pin	in	out/epp	x
1	out	out	in
2	in	out	out
3	in	out	out
4	in	out	out
5	in	out	out
6	in	out	out
7	in	out	out
8	in	out	out
9	in	out	out
10	in	in	in
11	in	in	in
12	in	in	in
13	in	in	in
14	out	out	in
15	in	in	in
16	out	out	in
17	out	out	in

タイプが指定されていない場合、デフォルトは out です。

epp のタイプは out と同じですが、hal_parport ドライバーは、ポートを EPP モードに切り替えるように要求します。 hal_parport ドライバーは EPP バスプロトコルを使用しませんが、一部のシステムでは、EPP モードは、一部の周辺ハードウェアの動作を改善する方法でポートの電気的特性を変更します。 Gecko G540 のチャージポンプは、一部のパラレルポートでこれを必要とすることが知られています。

モード x については上記の注を参照してください。

2 つのパラレルポートを使用した例これにより、システムで検出された 2 つのパラレルポートが有効になります。1 つ目は出力モード、2 つ目は入力モードです。

```
loadrt hal_parport cfg="0 out 1 in"
```

関数読み取りおよび書き込み関数を実行するように LinuxCNC に指示する必要もあります。

```
addf parport.0.read base-thread
addf parport.0.write base-thread
```

9.1.2 PCI ポートアドレス

1 つの優れた PCI パーポートカードは、Netmos9815 チップセットで作られています。 + 5V の信号が良好で、シングルポートまたはデュアルポートで供給できます。

PCI カードの I / O アドレスを見つけるには、ターミナルウィンドウを開き、listpci コマンドを使用します。

```
lspci -v
```

「Netmos」が含まれているエントリを探します。 2 ポートカードの例：

```
0000:01:0a.0 Communication controller: \
Netmos Technology PCI 9815 Multi-I/O Controller (rev 01)
Subsystem: LSI Logic / Symbios Logic 2POS (2 port parallel adapter)
Flags: medium devsel, IRQ 5
I/O ports at b800 [size=8]
I/O ports at bc00 [size=8]
I/O ports at c000 [size=8]
I/O ports at c400 [size=8]
I/O ports at c800 [size=8]
I/O ports at cc00 [size=16]
```

実験の結果、最初のポート（オンカードポート）はリストされている 3 番目のアドレス（c000）を使用し、2 番目のポート（リボンケーブルで接続されているポート）はリストされている最初のアドレス（b800）を使用していることがわかりました。 次の例は、デフォルトの出力方向を使用したオンボードパラレルポートと PCI パラレルポートを示しています。

```
loadrt hal_parport cfg="0x378 0xc000"
```

値が異なることに注意してください。 Netmos カードはプラグアンドプレイであり、どのスロットに挿入するかによって設定が変わる可能性があるため、「内部に潜り込んで」再配置したい場合は、開始する前にこれらの値を確認してください。 LinuxCNC。

9.1.3 ピン

- parport。<p> .pin- <n> -out（ビット）物理出力ピンを駆動します。
- parport。<p> .pin- <n> -in（ビット）物理入力ピンを追跡します。
- parport。<p> .pin- <n> -in-not（ビット）物理入力ピンを追跡しますが、反転します。

各ピンについて、<p>はポート番号、<n>は 25 ピン D シェルコネクタの物理ピン番号です。

物理出力ピンごとに、ドライバーは 1 つの HAL ピンを作成します（例：parport.0.pin-14-out）。

物理入力ピンごとに、ドライバーは 2 つの HAL ピンを作成します（例：parport.0.pin-12-in と parport.0.pin-12-in-not）。

-in HAL ピンは、物理ピンがハイの場合は TRUE であり、物理ピンがローの場合は FALSE です。 -in-not HAL ピンは反転され、物理ピンがハイの場合は FALSE になります。

9.1.4 パラメータ

- parport。<p> .pin- <n> -out-invert（ビット）出力ピンを反転します。
- parport。<p> .pin- <n> -out-reset（ビット）（出力ピンの場合のみ）-reset 関数の実行時にこのピンをリセットする必要がある場合は TRUE。
- parport。<p> .reset-time ’（U32）ピン間の時間（ナノ秒単位）は書き込みによって設定され、リセット機能の場合はリセット機能によってリセットされます。

-invert パラメータは、出力ピンがアクティブ High かアクティブ Low かを決定します。 -invert が FALSE の場合、HAL-out ピンを TRUE に設定すると、物理ピンが High になり、FALSE に設定すると Low になります。 -invert が TRUE の場合、HAL-out ピンを TRUE に設定すると、物理ピンがローに駆動されます。

9.1.5 機能

- parport。<p> .read（funct）ポート<portnum>の物理入力ピンを読み取り、HAL-in および-in-not ピンを更新します。
- parport.read-all（funct）すべてのポートの物理入力ピンを読み取り、HAL-in および-in-not ピンを更新します。
- parport。<p> .write（funct）ポート<p>の HAL 出力ピンを読み取り、そのポートの物理出力ピンを更新します。
- parport.write-all（funct）すべてのポートの HAL 出力ピンを読み取り、すべての物理出力ピンを更新します。
- parport。<p> .reset（funct）関連する書き込みからリセット時間が経過するまで待機し、ピンを-out-invert および-out-invert 設定で示される値にリセットします。 リセットは、後で書き込みと同じスレッドで行う必要があります。 ’-reset が TRUE の場合、リセット機能はピンを-out-invert の値に設定します。 これを stepgen の doublefreq と組み合わせて使用すると、期間ごとに 1 つのステップを生成できます。 doublefreq を有効にするには、そのピンの stepgen ステップスペースを 0 に設定する必要があります。

個々の機能は、1 つのポートを非常に高速なスレッドで更新する必要があるが、他のポートを低速のスレッドで更新して CPU 時間を節約できる場合に提供されます。 -all 関数と個々の関数の両方を同時に使用することはおそらく良い考えではありません。

9.1.6 一般的な問題

モジュールレポートをロードする場合

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

次に、標準カーネルモジュール parport_pc がロードされていないこと 1、およびシステム内の他のデバイスが I / O ポートを要求していないことを確認します。

モジュールがロードされても機能していないように見える場合は、ポートアドレスが正しくありません。

9.1.7 DoubleStep の使用

パラレルポートで DoubleStep を設定するには、parport.n.write の後に関数 parport.n.reset を追加し、ステップスペースを 0 に設定し、必要なリセット時間を設定する必要があります。 そのため、このステップは HAL のすべての期間でアサートされ、parport.n.reset-time で指定された時間アサートされた後、parport でオフに切り替えることができます。

例えば：

```
loadrt hal_parport cfg="0x378 out"
setp parport.0.reset-time 5000
loadrt stepgen step_type=0,0,0
addf parport.0.read base-thread
addf stepgen.make-pulses base-thread
addf parport.0.write base-thread
addf parport.0.reset base-thread
addf stepgen.capture-position servo-thread
...
setp stepgen.0.steplen 1
setp stepgen.0.stepspace 0
```

DoubleStep の詳細については、wiki を参照してください。

9.2 AX5214H ドライバー

9.3 GS2VFD ドライバー

9.4 MESAHOSTMOT2 ドライバー

9.5 MOTENC ドライバー

9.6 MB2HAL

9.7 OPTO22 ドライバー

9.8 ピコドライバー

9.9 プルート P ドライバー

9.10 POWERMAXMODBUS ドライバー

9.11 サーボトゥーゴードライバー

SERVO-TO-GO（STG）は、LINUXCNC でサポートされている最初の PC モーションコントロールカードの 1 つです。 これは ISA カードであり、さまざまなフレーバーで存在します（すべてこのドライバーでサポートされています）。 このボードには、最大 8 チャネルの直交エンコーダ入力、8 チャネルのアナログ入力および出力、32 ビットデジタル I / O、割り込み付きインターバルタイマー、および ウォッチドッグが含まれています。 詳細については、SERVO TOGO の WEB ページを参照してください。

NOTE

SERVO TO GO カードのオペアンプは、最新のスイッチモード DC-DC コンバーターを使用する新しい ATX 電源装置では機能しないという報告があります。 故障モードは、ドライバーが何をするように命令しているかに関係なく、STG カードが定電圧を出力することです。 リニア電圧レギュレータを備えた古い ATX 電源にはこの問題はなく、STG カードで正常に動作します。

9.11.1 インストール

9.11.2 PINS

9.11.3 パラメーター

9.11.4 関数

9.12 シャトル

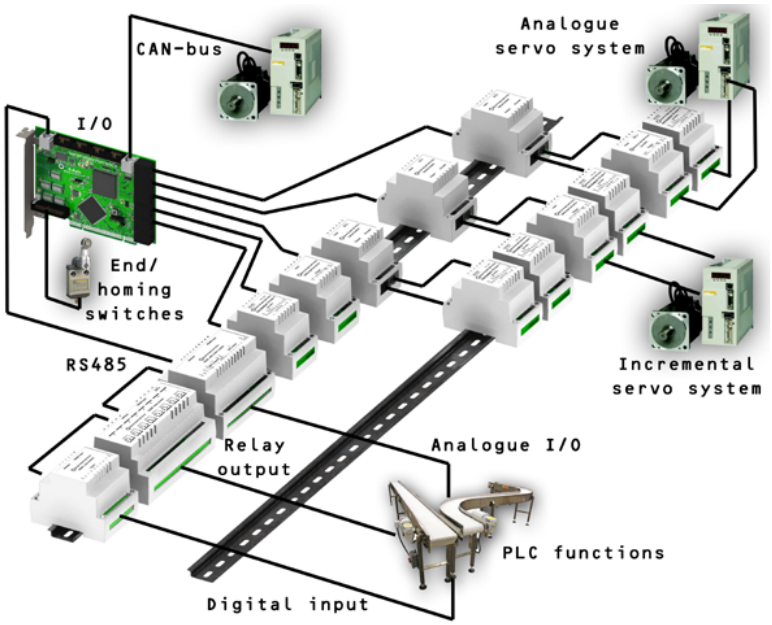
9.13 一般的なメカトロニクスドライバー

一般的なメカトロニクス GM6-PCI カードベースのモーションコントロールシステム

詳細な説明については、システム統合マニュアルを参照してください。

GM6-PCI モーションコントロールカードは、FPGA と PCI ブリッジインターフェイス ASIC に基づいています。 短時間のシステム統合手順で、小さな自動製造セルを制御できます。 次の図は、制御システムに関連するデバイスの一般的な接続を示しています。

- 最大 6 軸を制御でき、それぞれがステッパーまたは CAN バスインターフェースまたはアナログサーボになります。
- GPIO：4 回 8 つの I / O ピンが標準のフラットケーブルヘッダーに配置されます。
- RS485 I / O エキスパンダーモジュール：RS485 バスは、コンパクトな DIN レールに取り付けられたエキスパンダーモジュールとのインターフェース用に設計されました。 8 チャンネルデジタル入力、8 チャンネルリレー出力、およびアナログ I / O（4x +/- 10 ボルト出力および 8x +/- 5 ボルト入力）モジュールが利用可能になりました。 合計で最大 16 個のモジュールをバスに接続できます。
- 20 個の光学的に絶縁された入力ピン：2 つのエンドスイッチと各ジョイントに 1 つのホーミングセンサーを直接接続するための 3 つの 6 倍。 さらに、2 つの光学的に絶縁された非常停止入力。



インストール：

```
loadrt hal_gm
```

ロード中（またはロードの試行中）、ドライバーはいくつかの有用なデバッグメッセージをカーネルログに出力します。これは DMESG で表示できます。

1 つのシステムで最大 3 つのボードを使用できます。

GM6-PCI カードには次のコネクタがあります。

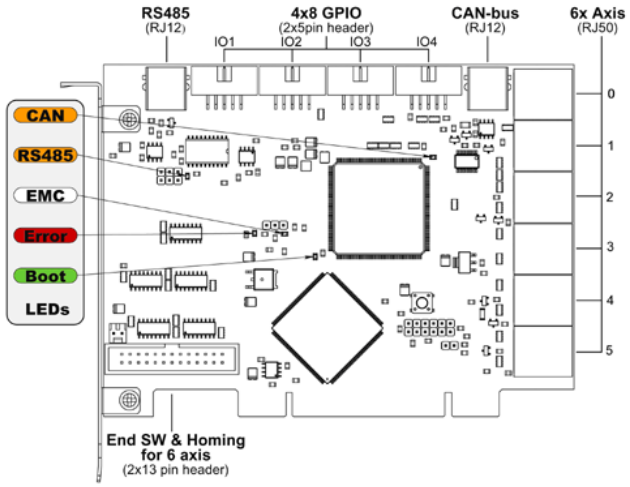


図 11-5

9.13.1 I / O コネクタ

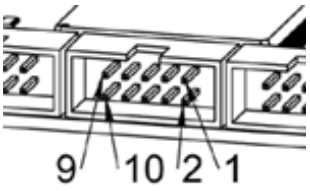


図 11-6

表 11.3：GPIO コネクタのピン配置

9	7	5	3	1
IOx/7	IOx/5	IOx/3	IOx/1	VCC

10	8	6	4	2
GND	IOx6	IOx4	IOx/2	IOx/0

各ピンは、デジタル入力または出力として構成できます。 GM6-PCI モーションコントロールカードには、4 つの汎用 I / O（GPIO）コネクタがあり、それぞれに 8 つの構成可能な I / O があります。 すべての GPIO ピンとパラメーター名は次のように始まります

```
gm.<nr. of card>.gpio.<nr of gpio con>
```

、ここで、<GPIO CON の NR>は 0 から 3 の形式です。例：

```
gm.0.gpio.0.in-0
```

GM6-PCI カードの最初の GPIO コネクタの最初のピンの状態を示します。 ハルピンは機能により更新されます

```
gm.<nr of card>.read
```

1.1.1.1 ピン

Table 11.4: GPIO pins

Pins	Type and direction	Pin description
.in-<0-7>	(bit, Out)	Input pin
.in-not-<0-7>	(bit, Out)	Negated input pin
.out-<0-7>	(bit, In)	Output pin. Used only when GPIO is set to output.

9.13.1.1 パラメーター

Table 11.5: GPIO parameters

Pins	Type and direction	Parameter description
.is-out-<0-7>	(bit, R/W)	When True, the corresponding GPIO is set to totem-pole output, other wise set to high impedance input.
.invert-out-<0-7>	(bit, R/W)	When True, pin value will be inverted. Used when pin is configured as output.

9.13.2 軸コネクタ

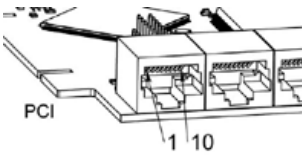


図 11-7

Table 11.6: Pinout of axis connectors

1	Encoder A
2	+5 Volt (PC)
3	Encoder B
4	Encoder Index
5	Fault
6	Power Enabled
7	Step/CCW/B
8	Direction/CW/A
9	Ground (PC)
10	DAC serial line

1.1.1.1 軸インターフェースモジュール

小型の DIN レールに取り付けられたインターフェースモジュールにより、さまざまなタイプのサーボモジュールを軸コネクタに簡単に接続できます。 一般的なアプリケーションを評価するために、7 つの異なるシステム構成がシステム統合マニュアルに示されています。 また、AXIS モジュールの詳細な説明は、システム統合マニュアルに記載されています。

次のブロック図に示すように、適切なサーボドライブ構造を評価するには、モジュールを接続する必要があります。

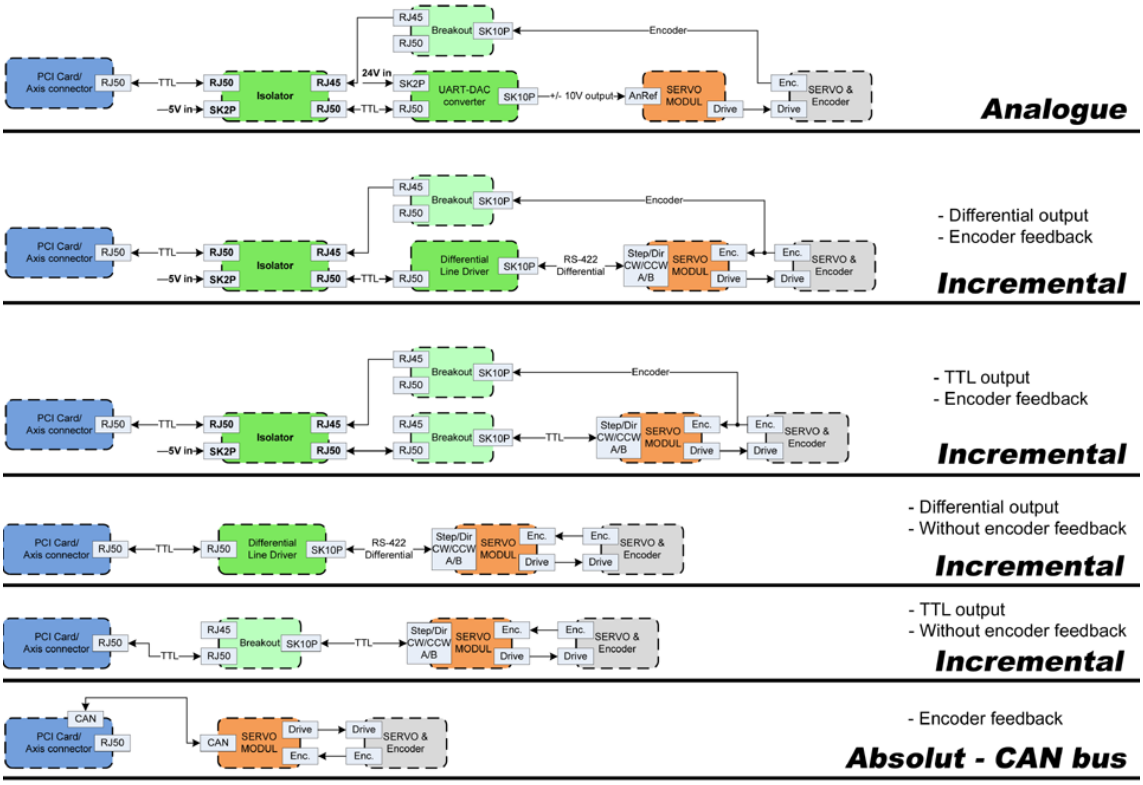


図 11-8

9.13.2.1 エンコーダー

GM6-PCI モーションコントロールカードには、6 つのエンコーダモジュールがあります。 各エンコーダモジュールには 3 つのチャネルがあります。

- CHANNEL-A
- CHANNEL-B
- CHANNEL-I（インデックス）

直交エンコーダ信号またはステップ/方向信号をカウントすることができます。 各エンコーダモジュールは、対応する RJ50 軸コネクタの入力に接続されています。

すべてのエンコーダピンとパラメータ名は次のように始まります。

```
gm.<nr. of card>.encoder.<nr of axis>
```

ここで、<軸の NR>は 0 から 5 の形式です。例：

```
gm.0.encoder.0.position
```

軸 0 のエンコーダモジュールの位置を指します。

GM6-PCI カードは、LINUXCNC とは独立してエンコーダ信号をカウントします。 ハルピンは機能によって更新されます：

```
gm.<nr of card>.read
```

Table 11.7: Encoder pins

Pins	Type and direction	Pin description
.reset	(bit, In)	When True, resets counts and position to zero.
.rawcounts	(s32, Out)	The raw count is the counts, but unaffected by reset or the index pulse.
.counts	(s32, Out)	Position in encoder counts.
.position	(float, Out)	Position in scaled units (=.counts/.position-scale).
.index-enabled	(bit, IO)	When True, counts and position are rounded or reset (depends on index-mode) on next rising edge of channel-I. Every time position is reset because of Index, index-enabled pin is set to 0 and remain 0 until connected hal pin does not set it.
.velocity	(float, Out)	Velocity in scaled units per second. GM encoder uses high frequency hardware timer to measure time between encoder pulses in order to calculate velocity. It greatly reduces quantization noise as compared to simply differentiating the position output. When the measured velocity is below min-speed-estimate, the velocity output is 0.

Table 11.8: Encoder parameters

Parameters	Type and Read/Write	Parameter description
.counter-mode	(bit, R/W)	When True, the counter counts each rising edge of the channel-A input to the direction determined by channel-B. This is useful for counting the output of a single channel (non-quadrature) or step/dir signal sensor. When false, it counts in quadrature mode.
.index-mode	(bit, R/W)	When True and .index-enabled is also true, .counts and .position are rounded (based on .counts-per-rev) at rising edge of channel-I. This is useful to correct few pulses error caused by noise. In round mode, it is essential to set .counts-per-rev parameter correctly. When .index-mode is False and .index-enabled is true, .counts and .position are reset at channel-I pulse.

Table 11.8: (continued)

Parameters	Type and Read/Write	Parameter description
.counts-per-rev	(s32, R/V)	Determine how many counts are between two index pulses. It is used only in round mode, so when both .index-enabled and .index-mode parameters are True. GM encoder process encoder signal in 4x mode, so for example in case of a 500 CPR encoder it should be set to 2000. This parameter can be easily measured by setting .index-enabled True and .index-mode False (so that .counts resets at channel-I pulse), than move axis by hand and see the maximum magnitude of .counts pin in halmeter.
.index-invert	(bit, R/W)	When True, channel-I event (reset or round) occur on falling edge of channel-I signal, otherwise on rising edge.
.min-speed-estimate	(float, R/W)	Determine the minimum measured velocity magnitude at which .velocity will be set as nonzero. Setting this parameter too low will cause it to take a long time for velocity to go to zero after encoder pulses have stopped arriving.
.position-scale	(float, R/W)	Scale in counts per length unit. .position=.counts/.position-scale. For example, if position-scale is 2000, then 1000 counts of the encoder will produce a position of 0.5 units.

HAL の例軸 0 のエンコーダモジュールを設定して、500 CPR 直交エンコーダ信号を受信し、リセットを使用して位置を丸めます。

```
setp gm.0.encoder.0.counter-mode 0 # 0: quad, 1: stepDir
setp gm.0.encoder.0.index-mode 1 # 0: reset pos at index, 1:round pos at index
setp gm.0.encoder.0.counts-per-rev 2000 # GM process encoder in 4x mode, 4x500=2000
setp gm.0.encoder.0.index-invert 0
setp gm.0.encoder.0.min-speed-estimate 0.1 # in position unit/s
setp gm.0.encoder.0.position-scale 20000 # 10 encoder rev cause the machine to
move one position unit (10x2000)
```

エンコーダ位置を LINUXCNC 位置フィードバックに接続します。

```
ネット XPOS-FBGM.0.ENCODER.0.POSITION => JOINT.0.MOTOR-POS-FB
```

9.13.2.2 STEPGEN モジュール

GM6-PCI モーションコントロールカードには、各関節に 1 つずつ、合計 6 つの STEPGEN モジュールがあります。 各モジュールには 2 つの出力信号があります。 ステップ/方向、アップ/ダウン、または直交（A / B）パルスを生成できます。 各 STEPGEN モジュールは、対応する RJ50 軸コネクタのピンに接続されます。

すべての STEPGEN ピンとパラメーター名は次のように始まります。

```
gm.<nr. of card>.stepgen.<nr of axis>
```

、ここで、軸の NR は 0 から 5 の形式です。例：

```
gm.0.stepgen.0.position-cmd
```

カード 0 の軸 0 の STEPGEN モジュールの位置指令を指します。

GM6-PCI カードは、LINUXCNC とは独立してステップパルスを生成します。 ハルピンは機能により更新されます

gm.<nr of card>.write

Table 11.9: Stepgen module pins

Pins	Type and direction	Pin description
.enable	(bit, In)	Stepgen produces pulses only when this pin is true.
.count-fb	(s32, Out)	Position feedback in counts unit.
.position-fb	(float, Out)	Position feedback in position unit.
.position-cmd	(float, In)	Commanded position in position units. Used in position mode only.
.velocity-cmd	(float, In)	Commanded velocity in position units per second. Used in velocity mode only.

Table 11.10: Stepgen module parameters

Parameters	Type and Read/Write	Parameter description
.step-type	(u32, R/W)	When 0, module produces Step/Dir signal. When 1, it produces Up/Down step signals. And when it is 2, it produces quadrature output signals.
.control-type	(bit, R/W)	When True, .velocity-cmd is used as reference and velocity control calculate pulse rate output. When False, .position-cmd is used as reference and position control calculate pulse rate output.
.invert-step1	(bit, R/W)	Invert the output of channel 1 (Step signal in StepDir mode)
.invert-step2	(bit, R/W)	Invert the output of channel 2 (Dir signal in StepDir mode)
.maxvel	(float, R/W)	Maximum velocity in position units per second. If it is set to 0.0, .maxvel parameter is ignored.
.maxaccel	(float, R/W)	Maximum acceleration in position units per second squared. If it is set to 0.0, .maxaccel parameter is ignored.
.position-scale	(float, R/W)	Scale in steps per length unit.
.steplen	(u32, R/W)	Length of step pulse in nano-seconds.
.stepspace	(u32, R/W)	Minimum time between two step pulses in nano-seconds.
.dirdelay	(u32, R/W)	Minimum time between step pulse and direction change in nano-seconds.

適切な値の評価については、以下のタイミング図を参照してください。

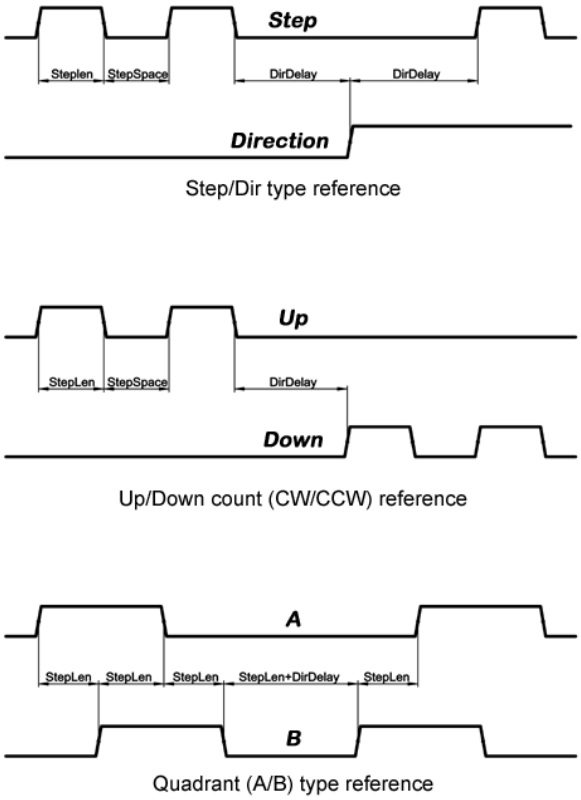


図 11-9

HAL の例位置単位ごとに 1000 ステップパルスを生成するように軸 0 の STEPGEN モジュールを設定します。

```
setp gm.0.stepgen.0.step-type 0 # 0:stepDir, 1:UpDown, 2:Quad
setp gm.0.stepgen.0.control-type 0 # 0:Pos. control, 1:Vel. Control
setp gm.0.stepgen.0.invert-step1 0
setp gm.0.stepgen.0.invert-step2 0
setp gm.0.stepgen.0.maxvel 0 # do not set maxvel for step
# generator, let interpolator control it.
setp gm.0.stepgen.0.maxaccel 0 # do not set max acceleration for
# step generator, let interpolator control it.
setp gm.0.stepgen.0.position-scale 1000 # 1000 step/position unit
setp gm.0.stepgen.0.steplen 1000 # 1000 ns = 1 us
setp gm.0.stepgen.0.stepspace1000 # 1000 ns = 1 us
setp gm.0.stepgen.0.dirdelay 2000 # 2000 ns = 2 us
```

STEPGEN を軸 0 の位置基準に接続し、ピンを有効にします。

```
net Xpos-cmd joint.0.motor-pos-cmd => gm.0.stepgen.0.position-cmd
net Xen joint.0.amp-enable-out => gm.0.stepgen.0.enable
```

9.13.2.3 イネーブル信号とフォールト信号

GM6-PCI モーションコントロールカードには、1 つのイネーブル出力と 1 つのフォールト入力 HAL ピンがあり、両方とも各 RJ50 軸コネクタと CAN コネクタに接続されています。

ハルピンは機能によって更新されます：

```
gm.<nr of card>.read
```

Table 11.11: Enable and Fault signal pins

Pins	Type and direction	Pin description
gm.<nr of card>.power-enable	(bit, In)	If this pin is True, * and Watch Dog Timer is not expired * and there is no power fault Then power enable pins of axis- and CAN connectors are set to high, otherwise set to low.
gm.<nr of card>.power-fault	(bit, Out)	Power fault input.

9.13.2.4 Axis DAC

GM6-PCI モーションコントロールカードには、各ジョイントに 1 つずつ、合計 6 つのシリアル軸 DAC ドライバモジュールがあります。 各モジュールは、対応する RJ50 軸コネクタのピンに接続されています。 すべての軸 DAC ピンとパラメータ名は次のように始まります。

```
gm.<nr. of card>.dac.<nr of axis>
```

、ここで、軸の NR は 0 から 5 の形式です。たとえば

```
gm.0.dac.0.value
```

軸 0 の DAC モジュールの出力電圧を指します。HAL ピンは機能によって更新されます。

```
gm.<nr of card>.write
```

Table 11.12: Axis DAC pins

Pins	Type and direction	Pin description
.enable	(bit, In)	Enable DAC output. When enable is false, DAC output is 0.0 V.
.value	(float, In)	Value of DAC output in Volts.

Table 11.13: Axis DAC parameters

Parameters	Type and direction	Parameter description
.offset	(float, R/W)	Offset is added to the value before the hardware is updated
.high-limit	(float, R/W)	Maximum output voltage of the hardware in volts.
.low-limit	(float, R/W)	Minimum output voltage of the hardware in volts.
.invert-serial	(float, R/W)	GM6-PCI card is communicating with DAC hardware via fast serial communication to highly reduce time delay compared to PWM. DAC module is recommended to be isolated which is negating serial communication line. In case of isolation, leave this parameter to default (0), while in case of none-isolation, set this parameter to 1.

9.13.3 CAN バスサーボアンプ

GM6-PCI モーションコントロールカードには、CAN サーボアンプを駆動するための CAN モジュールがあります。 CANOPEN のような高レベルのプロトコルの実装はさらなる開発です。 現在、GM が製造したパワーアンプには、ピンとパラメーターを HAL にエクスポートする上位レベルのドライバーがあります。 それらは位置基準を受け取り、CAN バスを介してエンコーダフィードバックを提供します。

フレームは標準（11 ビット）ID フレームで、データ長は 4 バイトです。 ボーレートは 1 メガビットです。 軸 0..5 の位置コンマ ID は 0x10..0x15 です。 軸 0..5 の位置フィードバック ID は 0x20..0x25 です。

これらの構成は、HAL_GM.C を変更し、LINUXCNC を再コンパイルすることで変更できます。

すべての CAN ピンとパラメータ名は次のように始まります。

```
gm.<nr. of card>.can-gm.<nr of axis>
```

、ここで、<軸の NR>は 0 から 5 の形式です。例：

```
gm.0.can-gm.0.position
```

軸 0 の出力位置を位置単位で表します。

ハルピンは機能によって更新されます：

gm.<nr of card>.write

1.1.1.1 ピン

Table 11.14: CAN module pins

Pins	Type and direction	Pin description
.enable	(bit, In)	Enable sending position references.
.position-cmd	(float, In)	Commanded position in position units.
.position-fb	(float, In)	Feed back position in position units.

9.13.3.1 パラメータ

Table 11.15: CAN module parameters

Parameters	Type and direction	Parameter description
.position-scale	(float, R/W)	Scale in per length unit.

9.13.4 ウォッチドッグタイマー

ウォッチドッグタイマーは機能時にリセットされます：

gm.<nr of card>.read

1.1.1.1 ピン

Table 11.16: Watchdog pins

Pins	Type and direction	Pin description
gm.<nr of card>.watchdog-expired	(bit, Out)	Indicates that watchdog timer is expired.

ウォッチドッグタイマーのオーバーランにより、ハードウェアのパワーイネーブルのセットがローになります。

9.13.4.1 パラメータ

Table 11.17: Watchdog parameters

Parameters	Type and direction	Parameter description
gm.<nr of card>.watchdog-enable	(bit, R/W)	Enable watchdog timer. It is strongly recommended to enable watchdog timer, because it can disables all the servo amplifiers by pulling down all enable signal in case of PC error.
gm.<nr of card>.watchdog-timeout-ns	(float, R/W)	Time interval in within the gm.<nr of card>.read function must be executed. The gm.<nr of card>.read is typically added to servo-thread, so watch timeout is typically set to 3 times of the servo period.

9.13.5 エンド、ホーミング、および非常停止スイッチ

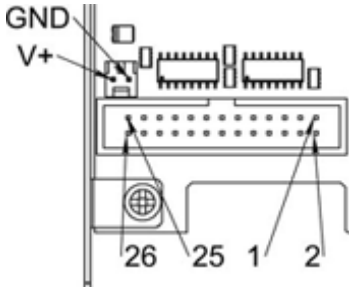


図 11-750

Table 11.18: End- and homing switch connector pinout

25	23	21	19	17	15	13	11	9	7	5	3	1
GND		1/End-	2/End+	2/Hom-ing	3/End-	4/End+	4/Hom-ing	5/End-	6/End+	6/Hom-ing	E-Stop 2	V+ (Ext.)

26	24	22	20	18	16	14	12	10	8	6	4	2
GND		1/End+	1/Hom-ing	2/End-	3/End+	3/Hom-ing	4/End-	5/End+	5/Hom-ing	6/End-	E-Stop 1	V+ (Ext.)

GM6-PCI モーションコントロールカードには、各ジョイントに 2 つのリミットスイッチ入力と 1 つのホーミングスイッチ入力があります。 これらのピンの名前はすべて次のように始まります。

```
gm.<nr. of card>.joint.<nr of axis>
```

、ここで、軸の NR は 0 から 5 の形式です。たとえば

```
gm.0.joint.0.home-sw-in
```

軸 0 ホームスイッチの状態を示します。

ハルピンは機能によって更新されます：

```
gm.<nr of card>.read
```

1.1.1.1 ピン

Table 11.19: End- and homing switch pins

Pins	Type and direction	Pin description
.home-sw-in	(bit, Out)	Home switch input
.home-sw-in-not	(bit, Out)	Negated home switch input
.neg-lim-sw-in	(bit, Out)	Negative limit switch input
.neg-lim-sw-in-not	(bit, Out)	Negated negative limit switch input
.pos-lim-sw-in	(bit, Out)	Positive limit switch input
.pos-lim-sw-in-not	(bit, Out)	Negated positive limit switch input

9.13.5.1 パラメータ

Table 11.20: E-stop switch parameters

Parameters	Type and direction	Parameter description
gm.0.estop.0.in	(bit, Out)	Estop 0 input
gm.0.estop.0.in-not	(bit, Out)	Negated Estop 0 input
gm.0.estop.1.in	(bit, Out)	Estop 1 input
gm.0.estop.1.in-not	(bit, Out)	Negated Estop 1 input

9.13.6 ステータス LED

1.1.1.1 CAN

オレンジ色

- データ通信中に点滅します。
- オン、いずれかのバッファがいっぱいになると、通信エラーが発生します。
- データ通信がない場合はオフ。

9.13.6.1 RS485

オレンジ色

- バス上のモジュールの初期化中に点滅
- オン、初期化されたすべてのモジュール間でデータ通信がアップしているとき。
- オフ。初期化されたモジュールのいずれかがエラーのためにドロップオフした場合。

9.13.6.2 EMC

カラー：ホワイト

- LinuxCNC の実行中に点滅します。
- それ以外の場合はオフ。

9.13.6.3 Boot

赤色

- オフ、システムに障害がない場合。
- PCI 通信エラー時に点滅します。
- オン、ウォッチドッグタイマーがオーバーフローしたとき。

9.13.7 RS485 I / O エキスパンダーモジュール

これらのモジュールは、GM6-PCI モーションコントロールカードの RS485 ラインに沿って I / O および機能機能を拡張するために開発されました。

使用可能なモジュールタイプ：

- 8 チャンネルリレー出力モジュール-各チャンネルの 3 極端子コネクタに 8 つの NO-NC リレー出力を提供します。
- 8 チャンネルデジタル入力モジュール-8 つの光学的に絶縁されたデジタル入力ピンを提供します。
- 8 チャンネル ADC および 4 チャンネル DAC モジュール-4 つのデジタル-アナログコンバーター出力と 8 つのアナログ-デジタル入力を提供します。 このモジュールは、GM6-PCI カードからも光学的に絶縁されています。

自動ノード認識：

バスに接続されている各ノードは、GM6-PCI カードによって自動的に認識されました。 LINUXCNC の起動中に、ドライバーは使用可能なすべてのモジュールのピンとパラメーターを自動的にエクスポートします。

障害処理：

モジュールが定期的に応答しない場合、GM6-PCI カードはモジュールをドロップダウンします。 出力のあるモジュールが正しい CRC のデータを定期的に取得しない場合、モジュールはエラー状態に切り替わり（緑色の LED が点滅）、すべての出力をエラー状態に切り替えます。

ノードの接続：

バス上のモジュールは、終端抵抗を備えたシリアルトポロジで接続する必要があります。 トポロジの開始は PCI カードであり、終了は最後のモジュールです。

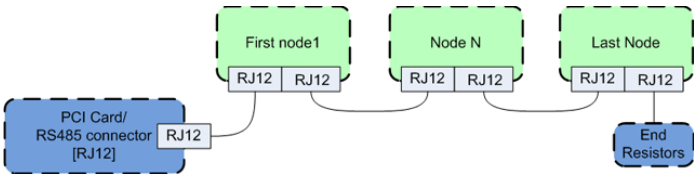


図 11-11

アドレッシング：

バス上の各ノードには、赤い DIP スイッチで設定できる 4 ビットの一意のアドレスがあります。

ステータス LED：

緑色の LED は、モジュールのステータスを示します。

- モジュールに電力が供給されているだけでジェットが識別されていない場合、またはモジュールがドロップダウンした場合に点滅します。
- オフ、識別中（コンピューターはオンですが、LINUXCNC は起動していません）
- オン、継続的に通信する場合。

1.1.1.1 リレー出力モジュール

モジュールのピン配置、接続、および電気的特性については、システム統合マニュアルを参照してください。

すべてのピンとパラメータは、次の機能によって更新されます。

```
gm.<nr. of card>.rs485
```

CPU の過負荷を避けるために、サーボスレッドまたは他のより長い周期のスレッドに追加する必要があります。 すべての RS485 モジュールのピンとパラメータ名は次のように始まります。

```
gm.<nr. of card>.rs485.<modul ID>
```

、ここで、<MODULID>は 00 から 15 の形式です。

Table 11.21: Relay output module pins

Pins	Type and direction	Pin description
.relay-<0-7>	(bit, Out)	Output pin for relay

Table 11.22: Relay output module parameters

Parameters	Type and direction	Parameter description
.invert-relay-<0-7>	(bit, R/W)	Negate relay output pin

HAL の例

```
gm.0.rs485.0.relay-0 # First relay of the node.
gm.0 # Means the first GM6-PCI motion control card (PCI card address = 0)
.rs485.0 # Select node with address 0 on the RS485 bus
.relay-0 # Select the first relay
```

9.13.7.1 デジタル入力モジュール

モジュールのピン配置、接続、および電気的特性については、システム統合マニュアルを参照してください。

すべてのピンとパラメータは、次の機能によって更新されます。

```
gm.<nr. of card>.rs485
```

CPU の過負荷を避けるために、サーボスレッドまたは他のより長い周期のスレッドに追加する必要があります。 すべての RS485 モジュールのピンとパラメータ名は次のように始まります。

```
gm.<nr. of card>.rs485.<modul ID>
```

、ここで、<MODULID>は 00 から 15 の形式です。

Table 11.23: Digital input output module pins

Pins	Type and direction	Pin description
.in-<0-7>	(bit, Out)	Input pin
.in-not-<0-7>	(bit, Out)	Negated input pin

HAL の例

```
gm.0.rs485.0.in-0 # First input of the node.
# gm.0 - Means the first GM6-PCI motion control card (PCI card address = 0)
# .rs485.0 - Select node with address 0 on the RS485 bus
# .in-0 - Select the first digital input module
```

9.13.7.2 DAC&ADC モジュール

モジュールのピン配置、接続、および電気的特性については、システム統合マニュアルを参照してください。

すべてのピンとパラメータは、次の機能によって更新されます。

```
gm.<nr. of card>.rs485
```

CPU の過負荷を避けるために、サーボスレッドまたは他のより長い周期のスレッドに追加する必要があります。 すべての RS485 モジュールのピンとパラメータ名は次のように始まります。

```
gm.<nr. of card>.rs485.<modul ID>
```

、ここで、<MODULID>は 00 から 15 の形式です。

Table 11.24: DAC & ADC module pins

Pins	Type and direction	Pin description
.adc-<0-7>	(float, Out)	Value of ADC input in Volts.
.dac-enable-<0-3>	(bit, In)	Enable DAC output. When enable is false DAC output is set to 0.0 V.
.dac-<0-3>	(float, In)	Value of DAC output in Volts.

Table 11.25: DAC & ADC module parameters

Parameters	Type and direction	Parameter description
.adc-scale-<0-7>	(float, R/W)	The input voltage will be multiplied by scale before being output to .adc- pin.
.adc-offset-<0-7>	(float, R/W)	Offset is subtracted from the hardware input voltage after the scale multiplier has been applied.
.dac-offset-<0-3>	(float, R/W)	Offset is added to the value before the hardware is updated.
.dac-high-limit-<0-3>	(float, R/W)	Maximum output voltage of the hardware in volts.
.dac-low-limit-<0-3>	(float, R/W)	Minimum output voltage of the hardware in volts.

HAL の例

```
gm.0.rs485.0.adc-0 # First analogue channel of the node.
# gm.0 - Means the first GM6-PCI motion control card (PCI card address = 0)
# .rs485.0 - Select node with address 0 on the RS485 bus
# .adc-0 - Select the first analogue input of the module
```

9.13.7.3 ペンダントモジュールを教える

モジュールのピン配置、接続、および電気的特性については、システム統合マニュアルを参照してください。

すべてのピンとパラメータは、次の機能によって更新されます。

```
gm.<nr. of card>.rs485
```

CPU の過負荷を避けるために、サーボスレッドまたは他のより長い周期のスレッドに追加する必要があります。 すべての RS485 モジュールのピンとパラメータ名は次のように始まります。

```
gm.<nr. of card>.rs485.<modul ID>
```

、ここで、<MODUL ID>は 00 から 15 の形式です。TEACHPENDANT モジュールでは変更できず、ゼロとして事前にプログラムされていることに注意してください。 リクエストに応じて、ファームウェアで事前にプログラムされた異なる ID で配信できます。

Table 11.26: Teach Pendant module pins

Pins	Type and direction	Pin description
.adc-<0-5>	(float, Out)	Value of ADC input in Volts.
.enc-reset	(bit, In)	When True, resets counts and position to zero.
.enc-counts	(s32, Out)	Position in encoder counts.
.enc-rawcounts	(s32, Out)	The raw count is the counts, but unaffected by reset.
.enc-position	(float, Out)	Position in scaled units (=.enc-counts/.enc-position-scale).
.in-<0-7>	(bit, Out)	Input pin
.in-not-<0-7>	(bit, Out)	Negated input pin

Table 11.27: Teach Pendant module parameters

Parameters	Type and direction	Parameter description
.adc-scale-<0-5>	(float, R/W)	The input voltage will be multiplied by scale before being output to .adc- pin.
.adc-offset-<0-5>	(float, R/W)	Offset is subtracted from the hardware input voltage after the scale multiplier has been applied.
.enc-position-scale	(float, R/W)	Scale in per length unit.

HAL の例

```
gm.0.rs485.0.adc-0 # First analogue channel of the node.
# gm.0 - Means the first GM6-PCI motion control card (PCI card address = 0)
# .rs485.0 - Select node with address 0 on the RS485 bus
# .adc-0 - Select the first analogue input of the module
```

9.13.8 エラッタ

1.1.1.1 GM6-PCI カードエラッタ

このセクションのリビジョン番号は、GM6-PCI カードデバイスのリビジョンを示しています。

REV. 1.2

- エラー：AXIS 1 の場合、PCI カードは起動しません。ENDB スイッチがアクティブ（ロー）です。2013 年 11 月 16 日に見つかりました。
- 理由：このスイッチが FPGA のブート設定ピンに接続されている
- 問題の修正/回避策：他のスイッチピンを使用するか、通常開のスイッチのみをこのスイッチ入力ピンに接続します。

9.14 VFS11VFD ドライバー

これは、東芝の VFD の S11 シリーズを制御するためのユーザースペース HAL プログラムです。

vfs11_vfd は、シリアル接続と TCP 接続をサポートします。 シリアル接続は RS232 または RS485 の場合があります。 RS485 は全二重モードと半二重モードでサポートされています。 TCP 接続は、パッシブ（着信接続を待機）またはアクティブな発信接続の場合があります。これは、TCP ベースのデバイスまたはターミナルサーバーを介して接続する場合に役立ちます。

接続タイプに関係なく、vfs11_vfd は Modbus マスターとして動作します。

このコンポーネントは、halcmd "loadusr"コマンドを使用してロードされます。

```
loadusr -Wn spindle-vfd vfs11_vfd -n spindle-vfd
```

上記のコマンドは次のように述べています：loadusr、名前付きがロードされるのを待つ、コンポーネント vfs11_vfd、名前付きスピンドル-vfd

9.14.1 コマンドラインオプション

VFS11_VFD は、ほとんどの場合、INIFILE オプションを使用して構成されます。 コマンドラインオプションは次のとおりです。

- -n または--name <halname>：HAL コンポーネント名を設定します
- -I または--ini <inifilename>：この ini ファイルから構成を取得します。 デフォルトは環境変数 INI_FILE_NAME です。
- -S または--section <セクション名>：ini ファイルのこのセクションから構成を取得します。 デフォルトは VFS11 です。

- -d または--debug は、コンソール出力でデバッグメッセージを有効にします。
- -m または--modbus-debug は、コンソール出力で modbus メッセージを有効にします
- -r または--report-device は、起動時にコンソールのデバイスプロパティを報告します

USR1 信号を vfs11_vfd プロセスに送信することにより、デバッグを切り替えることができます。Modbus デバッグは、USR2 信号を vfs11_vfd プロセスに送信することで切り替えることができます（例：kill -USR1 `pidof vfs11_vfd`）。

Note

シリアル構成エラーがある場合、verbose をオンにすると、タイムアウトエラーが大量に発生する可能性があります。

9.14.2 ピン

ここで、<n>は vfs11_vfd、またはロード中に-n オプションを使用して指定された名前です。

- <n> .acceleration-pattern（bit、in）true の場合、レジスタ F500 および F501 でそれぞれ定義されているように加速時間と減速時間を設定します。 発振を回避するために短いランプ時間を選択するために PID ループで使用されます。
- <n> .alarm-code（s32、out）ドライブがアラーム状態の場合、ゼロ以外。 アラーム情報を説明するビットマップ（レジスタ FC91 の説明を参照）。 err-reset（以下を参照）を使用して、アラームをクリアします。
- <n> .at-speed（bit、out）ドライブがコマンド速度の場合（下記の速度許容差を参照）
- <n> .current-load-percentage（float、out）が VFD から報告されました
- <n> .dc-brake（bit、in）は DC ブレーキを作動させます。 また、スピンドルオンをオフにします。
- <n> .enable（bit、in）は VFD を有効にします。 false の場合、すべての操作パラメーターは引き続き読み取られますが、制御は解放され、パネル制御が有効になります（VFD セットアップの対象）。
- <n> .err-reset（bit、in）リセットエラー（アラーム、別名トリップおよび非常停止ステータス）。VFD をリセットすると、再起動して Modbus が再び起動するまで 2 秒の遅延が発生する場合があります。
- <n> .estop（bit、in）は、VFD を緊急停止状態にします。 エラーリセットまたはパワーサイクリングでクリアされるまで、操作はできません。
- <n> .frequency-コマンド（float、out）VFD からの speed-command（RPM 単位）を介して設定された HZ 単位の現在のターゲット周波数
- <n> .frequency-out（float、out）VFD の現在の出力周波数

- <n> .inverter-load-percentage（float、out）VFD からの現在の負荷レポート
- <n> .is-e-stopped（bit、out）VFD は緊急停止状態です（パネルの「E」が点滅）。 err-reset を使用して VFD をリブートし、非常停止ステータスをクリアします。
- <n> .is-stopped（bit、out）VFD が 0Hz 出力を報告する場合は true
- <n> .max-rpm（float、R）VFD が生成する可能性のある最大周波数、およびモーターの銘板の値に基づく実際の RPM 制限。 たとえば、ネームプレート-HZ が 50 で、ネームプレート-RPM_が 1410 であるが、VFD が最大 80Hz を生成する可能性がある場合、max-rpm は 2256（80 * 1410/50）として読み取られます。 周波数制限は、起動時に VFD から読み取られます。 周波数の上限を上げるには、パネルで UL および FH パラメーターを変更する必要があります。 最大周波数の設定方法については、VF-S11 のマニュアルをご覧ください。
- <n> .modbus-ok（bit、out）Modbus セッションが正常に確立され、最後の 10 個のトランザクションがエラーなしで返された場合は true。
- <n> .motor-RPM（float、out）VFD からの現在の RPM 値の推定値
- VFD からの<n> .output-current-percentage（float、out）
- <n> .output-電圧-VFD からのパーセンテージ（フロート、アウト）
- <n> .VFD からの出力電圧（フロート、アウト）
- <n> .speed-コマンド（float、in）速度が RPM で VFD に送信されます。 VFD で設定されているモーター最大 RPM よりも速い速度を送信するとエラーになります
- <n> .spindle-fwd（bit、in）FWD の場合は 1、REV の場合は 0、VFD に送信
- <n> .spindle-on（bit、in）VFD に送信されるオンの場合は 1、オフの場合は 0、実行中のみオン
- <n> .spindle-rev（bit、in）ON の場合は 1、OFF の場合は 0、実行中のみオン
- <n> .jog-mode（bit、in）ON の場合は 1、OFF の場合は 0 で、VF-S11 ジョグモードを有効にします。 速度制御は無効であり、出力周波数はレジスタ F262（5Hz にプリセット）によって決定されます。 これは、スピンドルの向きに役立つ場合があります。 通常モードでは、周波数が 12Hz を下回ると、VFD はシャットオフします。
- <n> .status（s32、out）VFD のドライブステータス（TOSVERT VF-S11 通信機能取扱説明書、レジスタ FD01 を参照）。 ビットマップ。
- <n> .trip-code（s32、out）VF-S11 がトリップ状態の場合のトリップコード。
- <n> .error-count（s32、out）エラーを返した Modbus トランザクションの数
- <n> .max-speed（bit、in）は、ループ時間パラメーターを無視し、CPU 使用率を犠牲にして、Modbus を最大速度で実行します。 スピンドル位置決め中の推奨使用法。

9.14.3 パラメータ

ここで、<n>は vfs11_vfd、またはロード中に-n オプションを使用して指定された名前です。

- <n> .frequency-limit（float、RO）VFD セットアップから読み取られた上限。
- <n> .loop-time（float、RW）Modbus がポーリングされる頻度（デフォルト間隔 0.1 秒）
- <n> .nameplate-HZ（float、RW）モーターのネームプレート Hz（デフォルトは 50）。 speed-command で指定されたターゲット RPM 値のターゲット周波数を（ネームプレート-RPM とともに）計算するために使用されます。
- <n> .nameplate-RPM（float、RW）モーターのネームプレート RPM（デフォルトは 1410）
- <n> .rpm-limit（float、RW）は、モーター RPM のソフト制限を超えません（デフォルトはネームプレート-RPM）。
- <n> .tolerance（float、RW）スピンドルが速度にあるかどうかを判断するための速度許容値（デフォルトは 0.01）（0.01 は、出力周波数が目標周波数の 1%以内であることを意味します）

9.14.4 INI ファイルの構成

これは、vfs11_vfd によって理解されるすべてのオプションを一覧表示します。 RS232、RS485、および TCP の一般的なセットアップは、src / hal / user_comps / vfs11_vfd / *にあります。

```
[VFS11]
# serial connection
TYPE=rtu
# serial port
DEVICE=/dev/ttyS0
# TCP server - wait for incoming connection
TYPE=tcpserver
# tcp portnumber for TYPE=tcpserver or tcpclient
PORT=1502
# TCP client - active outgoing connection
TYPE=tcpclient
# destination to connect to if TYPE=tcpclient
TCPDEST=192.168.1.1
#---------- meaningful only if TYPE=rtu -------
# serial device detail
# 5 6 7 8
BITS= 5
# even odd none
PARITY=none
# 110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
```

```
BAUD=19200
# 1 2
STOPBITS=1
#rs232 rs485
SERIAL_MODE=rs485
# up down none
# this feature might not work with a stock Ubuntu
# libmodbus5/libmodbus-dev package, and generate a warning
# execution will continue as if RTS_MODE=up were given.
RTS_MODE=up
#---------------------
# modbus timers in seconds
# inter-character timer
BYTE_TIMEOUT=0.5
# packet timer
RESPONSE_TIMEOUT=0.5
# target modbus ID
TARGET=1
# on I/O failure, try to reconnect after sleeping
# for RECONNECT_DELAY seconds
RECONNECT_DELAY=1
# misc flags
DEBUG=10
MODBUS_DEBUG=0
POLLCYCLES=10
```

9.14.5 HALの例

```
#
# example usage of the VF-S11 VFD driver
#
#
loadusr -Wn spindle-vfd vfs11_vfd -n spindle-vfd
# connect the spindle direction pins to the VFD
net vfs11-fwd spindle-vfd.spindle-fwd <= spindle.0.forward
net vfs11-rev spindle-vfd.spindle-rev <= spindle.0.reverse
# connect the spindle on pin to the VF-S11
net vfs11-run spindle-vfd.spindle-on <= spindle.0.on
# connect the VF-S11 at speed to the motion at speed
net vfs11-at-speed spindle.0.at-speed <= spindle-vfd.at-speed
# connect the spindle RPM to the VF-S11
```

```
net vfs11-RPM spindle-vfd.speed-command <= spindle.0.speed-out
# connect the VF-S11 DC brake
# since this draws power during spindle off, the dc-brake pin would
# better be driven by a monoflop which triggers on spindle-on falling edge
#net vfs11-spindle-brake spindle.N.brake => spindle-vfd.dc-brake
# to use the VFS11 jog mode for spindle orient
# see orient.9 and motion.9
net spindle-orient spindle.0.orient spindle-vfd.max-speed spindle-vfd.jog-mode
# take precedence over control panel
setp spindle-vfd.enable 1
```

9.14.6 パネル操作

vfs11_vfd ドライバーは、有効になっている間はパネル制御よりも優先され（有効ピンを参照）、パネルを効果的に無効にします。 イネーブルピンをクリアすると、パネルが再び有効になります。 ピンとパラメータは引き続き設定できますが、イネーブルピンが設定されるまで VFD に書き込まれません。 バス制御が無効になっている間も、動作パラメータは読み取られます。 制御された方法で vfs11_vfd ドライバーを終了すると、VFD がバスから解放され、パネル制御が復元されます。

詳細については、EMC2 インテグレータのマニュアルを参照してください。 東芝 VFD のレジスタの詳細については、「TOSVERT VF-S11 通信機能取扱説明書」（東芝文書番号 E6581222）および「TOSVERT VF-S11 取扱説明書」（東芝文書番号 E6581158）を参照してください。

9.14.7 エラー回復

vfs11_vfd は、次のように I / O エラーから回復します。最初に、すべての HAL ピンがデフォルト値に設定され、ドライバーは RECONNECT_DELAY 秒（デフォルトは 1 秒）スリープします。

- シリアル（TYPE = rtu）モード：エラーが発生した場合は、シリアルポートを閉じてから再度開きます。
- TCP サーバー（TYPE = tcpserver）モード：TCP 接続が失われると、ドライバーは着信接続をリッスンするために戻ります。
- TCP クライアント（TYPE = tcpclient）モード：TCP 接続が失われると、ドライバーは TCPDEST：PORTNO に再接続します。

9.14.8 Modbus を使用するための VFS11VFD の構成

1.1.1.1 シリアルポートの接続

VF-S11 にはシリアル通信用の RJ-45 ジャックがあります。 残念ながら、標準の RS-232 プラグとロジックレベルはありません。 東芝が推奨する方法は、USB001Z USB-シリアル変換ユニットをドライ

ブに接続し、USB ポートを PC に接続することです。 より安価な代替手段は、自作のインターフェースです（東芝のサポートからのヒント、回路図）。

注：VFD からの 24V 出力には短絡保護がありません。

シリアルポートの工場出荷時のデフォルトは 9600/81/1 / even で、プロトコルのデフォルトは独自の「ToshibaInverterProtocol」です。

9.14.8.1 Modbus のセットアップ

VF-S11 がこのモジュールと通信する前に、いくつかのパラメータを設定する必要があります。 これは、コントロールパネルを使用して手動で実行することも、シリアルリンクを介して実行することもできます。東芝は、VFD のパラメータを読み取り/設定できる PCM001Z と呼ばれる Windows アプリケーションを提供しています。 注-PCM001Z は、Toshiba インバータプロトコルのみを通信します。したがって、変更する最後のパラメータはプロトコルです。ToshibaInverterProtocol から Modbus に設定します。 その後、Windows アプリは役に立たなくなります。

周波数の上限を上げるには、パネルで UL および FH パラメーターを変更する必要があります。 それらを 50 から 80 に増やしました。セットアップの非標準 VF-S11 パラメーターの説明については、dump-params.mio を参照してください。 このファイルは、modioModbus インタラクティブユーティリティ用です。

9.14.8.2 プログラミングノート

vfs11_vfd ドライバーは、gs2_vfd で使用されているバージョン 2 コードよりも新しい libmodbus バージョン 3 ライブラリを使用します。

Ubuntu の libmodbus5 および libmodbus-dev パッケージは、Ubuntu 12（Precise Pengolin）以降でのみ使用できます。 さらに、これらのパッケージは MODBUS_RTS_MODE_ *フラグのサポートを欠いています。 したがって、このライブラリを使用して vfs11_vfd をビルドすると、init ファイルで RTS_MODE =が指定されている場合に警告が生成される可能性があります。

明快で正確に全機能を使用するには：* libmodbus パッケージを削除します：sudo apt-get remove libmodbus5 libmodbus-dev *ここに概説されているように、ソースから libmodbus バージョン 3 をビルドしてインストールします。

Libmodbus は UbuntuHardy でビルドされないため、vfs11_vfd は hardy では使用できません。

10章　ドライバーの例

10.1　PCI パラレルポート

PCI バスに 2 つ目のパラレルポートを追加する場合、LinuxCNC で使用する前に、アドレスを確認する必要があります。 パラレルポートカードのアドレスを見つけるには、ターミナルウィンドウを開いて次のように入力します

```
lspci -v
```

これに似たものと、PCI バス上の他のすべての情報が表示されます。

```
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]
```

私の場合、アドレスが最初のものだったので、.hal ファイルをから変更しました

```
loadrt hal_parport cfg=0x378
```

に

```
loadrt hal_parport cfg="0x378 0xa800 in"
```

（アドレスを囲む二重引用符に注意してください。）

次に、パーポートが読み書きされるように、次の行を追加しました。

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

上記を実行した後、構成を実行し、パラレルポートが[マシン/ HAL 構成の表示]ウィンドウにロードされたことを確認します。

LinuxCNC は、最大 8 つのスピンドルを制御できます。 番号は INI ファイルに設定されています。 以下の例はすべて、spindle.0 のような名前のスピンドル制御ピンを備えたシングルスピンドル構成を参照しています。 。 。 マルチスピンドルマシンの場合、変更されるのは、スピンドルなどの名前の追加のピンが存在することだけです。 。 。

10.2 スピンドル制御

10.2.1 0-10v スピンドル速度

スピンドル速度がアナログ信号（たとえば、0〜10 ボルトの信号の VFD）によって制御されており、m5i20 などの DAC カードを使用して制御信号を出力している場合：

まず、信号を制御するためのスピンドル速度のスケールを把握する必要があります。 この例では、5000RPM のスピンドル最高速度は 10 ボルトに等しくなります。

$$\frac{10\,Volts}{5000\,RPM} = \frac{0.002\,Volts}{1\,RPM}$$

DAC カードがスケーリングを行わない場合、スピンドルをスケーリングするために、HAL ファイルにスケールコンポーネントを追加する必要があります。N.speed-out を VFD に必要な 0 から 10 に調整します。

```
loadrt scale count=1
addf scale.0 servo-thread
setp scale.0.gain 0.002
net spindle-speed-scale spindle.0.speed-out => scale.0.in
net spindle-speed-DAC scale.0.out => <your DAC pin name>
```

10.2.2 PWM スピンドル速度

スピンドルを PWM 信号で制御できる場合は、pwmgen コンポーネントを使用して信号を作成します。

```
loadrt pwmgen output_type=0
addf pwmgen.update servo-thread
addf pwmgen.make-pulses base-thread
net spindle-speed-cmd spindle.0.speed-out => pwmgen.0.value
net spindle-on spindle.0.on => pwmgen.0.enable
net spindle-pwm pwmgen.0.pwm => parport.0.pin-09-out
# Set the spindle’s top speed in RPM
setp pwmgen.0.scale 1800
```

これは、PWM に対するスピンドルコントローラーの応答が単純であることを前提としています。0%PWM は 0 RPM、10%PWM は 180 RPM などです。スピンドルを回転させるために必要な最小 PWM がある場合は、nist-lathe の例に従ってください。 スケールコンポーネントを使用するためのサンプル構成。

10.2.3 スピンドルの有効化

スピンドルイネーブル信号が必要な場合は、出力ピンをスピンドル.0.on にリンクします。 これらのピンをパラレルポートピンにリンクするには、次のようなものを.hal ファイルに入れ、制御デバイスに接続されているピンを選択していることを確認します。

```
net spindle-enable spindle.N.on => parport.0.pin-14-out
```

10.2.4 スピンドル方向

スピンドルの方向制御がある場合、HAL ピンのスピンドル.N。フォワードとスピンドル.N。リバースは M3 と M4 によって制御されます。 スピンドルモーションをオンにするには、M3 / M4 のスピンドル速度 Sn をゼロ以外の正の値に設定する必要があります。

これらのピンをパラレルポートピンにリンクするには、次のようなものを.hal ファイルに入れて、制御デバイスに接続されているピンを選択していることを確認します。

```
net spindle-fwd spindle.0.forward => parport.0.pin-16-out
net spindle-rev spindle.0.reverse => parport.0.pin-17-out
```

10.2.5 スピンドルソフトスタート

スピンドル速度コマンドをランプする必要があり、コントロールにその機能がない場合は、HAL で実行できます。 基本的に、spindle.N.speed-out の出力をハイジャックし、スケールが設定された limit2 コンポーネントを実行して、rpm を spindle.N.speed-out から rpm を受信するデバイスにランプするようにする必要があります。 2 番目の部分は、スピンドルが速度を上げていることを LinuxCNC に通知して、モーションを開始できるようにすることです。

0〜10 ボルトの例では、ラインネットスピンドル-速度-スケールスピンドル.0.speed-out => scale.0.in が次のように変更されます。

例：

HAL コンポーネント limit2 およびその近くの概要：

これまでに遭遇したことがない場合のために、以下で使用される 2 つの HAL コンポーネントの簡単な紹介を示します。

example.

- 「limit2」は、入力値を受け入れ、最大/最小範囲に制限され、指定された変化率を超えないように制限された出力を提供する HAL コンポーネント（浮動小数点）です。
- 「near」は、2 つの入力がほぼ等しいかどうかを示すバイナリ出力を持つ HAL コンポーネント（浮動小数点）です。

詳細については、HAL コンポーネントのドキュメント、または man ページから入手できます。manlimit2 またはターミナルの近くにいる man とだけ言ってください。

```
# load real time a limit2 and a near with names so it is easier to follow
loadrt limit2 names=spindle-ramp
loadrt near names=spindle-at-speed
# add the functions to a thread
addf spindle-ramp servo-thread
addf spindle-at-speed servo-thread
```

```
# set the parameter for max rate-of-change
# (max spindle accel/decel in units per second)
setp spindle-ramp.maxv 60
# hijack the spindle speed out and send it to spindle ramp in
net spindle-cmd <= spindle.0.speed-out => spindle-ramp.in
# the output of spindle ramp is sent to the scale in
net spindle-ramped <= spindle-ramp.out => scale.0.in
# to know when to start the motion we send the near component
# (named spindle-at-speed) to the spindle commanded speed from
# the signal spindle-cmd and the actual spindle speed
# provided your spindle can accelerate at the maxv setting.
net spindle-cmd => spindle-at-speed.in1
net spindle-ramped => spindle-at-speed.in2
# the output from spindle-at-speed is sent to spindle.0.at-speed
# and when this is true motion will start
net spindle-ready <= spindle-at-speed.out => spindle.0.at-speed
```

10.2.6 スピンドルフィードバック

1.1.1.1 スピンドル同期モーション

LinuxCNC は、ねじ切りや一定の表面速度などのスピンドル協調動作を実行するために、スピンドルフィードバックが必要です。 LinuxCNC は、最大 8 つのスピンドルのいずれかで同期モーションと CSS を実行できます。 使用するスピンドルは G コードで制御します。 CSS は、複数のスピンドルで同時に使用できます。

入力としてエンコーダーフェーズ A とエンコーダーインデックスを選択すると、StepConf ウィザードでシングルスピンドル構成の接続を実行できます。

ハードウェアの前提：

- エンコーダがスピンドルに接続され、フェーズ A で 1 回転あたり 100 パルスを出力します。
- エンコーダ A 相はパラレルポートピン 10 に接続されています
- エンコーダインデックスパルスはパラレルポートピン 11 に接続されています

コンポーネントを追加して構成するための基本的な手順： 1) 2) 3)

```
# add the encoder to HAL and attach it to threads.
loadrt encoder num_chan=1
addf encoder.update-counters base-thread
addf encoder.capture-position servo-thread
# set the HAL encoder to 100 pulses per revolution.
setp encoder.3.position-scale 100
# set the HAL encoder to non-quadrature simple counting using A only.
```

```
setp encoder.3.counter-mode true
# connect the HAL encoder outputs to LinuxCNC.
net spindle-position encoder.3.position => spindle.0.revs
net spindle-velocity encoder.3.velocity => spindle.0.speed-in
net spindle-index-enable encoder.3.index-enable <=> spindle.N.index-enable
# connect the HAL encoder inputs to the real encoder.
net spindle-phase-a encoder.3.phase-A <= parport.0.pin-10-in
net spindle-phase-b encoder.3.phase-B
net spindle-index encoder.3.phase-Z <= parport.0.pin-11-in
```

---

1)この例では、いくつかのエンコーダーが軸/ジョイント 0、1、および 2 にすでに発行されていると想定します。したがって、スピンドルに接続できる次のエンコーダーは 3 番です。状況は異なる場合があります。

2)HAL エンコーダーのインデックス有効化は、入力と出力の両方として動作するという点でルールの例外です。詳細については、エンコーダーのセクションを参照してください。

3)非求積単純カウントを選択したためです。 。 。 その上では、B 求積法の入力がなくても求積法のカウントを回避できます。

10.2.6.1 Spindle At Speed

LinuxCNC が一連の移動を実行する前に、スピンドルの速度が上がるのを待つことができるようにするため。 スピンドルが指令された速度にあるときは、spindle.N.at-speed を true に設定する必要があります。 これを行うには、エンコーダからのスピンドルフィードバックが必要です。 フィードバックと指令速度は通常完全に同じではないため、near コンポーネントを使用して 2 つを決定する必要があります 数字は十分に近いです。

必要な接続は、スピンドル速度コマンド信号から near.n.in1 まで、およびエンコーダから near.n.in2 までのスピンドル速度です。 次に、near.n.out が spindle.N.at-speed に接続されます。 near.n.scale は、出力をオンにする前に 2 つの数値をどれだけ近づける必要があるかを示すように設定する必要があります。 セットアップによっては、ハードウェアで機能するようにスケールを調整する必要がある場合があります。

以下は、Spindle AtSpeed を有効にするために HAL ファイルに必要な追加の典型的なものです。 すでに HAL ファイルに近い場合は、カウントを増やし、それに合わせてコードを調整します。 信号名が HAL ファイルで同じであることを確認してください。

```
# load a near component and attach it to a thread
loadrt near
addf near.0 servo-thread
# connect one input to the commanded spindle speed
net spindle-cmd => near.0.in1
# connect one input to the encoder-measured spindle speed
net spindle-velocity => near.0.in2
```

```
# connect the output to the spindle-at-speed input
net spindle-at-speed spindle.0.at-speed <= near.0.out
# set the spindle speed inputs to agree if within 1%
setp near.0.scale 1.01
```

10.3 MPG ペンダント

この例は、今日市場に出回っている一般的な MPG ペンダントを接続する方法を説明するためのものです。 この例では、PCI スロットに接続された 2 番目のパラレルポートに接続された CNC4PC の MPG3 ペンダントと C22 ペンダントインターフェイスカードを使用します。 この例では、0.1、0.01、0.001 の 3 ステップ増分で 3 つの軸が得られます。

custom.hal ファイルまたは jog.hal ファイルに以下を追加し、mux4 またはエンコーダーがすでに使用されていないことを確認します。 カウントを増やして参照番号を変更するだけの場合。 mux4 とエンコーダーの詳細については、HAL のマニュアルまたはマニュアルページを参照してください。

hal ファイルの追加の詳細については、マニュアルの HALIni セクションを参照してください。

ジョグ管理ピンは、各ジョイントとすべての座標文字に提供されます。 この例では、ワールドモードでのジョギングに軸ジョグピンを使用しています。 非同一運動学のマシンは、ジョイントモードでジョギングするために追加の接続を使用する必要がある場合があります。

jog.hal

```
# Jog Pendant
loadrt encoder num_chan=1
loadrt mux4 count=1
addf encoder.capture-position servo-thread
addf encoder.update-counters base-thread
addf mux4.0 servo-thread
# If your MPG outputs a quadrature signal per click set x4 to 1
# If your MPG puts out 1 pulse per click set x4 to 0
setp encoder.0.x4-mode 0
# For velocity mode, set to 1
# In velocity mode the axis stops when the dial is stopped
# even if that means the commanded motion is not completed,
# For position mode (the default), set to 0
# In position mode the axis will move exactly jog-scale
# units for each count, regardless of how long that might take,
setp axis.x.jog-vel-mode 0
setp axis.y.jog-vel-mode 0
setp axis.z.jog-vel-mode 0
```

```
# This sets the scale that will be used based on the input to the mux4
setp mux4.0.in0 0.1
setp mux4.0.in1 0.01
setp mux4.0.in2 0.001
# The inputs to the mux4 component
net scale1 mux4.0.sel0 <= parport.1.pin-09-in
net scale2 mux4.0.sel1 <= parport.1.pin-10-in
# The output from the mux4 is sent to each axis jog scale
net mpg-scale <= mux4.0.out
net mpg-scale => axis.x.jog-scale
net mpg-scale => axis.y.jog-scale
net mpg-scale => axis.z.jog-scale
# The MPG inputs
net mpg-a encoder.0.phase-A <= parport.1.pin-02-in
net mpg-b encoder.0.phase-B <= parport.1.pin-03-in
# The Axis select inputs
net mpg-x axis.x.jog-enable <= parport.1.pin-04-in
net mpg-y axis.y.jog-enable <= parport.1.pin-05-in
net mpg-z axis.z.jog-enable <= parport.1.pin-06-in
# The encoder output counts to the axis. Only the selected axis will move.
net encoder-counts <= encoder.0.counts
net encoder-counts => axis.x.jog-counts
net encoder-counts => axis.y.jog-counts
net encoder-counts => axis.z.jog-counts
```

マシンが MPG のクリックごとの動きをスムーズにするために高加速が可能な場合は、ilowpass コンポーネントを使用して加速を制限します。

jog.hal with ilowpass

```
loadrt encoder num_chan=1
loadrt mux4 count=1
addf encoder.capture-position servo-thread
addf encoder.update-counters base-thread
addf mux4.0 servo-thread
loadrt ilowpass
addf ilowpass.0 servo-thread
setp ilowpass.0.scale 1000
setp ilowpass.0.gain 0.01
# If your MPG outputs a quadrature signal per click set x4 to 1
# If your MPG puts out 1 pulse per click set x4 to 0
```

```
setp encoder.0.x4-mode 0
# For velocity mode, set to 1
# In velocity mode the axis stops when the dial is stopped
# even if that means the commanded motion is not completed,
```

10.4 GS2 スピンドル

この例は、Automation Direct GS2VFD を使用してスピンドルを駆動するために必要な接続を示しています。スピンドルの速度と方向は LinuxCNC によって制御されます。

GS2 コンポーネントを使用すると、セットアップはほとんど必要ありません。 Stepconf ウィザードで生成された構成から始めます。 パラレルポートの設定画面で、「SpindleCW」と「SpindlePWM」のピンが未使用に設定されていることを確認してください。

custom.hal ファイルに、LinuxCNC を GS2 に接続し、LinuxCNC にドライブを制御させるために以下を配置します。

GS2 Example

```
# load the user space component for the Automation Direct GS2 VFD's
loadusr -Wn spindle-vfd gs2_vfd -r 9600 -p none -s 2 -n spindle-vfd
# connect the spindle direction pin to the GS2
net gs2-fwd spindle-vfd.spindle-fwd <= spindle.N.forward
# connect the spindle on pin to the GS2
net gs2-run spindle-vfd.spindle-on <= spindle.N.on
# connect the GS2 at speed to the motion at speed
net gs2-at-speed spindle.N.at-speed <= spindle-vfd.at-speed
# connect the spindle RPM to the GS2
net gs2-RPM spindle-vfd.speed-command <= spindle.N.speed-out
```

note

正確な環境によっては、伝送速度を上げることができる場合があります。 ドライブとコマンドラインオプションの両方が一致する必要があります。 送信エラーをチェックするには、-v コマンドラインオプションを追加して、ターミナルから実行します。

GS2 ドライブ自体では、modbus 通信が機能する前にいくつかの設定を行う必要があります。 その他のパラメータは、物理的な要件に基づいて設定する必要がある場合がありますが、これらはこのマニュアルの範囲を超えています。 ドライブパラメータの詳細については、ドライブに付属の GS2 マニュアルを参照してください。

- 通信スイッチは RS-232C に設定する必要があります

- モーターパラメータは、モーターと一致するように設定する必要があります
- P3.00（操作コマンドのソース）は、RS-485 インターフェース、03 または 04 によって決定される操作に設定する必要があります
- P4.00（周波数コマンドのソース）は、RS232C / RS485 通信インターフェースによって決定される周波数に設定する必要があります。05
- P9.01（送信速度）は 9600 ボー、01 に設定する必要があります
- P9.02（通信プロトコル）は「ModbusRTU モード、8 データビット、パリティなし、2 ストップビット」に設定する必要があります、03

この例に基づく PyVCP パネルはここにあります。

11章　PLC

11.1　クラシックラダーの紹介

11.1.1　歴史

Classic Ladder は、LGPL の下でリリースされたラダーインタープリターの無料実装です。 それは MarcLeDouarain によって書かれました。 彼は彼のウェブサイトでプロジェクトの始まりを説明しています：

> 2001 年 2 月の当初、ラダー言語をテスト目的でのみプログラムすることにしました。当時勤務していた企業を辞めた後、新製品に参加する予定でした。 そして、それらの製品にラダー言語を含めることは、考慮すべき良い選択肢になると考えていました。 そこで、最小限の要素でラングを計算し、それを Gtk の下に動的に表示するための最初の行をコーディングし始め、これをすべて実現するための最初のアイデアが機能するかどうかを確認しました。
>
> そして、それが非常にうまく進んでいることにすぐに気づきましたが、タイマー、複数のラングなど、より複雑な要素を続けてきました。 。
>
> 出来上がり、これがこの作品です。 。 。 その他：それ以来、機能を追加し続けています。
>
> — ClassicLadder の Web サイトにある「Genesis」の MarcLe Douarain

Classic Ladder は、LinuxCNC の HAL で動作するように適合されており、現在 LinuxCNC と一緒に配布されています。 問題/問題/バグがある場合は、Enhanced MachineController プロジェクトに報告してください。

11.1.2　前書き

ラダーロジックまたはラダープログラミング言語は、電気論理回路図を描画する方法です。 現在では、プログラマブルロジックコントローラー（PLC）のプログラミングに非常に人気のあるグラフィカル言語です。 もともとはリレーから作られたロジックを説明するために発明されました。 この名前は、この言語のプログラムがはしごに似ており、2 つの垂直レールとそれらの間に一連の水平ラングがあるという観察に基づいています。 ドイツやヨーロッパの他の場所では、ページの上下に沿ってレールを水平に描画し、ラングを左から右に垂直に描画するスタイルがあります。

ラダー図とも呼ばれるラダーロジックのプログラムは、一連のリレー回路の回路図に似ています。 ラダーロジックは、類似しているため、追加のトレーニングをあまり行わなくても、さまざまなエンジニアや技術者が理解して使用できるため、便利です。

ラダーロジックは、プロセスまたは製造操作の順次制御が必要な PLC のプログラミングに広く使用されています。 ラダーロジックは、単純だが重要な制御システム、または古いハードワイヤードリレー回路の再加工に役立ちます。 プログラマブルロジックコントローラがより洗練されるにつれて、それは非常に複雑な自動化システムでも使用されてきました。

ラダーロジックは、手続き型言語ではなく、ルールベースの言語と考えることができます。 ラダーのラングはルールを表します。 リレーやその他の電気機械装置を使用して実装すると、さまざまなルールが同時に即座に実行されます。

プログラマブルロジックコントローラに実装されている場合、ルールは通常、ソフトウェアによってループ内で順番に実行されます。 ループを十分に速く、通常は 1 秒間に何度も実行することにより、同時かつ即時の実行の効果が得られます。

ラダーロジックは、これらの一般的な操作手順に従います。

- 入力の読み取り
- ロジックを解く
- 出力の更新

11.1.3 例

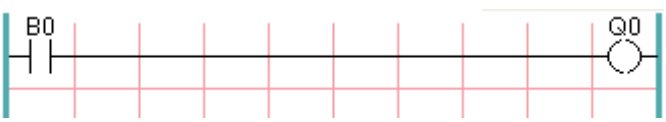
ラダーの最も一般的なコンポーネントは接点（入力）であり、これらは通常、NC（通常は閉じている）または NO（通常は開いている）、およびコイル（出力）です。

- NO 接点 
- NC 接点 
- コイル（出力） 

もちろん、完全なラダー言語にはさらに多くのコンポーネントがありますが、これらを理解すると、全体的な概念を理解するのに役立ちます。

ラダーは 1 つまたは複数のラングで構成されます。 これらのラングは水平トレース（ワイヤを表す）であり、コンポーネント（入力、出力など）があり、左から右に評価されます。

この例は最も単純なラングです。



左側の入力 B0（通常は開いている接点）は、右側のコイル（出力）Q0 に接続されています。 ここで、入力 B0 が真になるために電圧が左端に印加されると想像してください（たとえば、入力がアクティブ化されているか、

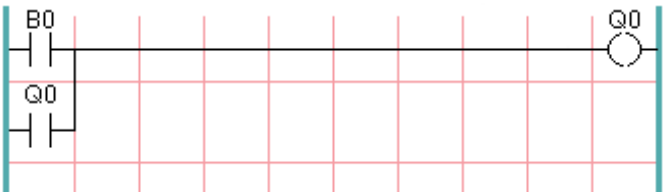
ユーザーが NO 接点を押した）。 電圧には、右側のコイル（出力）Q0 に到達するための直接的な経路があります。 結果として、Q0 コイル（出力）は 0 / off / false から 1 / on / true に変わります。 ユーザーが B0 を離すと、Q0 出力はすぐに 0 / off / false に戻ります。

11.1.4 基本的なラッチオンオフ回路

上記の例に基づいて、コイル Q0 がアクティブになるたびに閉じるスイッチを追加するとします。 これは、コイルがスイッチの接点をアクティブにできるリレーの場合です。 または、コンタクタの主な機能である大きな 3 相接点に加えて、いくつかの小さな補助接点が存在することが多いコンタクタ内。

この補助スイッチは、前の例ではコイル Q0 から駆動されるため、駆動するコイルと同じ番号を付けます。 これは、すべてのラダープログラミングで採用されている標準的な方法ですが、最初はコイルと同じラベルの付いたスイッチを見るのは奇妙に思えるかもしれません。 それでは、この補助接点を Q0 と呼び、前の例の B0 押しボタン接点に接続してみましょう。

それを見てみましょう：

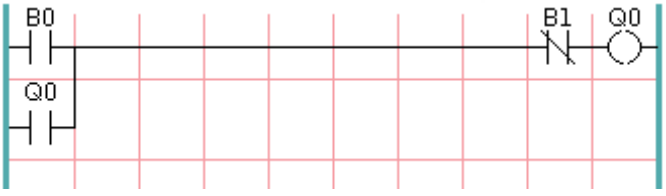


前と同じように、ユーザーが押しボタン B0 を押すと、コイル Q0 がオンになります。 そして、コイル Q0 がオンになると、スイッチ Q0 がオンになります。 今、興味深い部分が起こります。 ユーザーが押しボタン B0 を離しても、コイル Q0 は以前のように停止しません。 これは、この回路のスイッチ Q0 がユーザーの押しボタンを効果的に押し続けているためです。 したがって、開始押しボタンを離した後も、スイッチ Q0 がコイル Q0 をオンにしたままであることがわかります。

このように使用されるコイルまたはリレーのこのタイプの接点は、関連付けられているコイルを保持するため、保持接点と呼ばれることがよくあります。 シール接点と呼ばれることもあり、アクティブなときは回路がシールされていると言われます。

残念ながら、これまでのところ、私たちの回路はほとんど実用的ではありません。なぜなら、押しボタン B0 の形のオンまたは開始ボタンがありますが、一度開始するとこの回路を遮断する方法がないからです。 しかし、それは簡単に修正できます。 必要なのは、コイル Q0 への電力を遮断する方法だけです。 それでは、コイル Q0 の直前にノーマルクローズ（NC）押しボタンを追加しましょう。

これがどのように見えるかです：



これで、押しボタン B1 を追加または停止しました。 ユーザーが押すと、ラングからコイルへの接触が途切れます。 コイル Q0 が電力を失うと、0 / off / false に低下します。 コイル Q0 がオフになると、スイッチ Q0 もオフになるため、保持接点が破損するか、回路が開封されます。 ユーザーが停止押しボタンを離すと、接触はラングからコイル Q0 に復元されますが、ラングが停止しているため、コイルは元に戻りません。

この回路は、コンタクタによって制御される三相モーターを備えたほぼすべてのマシンで数十年にわたって使用されてきたため、ラダー/ PLC プログラマーによって採用されることは避けられませんでした。 また、スタートとストップの両方を同時に押すと、ストップ機能が常に優先されるという点で、非常に安全な回路です。

これは多くのラダープログラミングの基本的な構成要素であるため、これに慣れていない場合は、この回路がどのように動作するかを確実に理解することをお勧めします。

11.2 ラダーの概念

クラシックラダーは、もともと産業用 PLC に実装されているプログラミング言語の一種です（ラダープログラミングと呼ばれます）。 これは、リレー接点とコイルの概念に基づいており、多くのシステムインテグレーターに馴染みのある方法で論理チェックと機能を構築するために使用できます。 はしごは、分岐があり、電気回路に似ているラングで構成されています。 実行時にラダープログラムがどのように評価されるかを知ることは重要です。

各行が左から右に評価され、次に次の行が下に評価されるのは当然のようですが、ラダーロジックではこのようには機能しません。 ラダーロジックは、ラダーラングを 3 回スキャンして、出力の状態を変更します。

- 入力が読み取られ、更新されます
- 論理が理解されている
- 出力が設定されます

これは、ある行の出力が別のラングの入力によって読み取られる場合、最初は混乱する可能性があります。 出力が設定された後、2 番目の入力が true になる前に 1 回のスキャンがあります。

ラダープログラミングのもう 1 つの落とし穴は、「LastOneWins」ルールです。 ラダーの異なる場所に同じ出力がある場合、最後の出力の状態が出力の設定になります。

11.3 言語

クラシックラダーを使用するときに使用される最も一般的な言語はラダーです。 クラシックラダーは、シーケンシャルファンクションチャート（Grafcet）もサポートしています。

11.4 コンポーネント

クラシックラダーには2つのコンポーネントがあります。

- リアルタイムモジュール classicladder_rt
- ユーザースペースモジュール（GUI を含む）classicladder

11.4.1 ファイル

Stepconf で生成された構成から作業する場合、通常、従来のラダーコンポーネントは custom.hal ファイルに配置されます。 これらは custom_postgui.hal ファイルに配置しないでください。配置しないと、ラダーエディタメニューがグレー表示されます。

Note

ラダーファイル（.clp）には、名前に空白を含めることはできません。

11.4.2 リアルタイムモジュール

クラシックラダーリアルタイムモジュール（classicladder_rt）のロードは、HAL ファイルから、または halcmd 命令を使用して直接行うことができます。 最初の行は、ClassicLadder モジュールをリアルタイムでロードします。 2行目は、関数 classicladder.0.refresh をサーボスレッドに追加します。 この行は、クラシックラダーをサーボスレッドレートで更新します。

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

クラシックラダーが実行されているスレッドの速度は、入力と出力への応答性に直接影響します。 Classic Ladder が気付くよりも速くスイッチのオンとオフを切り替えることができる場合は、スレッドを高速化する必要があります。 クラシックラダーがラングを更新できる最速は1ミリ秒です。 あなたはそれをより速いスレッドに入れることができますが、それはそれ以上速く更新されません。 1ミリ秒より遅いスレッドに配置すると、ClassicLadder はラングの更新を遅くします。 現在のスキャン時間はセクションディスプレイに表示され、マイクロ秒に丸められます。 スキャン時間が1ミリ秒より長い場合は、ラダーを短くするか、低速のスレッドに配置することをお勧めします。

11.4.3 変数

クラシックラダーリアルタイムモジュールのロード中に、各タイプのラダーオブジェクトの数を構成することができます。 ラダーオブジェクトの数を設定しない場合、クラシックラダーはデフォルト値を使用します。

表 13.1：デフォルトの変数数

オブジェクト名	変数名	デフォルト値
ラングの数	(numRungs)	100
ビット数	(numBits)	20
単語変数の数	(numWords)	20
タイマーの数	(numTimers)	10
タイマーの数 IEC	(numTimersIec)	10
単安定マルチバイブの数	(numMonostables)	10
カウンターの数	(numCounters)	10
HAL 入力ビットピンの数	(numPhysInputs)	15
HAL 出力ビットピンの数	(numPhysOutputs)	15
算術式の数	(numArithmExpr)	50
セクション数	(numSections)	10
シンボルの数	(numSymbols)	Auto
S32 入力の数	(numS32in)	10
S32 出力の数	(numS32out)	10
浮動小数点入力の数	(numFloatIn)	10
浮動小数点出力の数	(numFloatOut)	10

最も重要なオブジェクトは、numPhysInputs、numPhysOutputs、numS32in、および numS32out です。

これらの数を変更すると、使用可能な HAL ビットピンの数が変更されます。 numPhysInputs および numPhysOutputs は、使用可能な HAL ビット（オン/オフ）ピンの数を制御します。 numS32in および numS32out は、使用可能な HAL 符号付き整数（+-整数範囲）ピンの数を制御します。

たとえば（いくつか変更するためにこれらすべてを変更する必要はありません）：

```
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
```

デフォルト数のオブジェクトをロードするには：

```
loadrt classicladder_rt
```

11.5 クラシックラダーユーザーモジュールのロード

クラシックラダー HAL コマンドは、GUI がロードされる前に実行する必要があります。そうしないと、メニュー項目のラダーエディターが機能しません。 Stepper Config Wizard を使用した場合は、Classic LadderHAL コマンドを custom.hal ファイルに配置します。

ユーザーモジュールをロードするには：

```
loadusr classicladder
```

note

ロードできる.clp ファイルは 1 つだけです。 ラダーを分割する必要がある場合は、セクションを使用してください。

ラダーファイルをロードするには：

```
loadusr classicladder myladder.clp
```

クラシックラダーローディングオプション

- --nogui-（ラダーエディタなしでロード）通常、デバッグの終了後に使用されます。
- --modbus_port = port-（modbus ポート番号をロードします）
- --modmaster-（MODBUS マスターを初期化します）は、ラダープログラムを同時にロードする必要があります。そうでない場合、TCP がデフォルトのポートになります。
- --modslave-（MODBUS スレーブを初期化します）TCP のみ

EMC なしで HAL でクラシックラダーを使用するには：

```
loadusr -w classicladder
```

-w は、ClassicLadder が終了するまで HAL 環境を閉じないように HAL に指示します。

--nogui オプションを使用して最初にラダープログラムをロードしてから、オプションなしでクラシックラダーを再度ロードすると、GUI は最後にロードされたラダープログラムを表示します。

AXIS では、*ファイル/ラダーエディタ*から GUI をロードできます。。。

11.6 クラシックラダー GUI

GUI を使用して ClassicLadder をロードすると、セクション表示とセクションマネージャーの 2 つのウィンドウが表示されます。

11.6.1 セクションマネージャー

Classic Ladder を最初に起動すると、空の SectionsManager ウィンドウが表示されます。

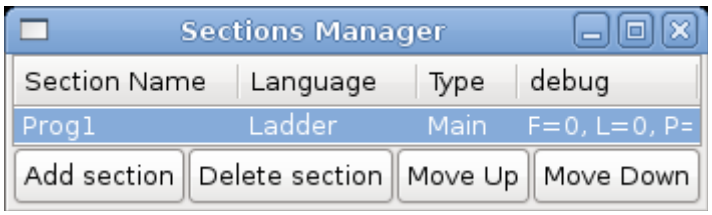


図 13-76

このウィンドウでは、セクションに名前を付けたり、作成または削除したり、そのセクションで使用する言語を選択したりできます。 これは、コールコイルのサブルーチンに名前を付ける方法でもあります。

11.6.2 セクション表示

クラシックラダーを最初に起動すると、空のセクション表示ウィンドウが表示されます。 空のラングが 1 つ表示されます。

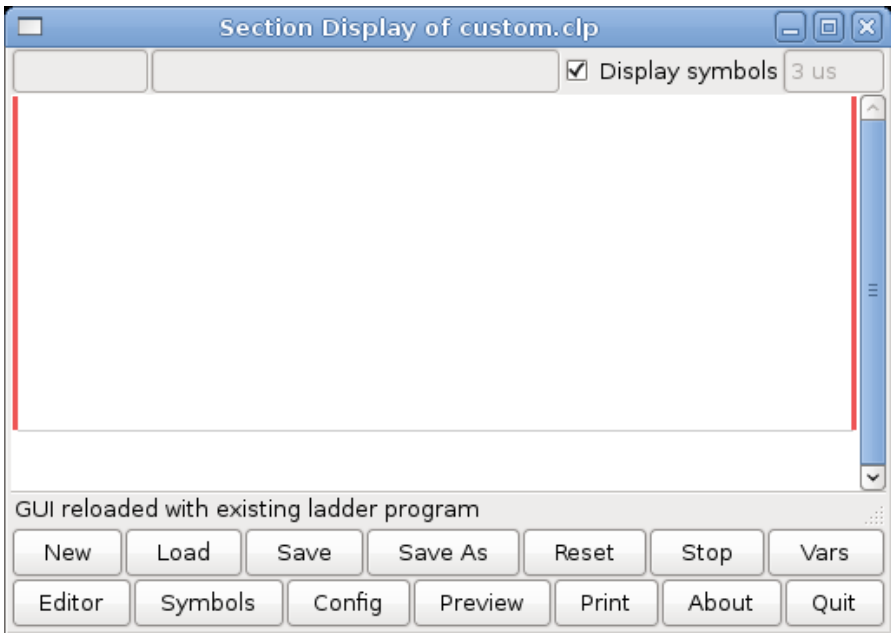


図 13-77

ほとんどのボタンは自明です：

[変数]ボタンは変数を表示するためのもので、切り替えて一方、他方、両方を表示し、ウィンドウを表示しません。

Config ボタンは modbus に使用され、リアルタイムモジュールでロードされたラダー要素の最大数を表示します。 [シンボル]ボタンは、変数のシンボルの編集可能なリストを表示します（ヒントには、入力、出力、コイルなどに名前を付けることができます）。 [終了]ボタンは、Modbus とディスプレイを意味するユーザープログラムをシャットダウンします。 リアルタイムラダープログラムは引き続きバックグラウンドで実行されます。

右上のチェックボックスでは、変数名とシンボル名のどちらを表示するかを選択できます

ラダープログラムの表示の下に、「プロジェクトを読み込めませんでした。..」という行があることに気付くかもしれません。これは、表示ウィンドウでクリックしたラダープログラムの要素に関する情報を提供するステータスバーです。 このステータス行には、変数%I、%Q、および最初の%W（方程式内）の HAL 信号名が表示されます。ラングに（103）などの面白いラベルが表示される場合があります。 これは（意図的に）古いバグのために表示されます。要素を消去すると、古いバージョンでは正しいコードでオブジェクトが消去されないことがありました。 古いバージョンでは、長い水平接続ボタンが機能しない場合があることに気付いたかもしれません。これは、無料のコードを探したが、何か他のものを見つけたためです。 括弧内の数字は認識されないコードです。 ラダープログラムは引き続き正常に動作し、エディタでコードを消去してプログラムを保存するように修正します。

11.6.3 可変ウィンドウ

これは、ビットステータスウィンドウ（ブール値）とウォッチウィンドウ（符号付き整数）の 2 つの可変ウィンドウです。 Vars ボタンは SectionDisplay Window にあり、Vars ボタンを切り替えて一方、他方、両方を表示し、その後変数ウィンドウを表示しません。

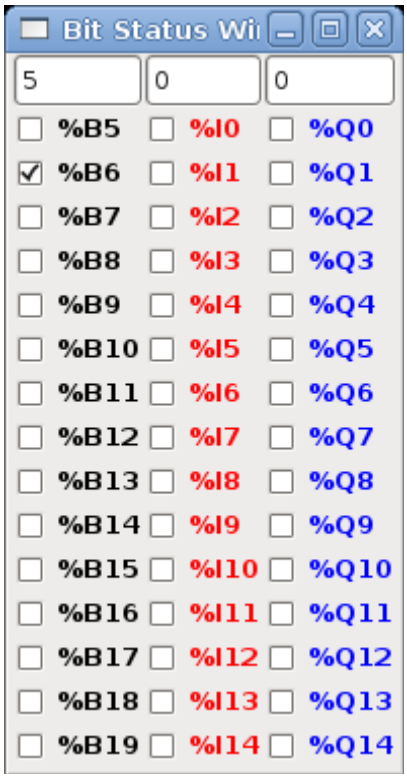


図 13-78

ビットステータスウィンドウには、ブール（オン/オフ）変数データの一部が表示されます。 すべての変数が%記号で始まることに注意してください。 %I 変数は、HAL 入力ビットピンを表します。 %Q は、リレーコイルと HAL 出力ビットピンを表します。 %B は、内部リレーコイルまたは内部接点を表します。 上部の 3 つの編集領域では、各列に表示される 15 個の変数を選択できます。 たとえば、%B 変数列の高さが 15 エントリで、列の上部に 5 を入力した場合、変数%B5 から%B19 が表示されます。 チェックボックスを使用すると、ラダープログラムが出力として設定していない限り、%B 変数を手動で設定および設定解除できます。 クラシックラダーの実行中にプログラムによって出力として設定されたビットは変更できず、オンの場合はオン、オフの場合はオフとして表示されます。

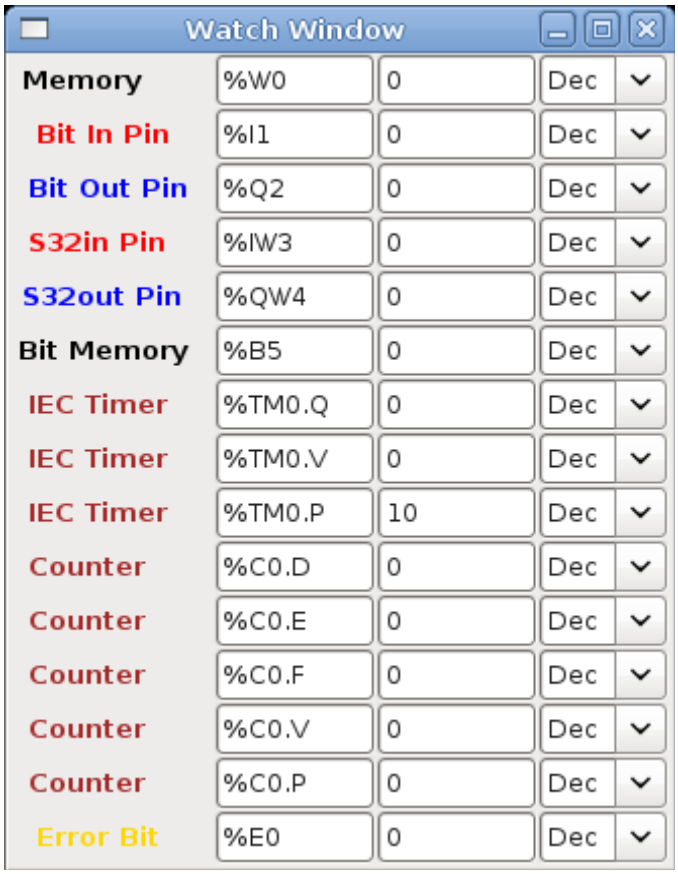


図 13-79

ウォッチウィンドウに可変ステータスが表示されます。 その横にある編集ボックスは、変数に格納されている数値であり、横にあるドロップダウンボックスでは、数値を 16 進数、10 進数、または 2 進数のいずれで表示するかを選択できます。 表示されている単語変数のシンボルウィンドウでシンボル名が定義されていて、セクション表示ウィンドウで[シンボルの表示]チェックボックスがオンになっている場合、シンボル名が表示されます。表示される変数を変更するには、変数番号を入力します。例： %W2（[シンボルの表示]チェックボックスがオフの場合）または既存の変数番号/名前の上にシンボル名（[シンボルの表示]チェックボックスがオンの場合）を入力して、Enter キーを押します。

11.6.4 シンボルウィンドウ

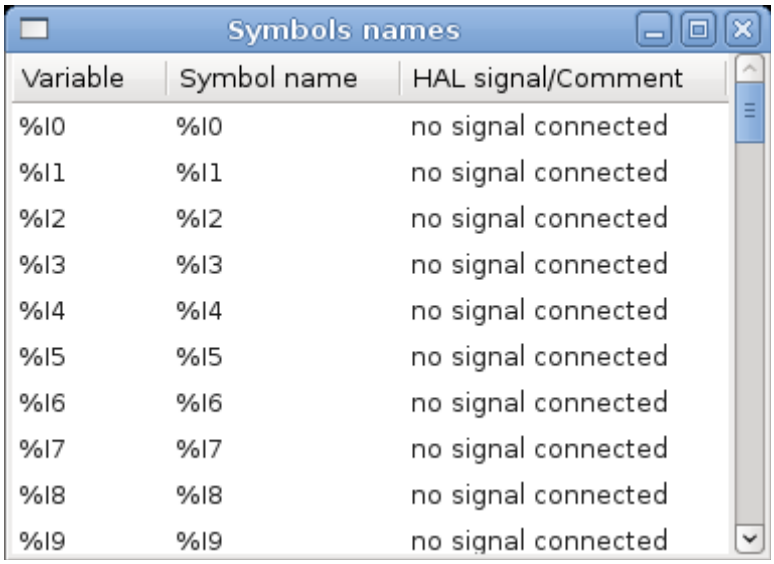


図 13-80

これは、[シンボルの表示]チェックボックスがオンになっているときにセクションウィンドウに表示される変数名の代わりに使用するシンボル名のリストです。 変数名（%記号と大文字を覚えておいてください）、記号名を追加します。 変数に HAL 信号を接続できる場合（%I、%Q、および%W-リアルタイムモジュールで s32 ピンをロードした場合）、コメントセクションに現在の HAL 信号名またはその欠如が表示されます。 表示しやすくするために、シンボル名は短くする必要があります。 セクションウィンドウで%I、%Q、および%W 変数の長い HAL 信号名をクリックすると、それらをクリックして表示できることに注意してください。 2 つの間で、ラダープログラムが何に接続されているかを追跡できるはずです。

11.6.5 エディターウィンドウ

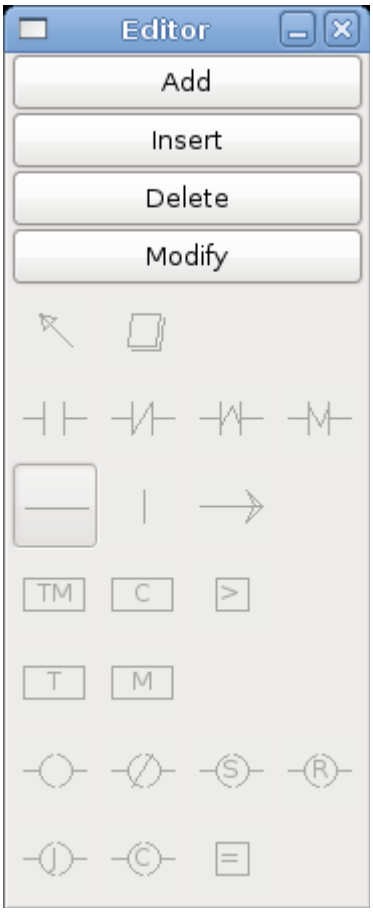


図 13-81

- 追加-選択したラングの後にラングを追加します
- 挿入-選択したラングの前にラングを挿入します
- 削除-選択したラングを削除します
- 変更-選択したラングを編集用に開きます

左上の画像から開始：

- オブジェクトセレクター、消しゴム
- N.O. 入力、N.C。入力、立ち上がりエッジ入力、立ち下がりエッジ入力
- 水平接続、垂直接続、長い水平接続
- タイマー IEC ブロック、カウンターブロック、変数の比較
- 古いタイマーブロック、古い単安定ブロック（これらは IEC タイマーに置き換えられました）
- コイル-N.O. 出力、N.C。出力、出力の設定、出力のリセット
- ジャンプコイル、コールコイル、変数の割り当て

各ボタンの簡単な説明：

- セレクター-既存のオブジェクトを選択して情報を変更できます。
- 消しゴム-オブジェクトを消去します。
- N.O. 連絡先-通常開の連絡先を作成します。 これは、外部 HAL ピン（%I）入力接点、内部ビットコイル（%B）接点、または外部コイル（%Q）接点にすることができます。 HAL ピンが真の場合、HAL ピン入力接点は閉じます。 対応するコイルがアクティブになるとコイル接点が閉じます（%Q2 コイルがアクティブになると%Q2 接点が閉じます）。
- N.C.連絡先-通常は閉じている連絡先を作成します。 N.O.と同じです。 HAL ピンが真であるかコイルがアクティブであるときに接点が開いていることを除いて、接点。
- '立ち上がりエッジ接点-HAL ピンが False から true になるか、コイルが非アクティブからアクティブになると閉じる接点を作成します。
- 立ち下がりエッジ接点-HAL ピンが true から false に移行したとき、またはコイルがアクティブから非アクティブに移行したときに閉じられる接点を作成します。
- 水平接続-オブジェクトへの水平接続を作成します。
- 垂直接続-水平線への垂直接続を作成します。
- 水平方向の実行接続-2 つのオブジェクト間に水平方向の接続を作成し、複数のブロックが離れているオブジェクトをすばやく接続する方法です。
- IEC タイマー-タイマーを作成し、タイマーを置き換えます。
- タイマー-タイマーモジュールを作成します（減価償却は代わりに IEC タイマーを使用します）。
- 単安定-ワンショット単安定モジュールを作成します
- Counter-カウンターモジュールを作成します。
- 比較-変数を値または他の変数と比較するための比較ブロックを作成します。 （例：%W1 <= 5 または%W1 =%W2）比較は、セクション表示の右端に配置できません。
- 変数の割り当て-変数に値を割り当てることができるように、割り当てブロックを作成します。 （例：%W2 = 7 または%W1 =%W2）ASSIGNMENT 関数は、セクション表示の右端にのみ配置できます。

11.6.6 設定ウィンドウ

設定ウィンドウには現在のプロジェクトステータスが表示され、Modbus 設定タブがあります。

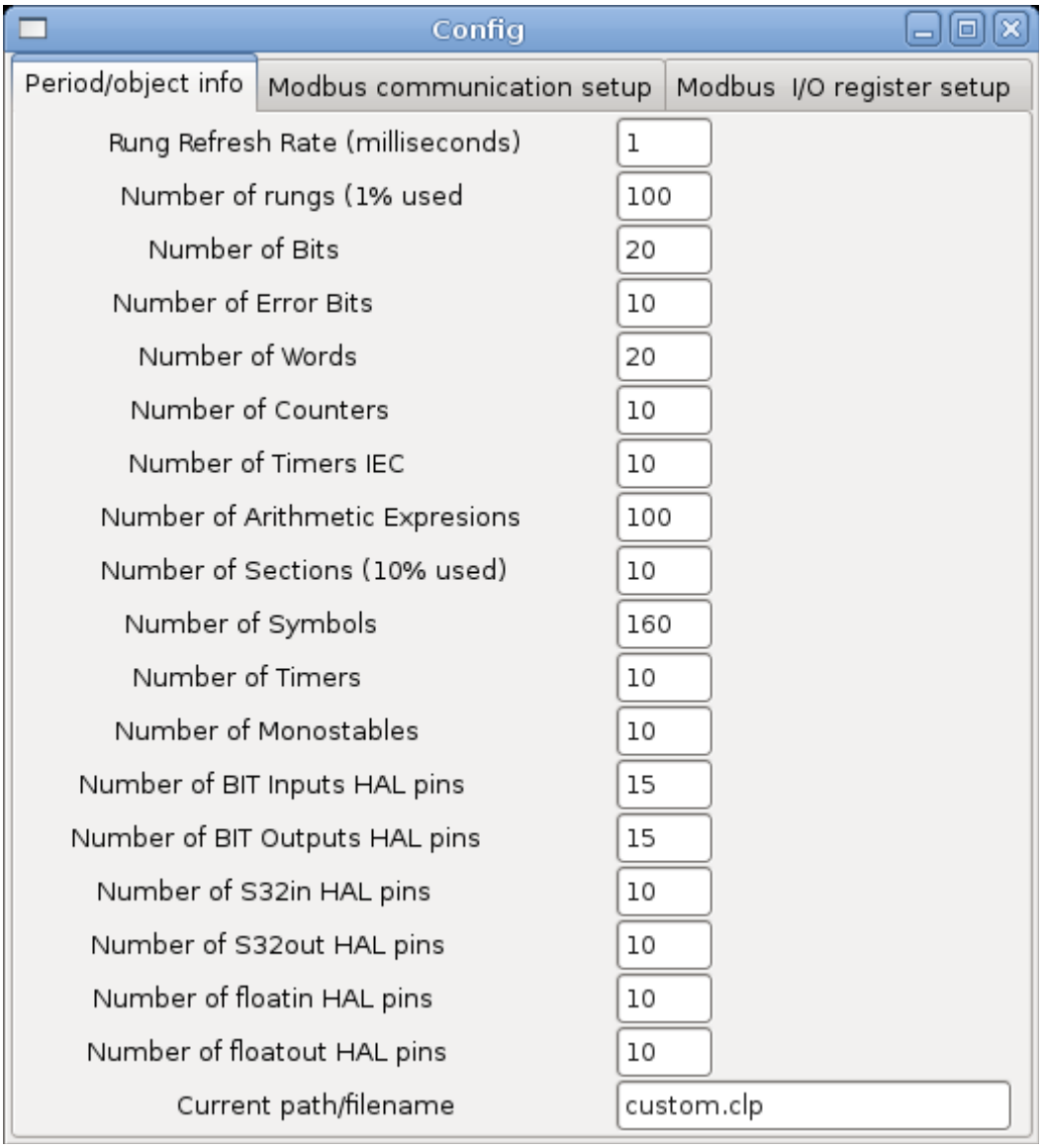


図 13-82

11.7 ラダーオブジェクト

11.7.1 連絡先

スイッチまたはリレー接点を表します。 それらは、それらに割り当てられた可変の文字と数字によって制御されます。

可変文字は B、I、または Q にすることができ、数字は最大 3 桁の数字にすることができます。 %I2、%Q3、または%B123。 変数 I は、対応する番号の HAL 入力ピンによって制御されます。 変数 B は内部接点用で、対応する番号の B コイルによって制御されます。 変数 Q は、対応する番号の Q コイルによって制御されます。 （複数の接点を持つリレーのように）。 例えば。 HAL ピン classicladder.0.in-00 が true の場合、%I0 N.O. 連絡先はオンになります（閉じた、本当、あなたがそれを呼びたいものは何でも）。 %B7 コイルが通電されている場合（オン、真など）、%B7 N.O. 連絡先がオンになります。 %Q1 コイルが通電されている場合、%Q1 N.O. 連絡先がオンになります（HAL ピン classicladder.0.out-01 が真になります）。

- N.O. 連絡先-  （通常開）変数が false の場合、スイッチはオフになっています。
- N.C.連絡先-  （通常は閉じています）変数が false の場合、スイッチはオンになっています。
- Rising Edge Contact-変数が false から true に変わると、スイッチは PULSED オンになります。
- 立ち下がりエッジ接触-変数が true から false に変わると、スイッチは PULSED オンになります。

11.7.2 IEC タイマー

新しいカウントダウンタイマーを表します。 IEC タイマーは、タイマーと単安定マルチバイブに取って代わります。

IEC タイマーには 2 つの接点があります。

- I-入力接点
- Q-出力接点

TON、TOF、TP の 3 つのモードがあります。

- TON-タイマー入力が true の場合、カウントダウンが開始され、入力が true である限り続行されます。 カウントダウンが行われた後、タイマー入力がまだ true である限り、出力は true になります。
- TOF-タイマー入力が true の場合、出力を true に設定します。 入力が false の場合、タイマーはカウントダウンし、出力を false に設定します。
- TP-タイマー入力が true にパルスされるか、true に保持されると、タイマーはタイマーがカウントダウンするまで出力を true に設定します。 （ワンショット）

時間間隔は、100 ミリ秒、秒、または分の倍数で設定できます。

比較または操作ブロックで読み取りおよび/または書き込みが可能な IEC タイマー用の変数もあります。

- %TMxxx.Q-タイマー完了（ブール値、読み取り/書き込み）
- %TMxxx.P-タイマープリセット（読み取り/書き込み）
- %TMxxx.V-タイマー値（読み取り/書き込み）

11.7.3 タイマー

カウントダウンタイマーを表します。 これは非推奨になり、IEC タイマーに置き換えられました。

タイマーには 4 つの接点があります。

- E-イネーブル（入力）は、true の場合にタイマーを開始し、false の場合にリセットします

- C-タイマーを実行するには、制御（入力）がオンになっている必要があります（通常は E に接続します）
- D-タイマーがタイムアウトしたとき、E が true のままである限り、done（出力）true
- R-実行中（出力）タイマーが実行中の場合は true

タイマーベースは、ミリ秒、秒、または分の倍数にすることができます。

比較または操作ブロックで読み取りおよび/または書き込みが可能なタイマー用の変数もあります。

- %Txx.R-実行中のタイマー xx（ブール値、読み取り専用）
- %Txx.D-タイマー xx が完了しました（ブール値、読み取り専用）
- %Txx.V-タイマー xx の現在の値（整数、読み取り専用）
- %Txx.P-タイマー xx プリセット（整数、読み取りまたは書き込み）

11.7.4 単安定マルチバイブ

元のワンショットタイマーを表します。 これは非推奨になり、IEC タイマーに置き換えられました。

単安定マルチバイブには、I と R の 2 つの接点があります。

- I-入力（入力）はモノラルタイマーの実行を開始します。
- R-実行中（出力）は、タイマーの実行中は true になります。

I 接点は立ち上がりエッジに敏感です。つまり、false から true（または off から on）に変更されたときにのみタイマーが開始されます。 タイマーの実行中、I の連絡先は、実行中のタイマーに影響を与えることなく変更できます。 R は真になり、タイマーがゼロまでカウントを終了するまで真のままになります。 タイマーベースは、ミリ秒、秒、または分の倍数にすることができます。

比較または操作ブロックで読み取りおよび/または書き込みが可能な単安定マルチバイブの変数もあります。

- %Mxx.R-単安定 xx 実行中（ブール値、読み取り専用）
- %Mxx.V-単安定 xx 現在の値（整数、読み取り専用）
- %Mxx.P-単安定 xx プリセット（整数、読み取りまたは書き込み）

11.7.5 カウンター

アップ/ダウンカウンターを表します。

7 つの連絡先があります：

- R-リセット（入力）はカウントを 0 にリセットします。
- U-カウントアップ（入力）は、カウントに 1 を追加します。

- D-ダウンカウント（入力）は、カウントから 1 を減算します。
- E-カウントが 0 から 9999 にロールオーバーすると、アンダーフロー（出力）が真になります。
- D-カウントがプリセットと等しい場合、done（出力）は true になります。
- F-カウントが 9999 から 0 にロールオーバーすると、オーバーフロー（出力）が真になります。

アップカウントとダウンカウントの連絡先はエッジに依存します。つまり、連絡先が false から true に変更された場合（または必要に応じてオフからオンに変更された場合）にのみカウントされます。

範囲は 0〜9999 です。

比較または操作ブロックで読み取りおよび/または書き込みが可能なカウンター用の変数もあります。

- %Cxx.D-カウンター xx が完了しました（ブール値、読み取り専用）
- %Cxx.E-カウンター xx の空のオーバーフロー（ブール値、読み取り専用）
- %Cxx.F-カウンター xx フルオーバーフロー（ブール値、読み取り専用）
- %Cxx.V-カウンター xx の現在の値（整数、読み取り、または書き込み）
- %Cxx.P-カウンター xx プリセット（整数、読み取りまたは書き込み）

11.7.6 比較

算術比較用。 変数%XXX =この数値（または評価された数値）になりますか

比較が真の場合、比較ブロックは真になります。 ほとんどの数学記号を使用できます。

- +、-、*、/、=（標準の数学記号）
- <（より小さい）、>（より大きい）、<=（以下）、> =（以上）、<>（等しくない）
- （、）グループに分ける例例%IF1 = 2、＆%IF2 <5 の擬似コードは、%IF1 が 2 に等しく、%IF2 が 5 未満の場合、比較は真になります。 比較の 2 つのグループを区切るコンマに注意してください。
- ^（指数）、%（モジュラス）、＆（および）、|（また）、。 -
- ABS（絶対）、MOY（平均フランス語）、AVG（平均）

たとえば、ABS（%W2）= 1、MOY（%W1、%W2）<3 です。

比較式ではスペースを使用できません。 たとえば、%C0.V>%C0.P は有効な比較式ですが、%C0.V>%CO.P は有効な式ではありません。

ラダーオブジェクトの読み取りと書き込みに使用できる変数のリストがページの下にあります。 新しい比較ブロックを開くときは、比較を入力するときに必ず＃記号を削除してください。

ワード変数＃1 がカウンター＃0 の現在の値の 2 倍未満であるかどうかを確認するには、構文は次のようになります。

```
%W1<2*%C0.V
```

S32in ビット 2 が 10 に等しいかどうかを確認するには、構文は次のようになります。

```
%IW2=10
```

注：Compare は、プログラマーが慣れている doubleequals ではなく算術 equals を使用します。

11.7.7 変数の割り当て

変数の割り当ての場合、例： この数値（または評価された数値）をこの変数%xxx に割り当てます。変数の最大値（0x80000000）と最小値（0x07FFFFFF）（符号付きの値を考えてください）をチェックし、それらが超えないようにする 2 つの数学関数 MINI と MAXI があります。

新しい変数割り当てブロックを開くときは、割り当てを入力するときに必ず＃記号を削除してください。

IEC タイマー 0 のタイマープリセットに値 10 を割り当てるには、構文は次のようになります。

```
%TM0.P=10
```

12 の値を s32out ビット 3 に割り当てるための構文は、次のようになります。

```
%QW3=12
```

Note

変数割り当てブロックを使用して変数に値を割り当てると、変数割り当てブロックを使用して新しい値を割り当てるまで、値は保持されます。 LinuxCNC の起動時に、最後に割り当てられた値が復元されます。

次の図は、割り当てと比較の例を示しています。 %QW0 は S32out ビットであり、%IW0 は S32in ビットです。 この場合、HAL ピン classicladder.0.s32out-00 は値 5 に設定され、HAL ピン classicladder.0.s32in-00 が 0 の場合、HAL ピン classicladder.0.out-00 は True に設定されます。 。

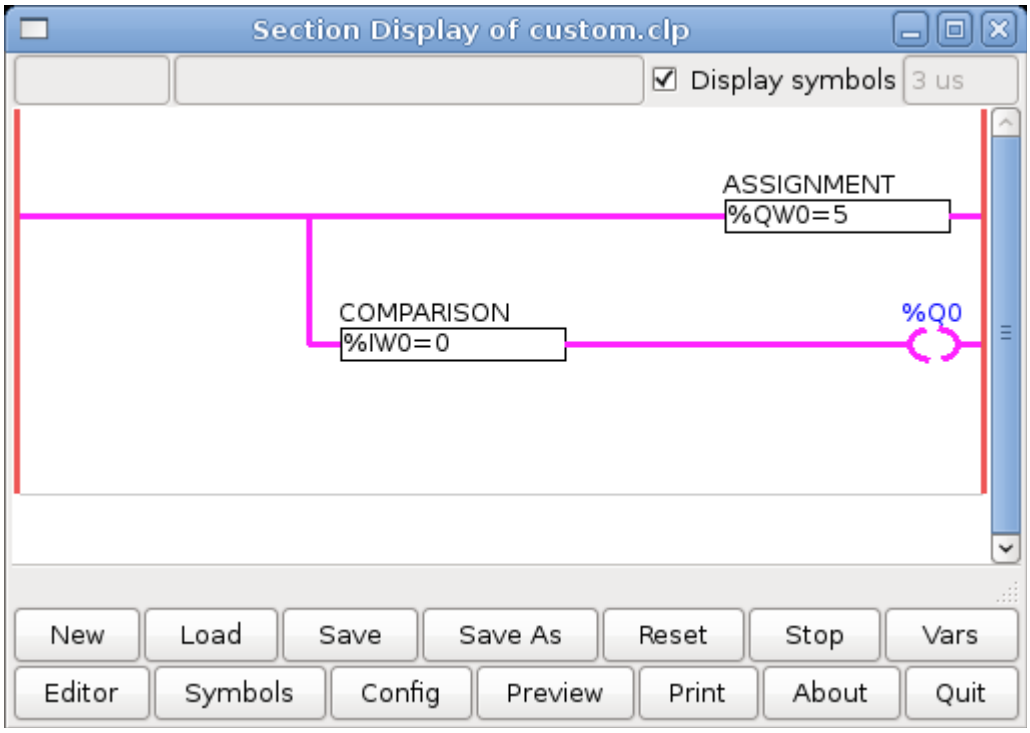

Section Display of custom.clp
Display symbols
3 us
ASSIGNMENT
%QW0=5
COMPARISON
%IW0=0
%Q0
New
Load
Save
Save As
Reset
Stop
Vars
Editor
Symbols
Config
Preview
Print
About
Quit

図 13-83


Properties
Expression
%QW0=5

Apply

図 13-84


Properties
Expression
%IW0=0

Apply

11.7.8 コイル

コイルはリレーコイルを表します。 それらは、それらに割り当てられた可変の文字と数字によって制御されます。

可変文字はBまたはQにすることができ、数字は最大3桁の数字にすることができます。 %Q3、または%B123。 Qコイルは HAL 出力ピンを制御します。 %Q15 がオンになっている場合、HAL ピン classicladder.0.out-15 が真になります。 Bコイルは、プログラムフローを制御するために使用される内部コイルです。

- N.O. COIL-（リレーコイル。）コイルに通電すると、N.O。 連絡先は閉じられます（on、true など）
- N.C. COIL-（接点を反転させるリレーコイル。）コイルに通電すると、N.O。接点が開きます（オフ、偽など）。
- SET COIL-（ラッチ接点付きのリレーコイル）コイルに通電すると、N.O。 接点はラッチで閉じられます。
- RESET COIL-（ラッチ接点付きのリレーコイル）コイルに通電すると N.0 になります。 接点はラッチで開きます。
- ジャンプコイル-（後藤コイル）コイルが通電されると、ラダープログラムはラングにジャンプします（現在のセクション）-ジャンプポイントはラングラベルで示されます。 （セクション表示、左上のラベルボックスにラングラベルを追加します）
- CALL COIL-（gosub コイル）コイルが通電されると、プログラムはサブルーチン番号で指定されたサブルーチンセクションにジャンプします-サブルーチンは SR0 から SR9 で指定されます（セクションマネージャーで指定します）

警告

N.C.コイルと N.C.接点を使用する場合、ロジックは機能します（コイルに通電すると、接点が閉じます）が、それを追跡するのは非常に困難です。

ジャンプコイルジャンプコイルは、BASIC プログラミング言語の goto のように、別のセクションにジャンプするために使用されます。

セクション表示ウィンドウの左上を見ると、小さなラベルボックスとその横に長いコメントボックスがあります。 次に、Editor！Modify に移動し、小さなボックスに戻って名前を入力します。

先に進み、コメントセクションにコメントを追加します。 このラベル名はこのラングの名前のみであり、ジャンプコイルがどこに行くかを識別するために使用されます。

ジャンプコイルを配置するときは、右端の位置に追加し、ジャンプしたいラングにラベルを変更します。 CALL COIL CALL COIL は、BASIC プログラミング言語の gosub のように、サブルーチンセクションに移動してから戻るために使用されます。 セクションマネージャウィンドウに移動する場合は、[セクションの追加]ボタン

をクリックします。 このセクションに名前を付け、使用する言語（ラダーまたはシーケンシャル）を選択し、タイプ（メインまたはサブルーチン）を選択できます。

サブルーチン番号（SR0 など）を選択します。 空のセクションが表示され、サブルーチンを作成できます。 それが終わったら、セクションマネージャーに戻り、メインセクション（デフォルト名は prog1）をクリックします。

これで、CALLCOIL をプログラムに追加できます。 コールコイルはラングの右端の位置に配置します。 ラベルを前に選択したサブルーチン番号に変更することを忘れないでください。

11.8 クラシックラダー変数

これらの変数は、COMPARE または OPERATE で使用され、カウンタープリセットの変更や、タイマーの実行が完了したかどうかの確認など、ラダーオブジェクトに関する情報を取得したり、仕様を変更したりします。

変数のリスト：

- %Bxxx-ビットメモリ xxx（ブール値）
- %Wxxx-ワードメモリ xxx（32 ビット符号付き整数）
- %IWxxx-ワードメモリ xxx（ピンの S32）
- %QWxxx-ワードメモリ xxx（S32 出力ピン）
- %IFxx-ワードメモリ xx（ピンにフロート）（クラシックラダーで S32 に変換）
- %QFxx-ワードメモリ xx（フロート出力ピン）（クラシックラダーで S32 に変換）
- %Txx.R-実行中のタイマー xx（ブール値、ユーザー読み取り専用）
- %Txx.D-タイマー xx が完了しました（ブール値、ユーザー読み取り専用）
- %Txx.V-タイマー xx の現在の値（整数、ユーザー読み取り専用）
- %Txx.P-タイマー xx プリセット（整数）
- %TMxxx.Q-タイマー xxx が完了しました（ブール値、読み取り/書き込み）
- %TMxxx.P-タイマー xxx プリセット（整数、読み取り/書き込み）
- %TMxxx.V-タイマー xxx 値（整数、読み取り/書き込み）
- %Mxx.R-単安定 xx 実行中（ブール値）
- %Mxx.V-単安定 xx の現在の値（整数、ユーザー読み取り専用）
- %Mxx.P-単安定 xx プリセット（整数）
- %Cxx.D-カウンター xx が完了しました（ブール値、ユーザー読み取り専用）

- %Cxx.E-カウンター xx の空のオーバーフロー（ブール値、ユーザー読み取り専用）
- %Cxx.F-カウンター xx フルオーバーフロー（ブール値、ユーザー読み取り専用）
- %Cxx.V-カウンター xx の現在の値（整数）
- %Cxx.P-カウンター xx プリセット（整数）
- %Ixxx-物理入力 xxx（ブール値）（HAL 入力ビット）
- %Qxxx-物理出力 xxx（ブール値）（HAL 出力ビット）
- %Xxxx-ステップ xxx のアクティビティ（順次言語）
- %Xxxx.V-ステップ xxx の秒単位のアクティビティ時間（順次言語）
- %Exx-エラー（ブール値、読み取り/書き込み（上書きされます））
- インデックス付きまたはベクトル化された変数-これらは、別の変数によってインデックス付けされた変数です。 これをベクトル化された変数と呼ぶ人もいます。 例：%W0 [%W4] =>%W4 が 23 に等しい場合、%W23 に対応します

11.9 GRAFCET（ステートマシン）プログラミング

注意

これはおそらく、クラシックラダーの最も使用されておらず、最もよく理解されていない機能です。 シーケンシャルプログラミングは、一連のラダーイベントが常に所定の順序で発生するようにするために使用されます。 シーケンシャルプログラムは単独では機能しません。 変数を制御するラダープログラムも常にあります。 シーケンシャルプログラムを管理する基本的なルールは次のとおりです。

- ルール 1：初期状況-初期状況は、操作の開始時に定義上アクティブ状態にある初期ステップによって特徴付けられます。少なくとも 1 つの初期ステップが必要です。
- ルール 2：R2、トランジションのクリア-トランジションは有効または無効になっています。 対応する遷移シンボルにリンクされている直前のすべてのステップがアクティブな場合に有効になると言われ、それ以外の場合は無効になります。 トランジションは、有効になっておらず、関連するトランジション条件が true でない限り、クリアできません。
- ルール 3：R3、アクティブなステップの進化-遷移をクリアすると、直後のステップのアクティブ状態と直前のステップの非アクティブ状態が同時に発生します。
- ルール 4：R4、遷移の同時クリア-すべての同時クリアされた遷移は同時にクリアされます。

- ルール5：R5、ステップの同時アクティブ化と非アクティブ化-操作中にステップが同時にアクティブ化と非アクティブ化される場合、アクティブ化が優先されます。

これは、左上の画像から始まるSEQUENTIALエディターウィンドウです：セレクター矢印、消しゴム通常のステップ、初期（開始）ステップ遷移、ステップおよび遷移遷移リンク-下側、遷移リンク-上側パススルーリンク-下側、パススルーリンク-上側 ジャンプリンクコメントボックス[シーケンシャルプログラムを表示]

- ORDINARYSTEP-それぞれに固有の番号があります
- 開始ステップ-シーケンシャルプログラムには1つ必要です。 ここからプログラムが始まります。
- TRANSITION-これは、制御が次のステップに渡されるために真でなければならない変数を示しています。
- ステップとトランジション-利便性のために組み合わせる
- TRANSITION LINK-DOWNSIDE-次のステップのどちらが最初に真であるかに基づいて、ロジックフローを2つの可能なラインの1つに分割します（ORロジックを考えてください）
- TRANSITION LINK = UPSIDE-2つの（OR）ロジックラインを1つに結合します
- パススルーリンク-ダウンサイド-ロジックフローを2行に分割します。続行するには、両方が真である必要があります（ANDロジックを考えてください）。
- PASS-THROUGH LINK-UPSIDE-2つの同時（ANDロジック）ロジックラインを結合して元に戻します
- ジャンプリンク-最後のステップを最初のステップに接続するなど、互いに下にないステップを接続します
- コメントボックス-コメントを追加するために使用されます

リンクを使用するには、ステップがすでに配置されている必要があります。 リンクのタイプを選択してから、2つのステップまたはトランザクションを一度に1つずつ選択します。 練習が必要です！

シーケンシャルプログラミングの場合：変数%Xxxx（例：%X5）は、ステップがアクティブかどうかを確認するために使用されます。 変数%Xxxx.V（例：%X5.V）は、ステップがアクティブであった期間を確認するために使用されます。 %X変数と%X.v変数は、ラダーロジックで使用されます。 遷移に割り当てられた変数（例：%B）は、ロジックが次のステップに渡されるかどうかを制御します。 ステップがアクティブになった後、それをアクティブにした遷移変数は、それを制御できなくなります。 最後のステップは、最初のステップにのみリンクをジャンプバックする必要があります。

11.10 Modbus

考慮事項：

- Modbusはユーザースペースプログラムであるため、負荷の高いコンピューターでは遅延の問題が発生する可能性があります。
- Modbusは、モーターの位置制御や非常停止の制御などのハードリアルタイムイベントにはあまり適していません。

- Modbus を実行するには、Classic LadderGUI が実行されている必要があります。
- Modbus は完全には完成していないため、すべての modbus 機能を実行するわけではありません。

MODBUS を初期化するには、ClassicLadder ユーザースペースプログラムをロードするときにそれを指定する必要があります。

Modbus の読み込み

```
loadusr -w classicladder --modmaster myprogram.clp
```

-w を使用すると、HAL は Classic Ladder を閉じるまで待機してから、リアルタイムセッションを閉じます。コマンドラインで--modserver を追加すると、ClassicLadder は TCPmodbus スレーブもロードします。

MODBUS 機能

- 1-コイルを読み取る
- 2-入力の読み取り
- 3-保持レジスタを読み取ります
- 4-入力レジスタの読み取り
- 5-シングルコイルを書く
- 6-単一レジスタを書き込む
- 8-エコーテスト
- 15-複数のコイルを書く
- 16-複数のレジスタを書き込む

クラシックラダーユーザープログラムのロード時に--modmaster を指定しない場合、このページは表示されません。

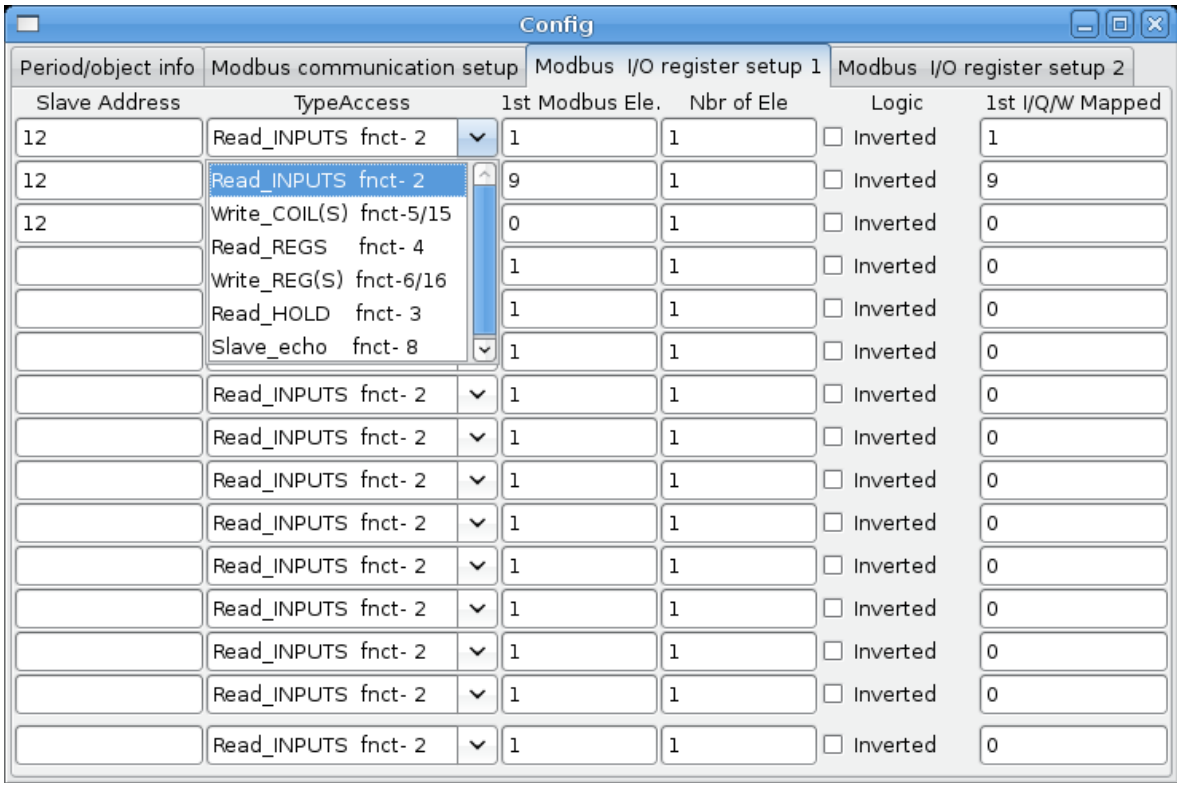


図 13-9

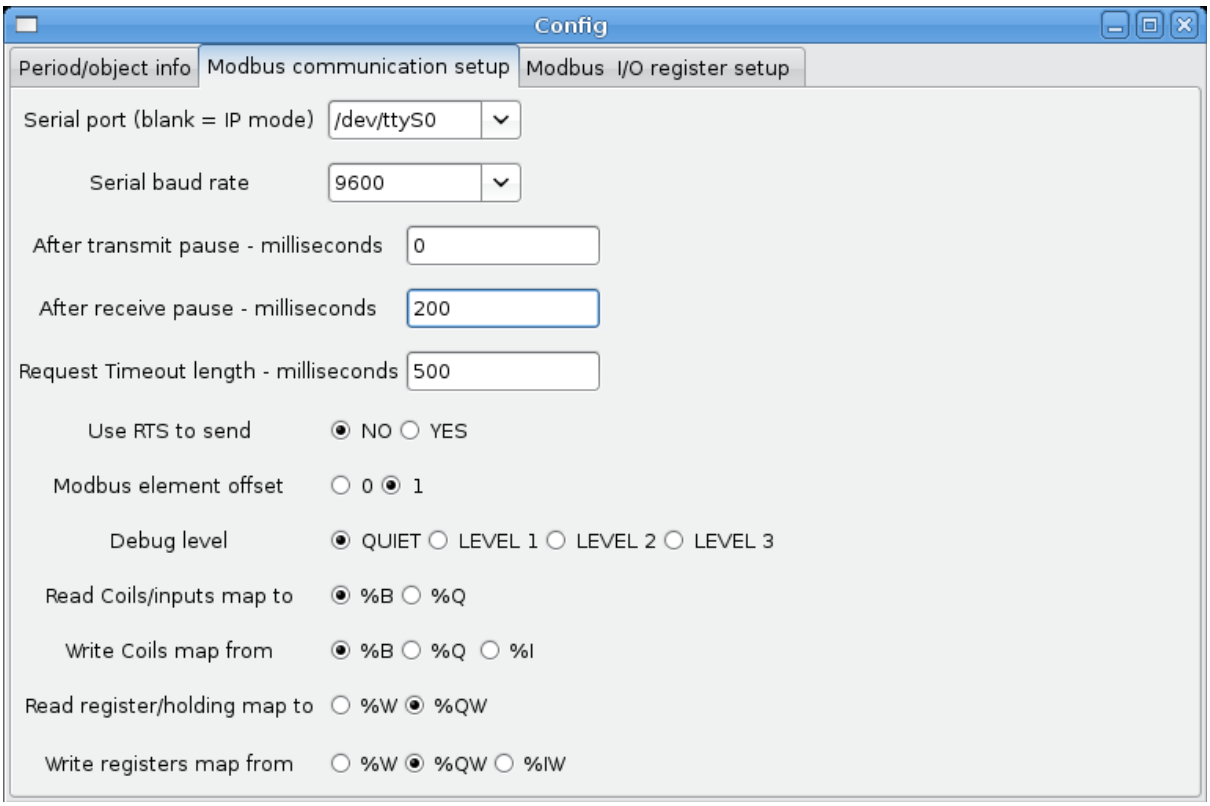


図 13-85

- シリアルポート-IP ブランク用。 シリアルの場合、シリアルドライバの場所/名前。 / dev / ttyS0（または USB からシリアルへのコンバーターの場合は/ dev / ttyUSB0）。

- シリアル速度-スレーブが設定されている速度に設定する必要があります-
  300、600、1200、2400、4800、9600、19200、38400、57600、115200 がサポートされています。
- 送信後の一時停止-送信後、応答を受信する前に一時停止（ミリ秒）する一部のデバイス（USB からシリアルへのコンバーターなど）。
- インターフレームの一時停止-スレーブから応答を受信した後、一時停止（ミリ秒）します。 これにより、リクエストのデューティサイクルが設定されます（各リクエストの一時停止です）。
- REQUEST TIMEOUT LENGTH-スレーブが応答しなかったと判断するまでの時間（ミリ秒）。
- MODBUS ELEMENT OFFSET-要素番号を 1 だけオフセットするために使用されます（メーカーの番号の違いの場合）。
- DEBUG LEVEL-これを 0〜3 に設定します（0 は、無応答エラー以外のデバッグ情報の出力を停止します）。
- コイル/入力マップの読み取り-コイル/入力を読み取る変数を更新するものを選択します。 （B または Q）。
- コイルマップの書き込み-コイルを書き込む変数を選択します。from（B、Q、または I）。
- READ REGISTERS / HOLDING-読み取りレジスタが更新する変数を選択します。 （W または QW）。
- レジスタマップの書き込み-レジスタを読み取る変数を更新する変数を選択します。 （W、QW、または IW）。
- スレーブアドレス-シリアルの場合、通常はスレーブデバイスで設定可能なスレーブ ID 番号（通常は 1〜256）。IP の場合、スレーブ IP アドレスとオプションでポート番号。
- TYPE ACCESS-これは、スレーブに送信する MODBUS 機能コードを選択します（たとえば、どのタイプの要求）。
- コイル/入力-入力とコイル（ビット）は、I、B、または Q 変数（ユーザーが選択）からの読み取り/書き込み（ユーザーが選択）します。
- レジスター（WORDS）-レジスター（Words / Numbers）は、IW、W、または QW 変数（ユーザーが選択）にマップされます。
- 1 番目の MODBUSELEMENT-グループ内の最初の要素のアドレス（またはレジスタ番号）。 （MODBUS ELEMENT OFFSET を正しく設定することを忘れないでください）。
- 要素の数-このグループの要素の数。
- LOGIC - You can invert the logic here.
- 1st%I%Q IQ WQ MAPPED-これは、modbus 要素グループに/からマップされる%B、%I、%Q、%W、%IW、または%QW 変数の開始番号です（最初の modbus から開始） 要素番号）。

上記の例では、次のようになっています。ポート番号-私のコンピューターの場合、/ dev / ttyS0 は私のシリアルポートでした。

シリアル速度は 9600 ボーに設定されています。

スレーブアドレスは 12 に設定されています（私の VFD では、これを 1〜31 に設定できます。つまり、1 つのシステムで最大 31 の VFD と通信できます）。

最初の行は、最初のレジスタ番号（レジスタ 1）から始まる 8 つの入力ビット用に設定されています。 したがって、レジスタ番号 1〜8 は、%B1 で始まり%B8 で終わる Classic Ladder の%B 変数にマップされます。

2 行目は、9 番目のレジスタ番号（レジスタ 9）から始まる 2 つの出力ビットに設定されているため、レジスタ番号 9〜10 は、%Q9 で始まり%Q10 で終わる Classic Ladder の%Q 変数にマップされます。

3 行目は、0 番目のレジスタ番号（レジスタ 0）から始まる 2 つのレジスタ（各 16 ビット）を書き込むように設定されているため、レジスタ番号 0-1 は、%W0 から始まり%W1 で終わるクラシックラダーの%W 変数にマップされます。

modbus 要素が 0 ではなく 1 から参照されることがあるため、1 つずつエラーを起こすのは簡単です（実際には、想定されている標準によって！）modbus 要素のオフセットラジオボタンを使用すると、 これとともに。

modbus スレーブデバイスのドキュメントには、レジスタの設定方法が記載されています。標準的な方法はありません。

SERIAL PORT、PORT SPEED、PAUSE、および DEBUG レベルは、変更のために編集可能です（ラジオボタンはすぐに適用されますが、構成ウィンドウの値が適用されます）。

エコー機能を使用するには、エコー機能を選択し、テストするスレーブ番号を追加します。 変数を指定する必要はありません。

番号 257 が指定したスレーブ番号に送信され、スレーブはそれを送り返す必要があります。 メッセージを表示するには、ターミナルでクラシックラダーを実行する必要があります。

11.10.1 MODBUS 設定

シリアル：

- クラシックラダーは RTU プロトコル（ASCII ではない）を使用します。
- 8 データビット、パリティは使用されません。1 ストップビットは 8-N-1 とも呼ばれます。
- ボーレートは、スレーブとマスターで同じである必要があります。 クラシックラダーは 1 つのボーレートしか持てないため、すべてのスレーブを同じレートに設定する必要があります。
- フレーム間一時停止は、回答を受け取った後に一時停止する時間です。
- MODBUS_TIME_AFTER_TRANSMIT は、要求を送信してから応答を受信するまでの一時停止の長さです（これは、低速の USB コンバーターで明らかに役立ちます）。

11.10.2 MODBUS 情報

- クラシッククラダーは、modbus プロトコル（「マスター」：スレーブのポーリング）を使用して、モジュールで分散入力/出力を使用できます。
- スレーブとその I / O は、設定ウィンドウで設定できます。
- 2 つの排他的モードが利用可能です：Modbus / TCP を使用するイーサネットと Modbus / RTU を使用するシリアル。
- パリティは使用されません。
- シリアルのポート名が設定されていない場合は、TCP / IP モードが使用されます。 。 。
- スレーブアドレスは、スレーブアドレス（Modbus / RTU）または IP アドレスです。
- 使用するポート番号（xx.xx.xx.xx：pppp）ごとに IP アドレスを追跡できます。それ以外の場合は、デフォルトでポート 9502 が使用されます。
- テストには、Modbus / TCP 製品（Adam-6051、http：//www.advantech.com）とシリアル Modbus / RTU 製品（http://www.ipac.ws）の 2 つの製品が使用されています。
- 例を参照してください：adam-6051 および modbus_rtu_serial。
- Web リンク：http：//www.modbus.org およびこの興味深いリンク：http：//www.iatips.com/modbus.html
- MODBUSTCP サーバーが含まれています
- クラシッククラダーには、Modbus / TCP サーバーが統合されています。 デフォルトのポートは 9502 です（以前の標準 502 では、root 権限でアプリケーションを起動する必要があります）。
- サポートされている Modbus 機能コードのリストは、1、2、3、4、5、6、15、および 16 です。
- Modbus ビットとワードの対応テーブルは実際にはパラメトリックではなく、%B 変数と%W 変数に直接対応します。

modbus プロトコルの詳細については、インターネットで入手できます。

http://www.modbus.org/

11.10.3 通信エラー

通信エラーが発生した場合、警告ウィンドウがポップアップし（GUI が実行されている場合）、%E0 が true になります。 Modbus は引き続き通信を試みます。 %E0 は、エラーに基づいて決定を下すために使用できます。 タイマーを使用して、タイムアウトした場合などにマシンを停止することができます。

11.10.4 modbus の問題のデバッグ

プロトコルの良いリファレンス：

http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf

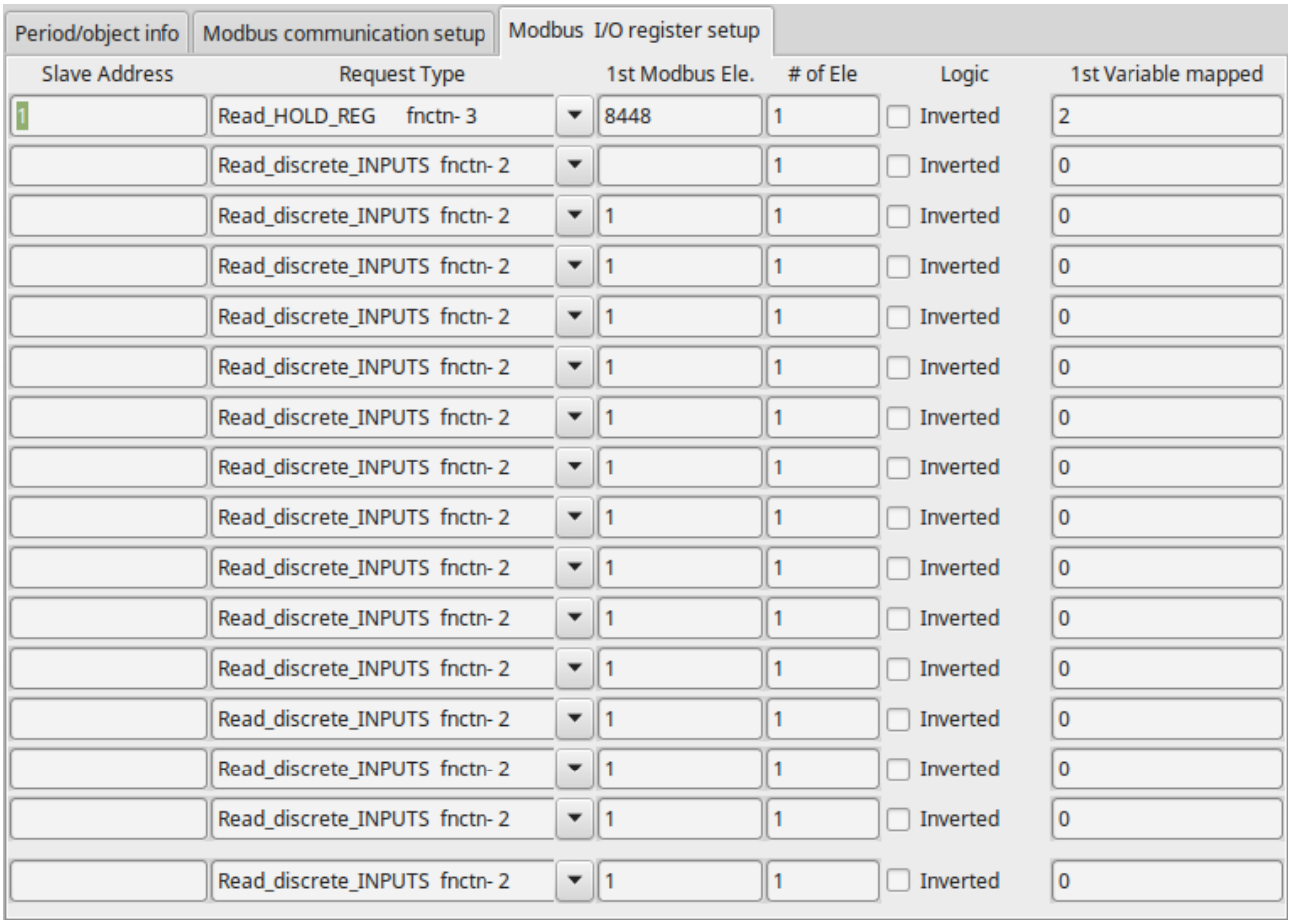
ターミナルから linuxcnc / classocladder を実行すると、Modbus コマンドとスレーブ応答が出力されます。

ここでは、Classiclader がスレーブ 1 を要求するように設定しました。

アドレス 8448（0x2100）から始まる保持レジスタ（機能コード 3）を読み取る

1（2 バイト幅）のデータ要素を返すように要求します

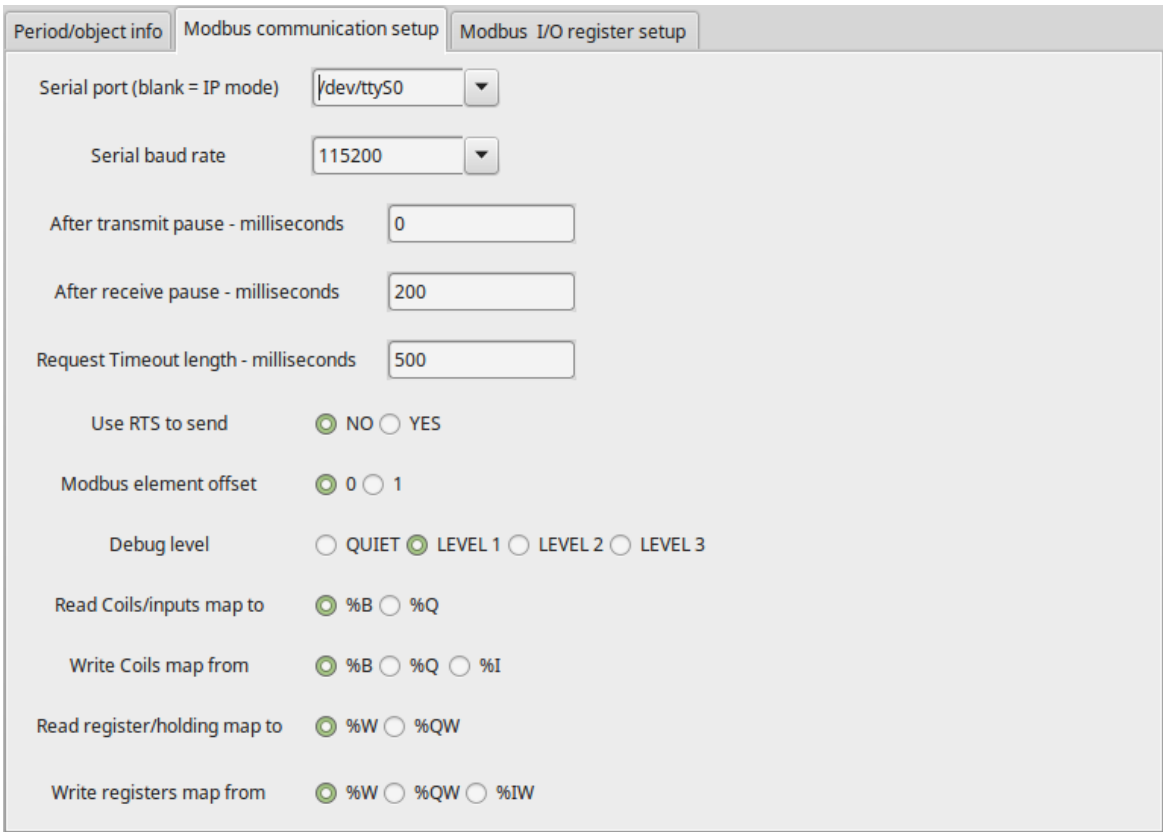
2 から始まる Classicladder 変数にマップします



この画像では、デバッグレベルを 1 に設定しているため、modbus メッセージが端末に出力されます。

読み取りおよび書き込みの保持レジスタを classicladder の%W 変数にマッピングしました

したがって、返されたデータは、2 番目の要素からデータをマッピングした他の画像と同様に%W2 になります。



1.1.1.1 リクエスト

8448 Decimal（0x2100 Hex）で 1 つのホールドレジスタを読み取る例を見てみましょう。

Modbus プロトコルリファレンスを見る：

表 13.2：保留登録要求の読み取り

名前	バイト数	値（16 進数）
機能コード	（1 バイト）	3（0x03）
開始アドレス	（2 バイト）	0～65535（0x0000～0xFFFF）
レジスターの数	（2 バイト）	1～125（0x7D）
チェックサム	（2 バイト）	自動的に計算

ターミナル（すべて 16 進数）に出力された送信コマンドの例を次に示します。

```
INFO CLASSICLADDER- Modbus I/O module to send: Lgt=8 <- Slave address-1 Function code-3  -
Data-21 0 0 1 8E 36
```

意味（16 進数）：

- Lgt = 8 =メッセージの長さはスレーブ番号とチェックサム番号を含めて 8 バイトです

- スレーブ番号= 1（0x1）=スレーブアドレス 1
- 機能コード= 3（0x3）=保持レジスタの読み取り
- アドレスから開始=ハイバイト 33（0x21）ローバイト 0（0x00）=結合アドレス= 8448（0x2100）
- レジスタ数= 1（0x1）=戻り値 1 2 バイトレジスタ（レジスタの保持と読み取りは常に 2 バイト幅です）
- チェックサム=上位バイト 0x8E 下位バイト 0x36 =（0x8E36）

11.10.4.1 エラー応答

エラー応答がある場合は、機能コードに加えて 0x80、エラーコード、およびチェックサムを送信します。

エラー応答を取得するということは、スレーブが要求コマンドを認識しているが、有効なデータを提供できないことを意味します。

Modbus プロトコルリファレンスを見る：

表 13.3：機能コード 3（保持レジスタの読み取り）に対して返されるエラー

名前	バイト数	値（16 進数）
エラーコード	1 バイト	131（0x83）
例外コード	1 バイト	1-4（0x01 から 0x04）
チェックサム	（2 バイト）	自動的に計算

例外コードの意味：

- 1-違法な機能
- 2-不正なデータアドレス
- 3-不正なデータ値
- 4-スレーブデバイスの障害

ターミナル（すべて 16 進数）に出力された受信コマンドの例を次に示します。

```
INFO CLASSICLADDER- Modbus I/O module received: Lgt=5 -> (Slave address-1 Function -code-83 ) 2 C0 F1
```

意味（16 進数）：

- スレーブ番号= 1（0x1）=スレーブアドレス 1
- 機能コード= 131（0x83）=保持レジスタの読み取り中にエラーが発生しました
- •エラーコード= 2（0x2）=不正なデータアドレスが要求されました

- チェックサム=（0x8E36）

11.10.4.2 データ応答

応答の Modbus プロトコルリファレンスを調べます。

表 13.4：機能コード 3 のデータ応答（保持レジスタの読み取り）

名前	バイト数	値（16 進数）
機能コード	1 バイト	3（0x03）
バイトカウント	1 バイト	2 x N *
チェックサム	（2 バイト）	自動的に計算
（* N =レジスターの数）		

ターミナル（すべて 16 進数）に出力された受信コマンドの例を次に示します。

```
INFO CLASSICLADDER- Modbus I/O module received: Lgt=7 -> (Slave address-1 Function -code-3 2 0 0 B8 44
```

意味（16 進数）：

- スレーブ番号= 1（0x1）=スレーブアドレス 1
- 要求された機能コード= 3（0x3）=要求された保持レジスタの読み取り
- バイトレジスタの数= 2（0x1）= 2 バイトを返します（各レジスタ値は 2 バイト幅です）
- ハイバイトの値= 0（0x0）=アドレス 8448 の上位バイト値（0x2100）
- ローバイトの値= 0（0x0）=アドレス 8448 の上位バイト値（0x2100）
- •チェックサム=（0xB844）

（上位バイトと下位バイトを組み合わせて 16 ビット値を作成し、Classicladder の変数に転送します）

読み取りレジスタは、%W または%QW（内部メモリまたは HAL 出力ピン）にマップできます。

書き込みレジスタは、%W、%QW、または%IW（内部メモリ、HAL 出力ピンまたは HAL 入力ピン）からマッピングできます。

変数番号は、modbus I / O レジストリ設定ページの列に入力された番号から始まります。最初の変数がマップされます 1 回の読み取り/書き込みで複数のレジスタが要求された場合、変数番号は最初のレジスタの後に連続します。

11.10.5 MODBUS のバグ

- 比較ブロックでは、関数%W = ABS（%W1-%W2）が受け入れられますが、正しく計算されません。 現在、%W0 = ABS（%W1）のみが有効です。
- ラダープログラムをロードすると、Modbus 情報がロードされますが、ClassicLadder に Modbus を初期化するようには指示されません。 --modmaster を追加して、GUI を最初にロードするときに Modbus を初期化する必要があります。
- セクションマネージャがセクションディスプレイの上に配置されている場合、スクロールバーを横切って終了をクリックすると、ユーザープログラムがクラッシュします。
- --modmaster を使用する場合は、ラダープログラムを同時にロードする必要があります。そうしないと、TCP のみが機能します。
- Modbus での複数のレジスタの読み取り/書き込みにはチェックサムエラーがあります。

11.11 クラシックラダーの設定

このセクションでは、クラシックラダーを Stepconf ウィザードで生成された構成に追加するために必要な手順について説明します。 Stepconf Wizard の[ConfigurationOptions]ページで、[Include Classic LadderPLC]をオフにします。

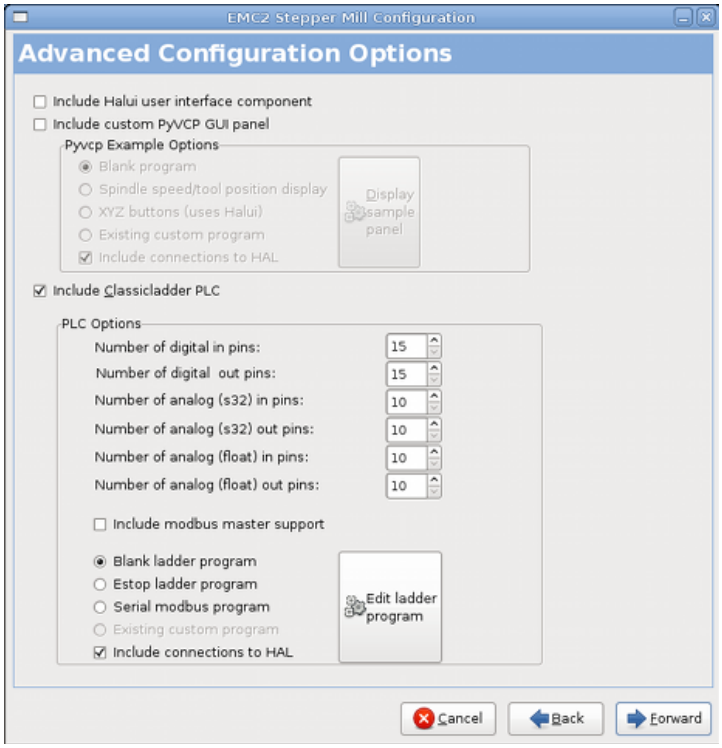


図 13-86

11.11.1 モジュールを追加する

Stepconf ウィザードを使用してクラシッククラダーを追加した場合は、この手順をスキップできます。

クラシッククラダーを手動で追加するには、最初にモジュールを追加する必要があります。 これは、custom.hal ファイルに数行を追加することによって行われます。

この行は、リアルタイムモジュールをロードします。

```
loadrt classicladder_rt
```

この行は、サーボスレッドにクラシッククラダー機能を追加します。

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

11.11.2 ラダーロジックの追加

次に、構成を起動し、「ファイル/ラダーエディター」を選択してクラシッククラダー GUI を開きます。 上記のように、空白の[セクション表示]ウィンドウと[セクションマネージャー]ウィンドウが表示されます。 [セクション表示]ウィンドウで、エディターを開きます。 [エディター]ウィンドウで、[変更]を選択します。 これで、プロパティウィンドウがポップアップし、セクション表示にグリッドが表示されます。 グリッドははしごの 1 つのラングです。 ラングにはブランチを含めることができます。 単純なラングには、1 つの入力、コネクタライン、および 1 つの出力があります。 ラングには、最大 6 つの水平ブランチを含めることができます。 実行中に複数の回路を使用することは可能ですが、結果は予測できません。

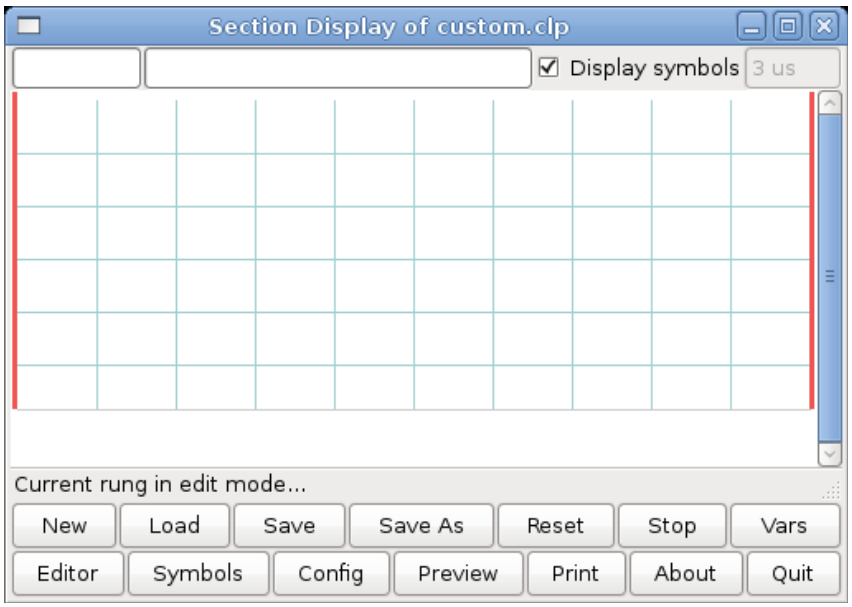


図 13-87

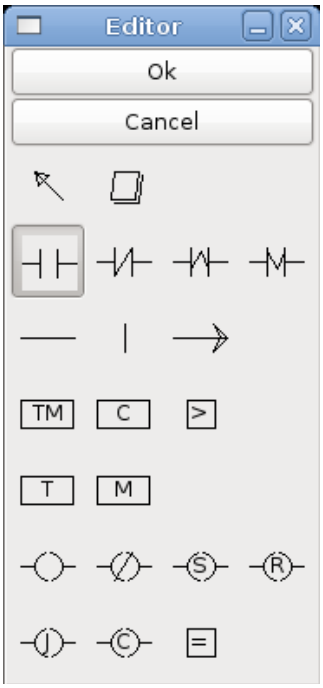


図 13-88

次に、左上のグリッドをクリックして N.O を配置します。 はしごに入力します。

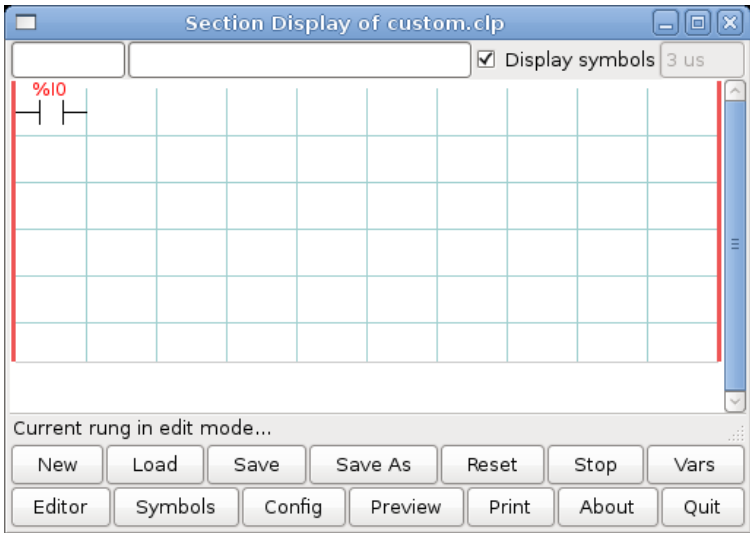


図 13-89

上記の手順を繰り返して、N.O を追加します。 右上のグリッドに出力し、水平接続を使用して 2 つを接続します。 次のようになります。 そうでない場合は、消しゴムを使用して不要なセクションを削除します。

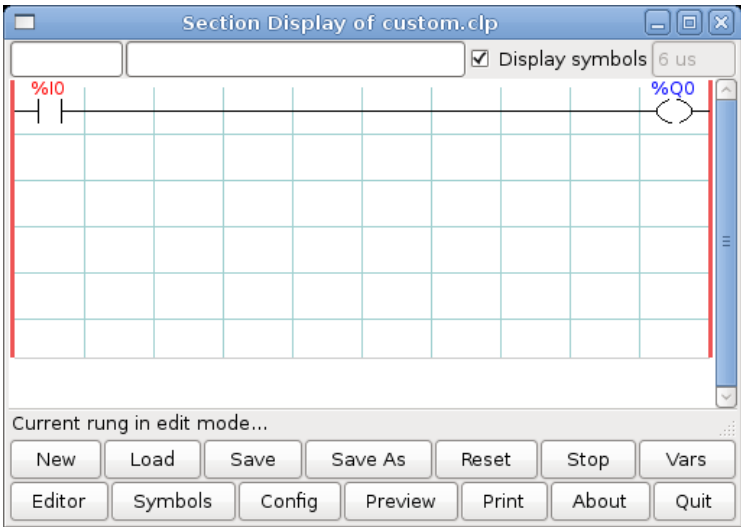


図 13-90

次に、エディタウィンドウの[OK]ボタンをクリックします。 これで、セクション表示は次のようになります。

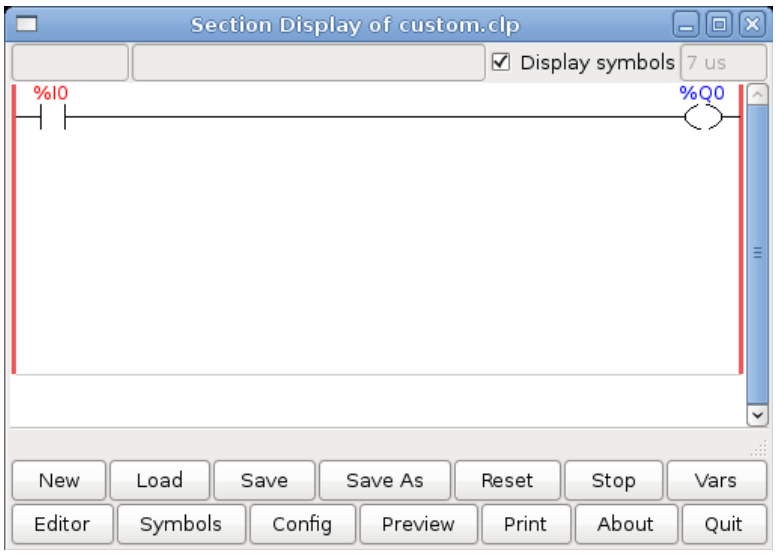


図 13-91

新しいファイルを保存するには、[名前を付けて保存]を選択し、名前を付けます。 .clp 拡張子は自動的に追加されます。 保存する場所として、デフォルトで実行中の config ディレクトリに設定する必要があります。

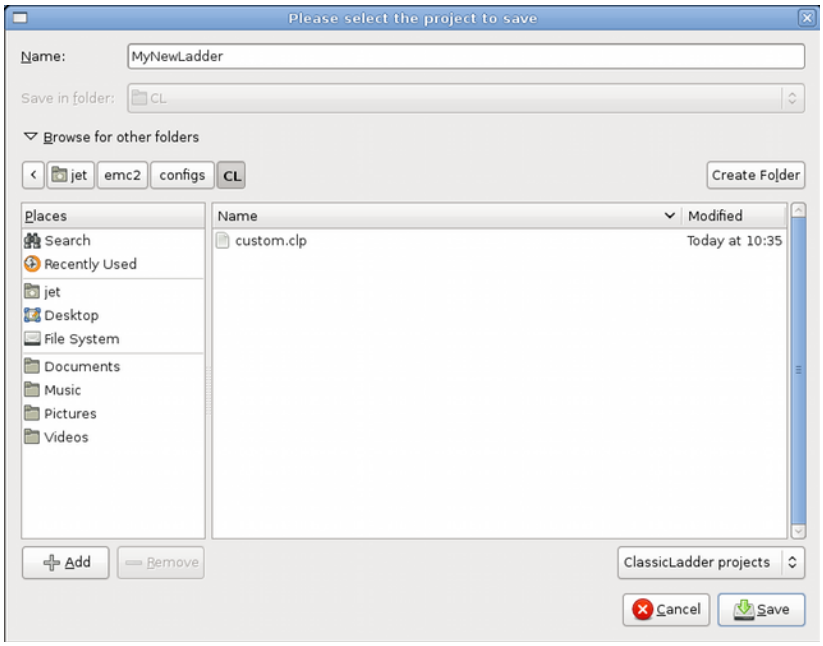


図 13-92

ここでも、Stepconf ウィザードを使用してクラシックラダーを追加した場合は、この手順をスキップできます。

ラダーを手動で追加するには、ラダーファイルをロードする行を custom.hal ファイルに追加する必要があります。 LinuxCNC セッションを閉じて、この行を custom.hal ファイルに追加します。

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

これで、LinuxCNC 構成を起動すると、ラダープログラムも実行されます。 「ファイル/ラダーエディタ」を選択すると、作成したプログラムがセクション表示ウィンドウに表示されます。

11.12 クラシックラダーの例

11.12.1 ラッピングカウンター

カウンターをラップアラウンドするには、プリセットピンとリセットピンを使用する必要があります。 カウンターを作成するときは、ラップアラウンドする前に到達したい数にプリセットを 0 に設定します。ロジックは、カウンター値がプリセットを超えている場合はカウンターをリセットし、アンダーフローがオンの場合はカウンター値をプリセットに設定します。 価値。 例でわかるように、カウンター値がカウンタープリセットよりも大きい場合、カウンターリセットがトリガーされ、値は 0 になります。アンダーフロー出力%Q2 は、逆方向にカウントするときに、カウンター値をプリセットに設定します。

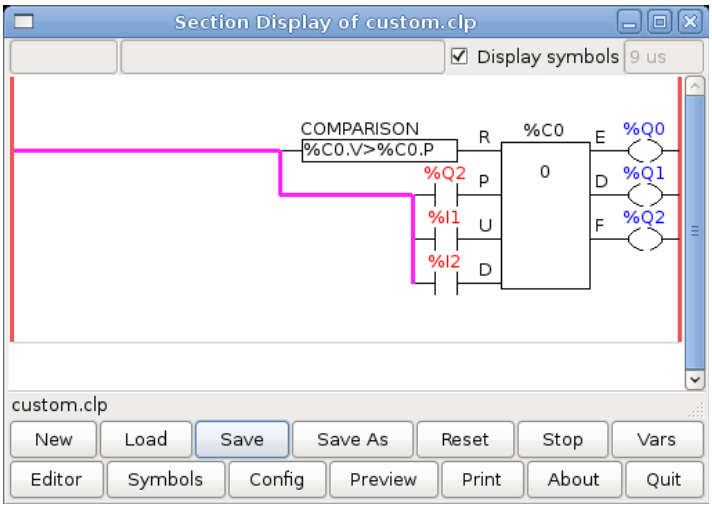


図 13-93

11.12.2 余分なパルスを拒否する

この例は、入力からの余分なパルスを拒否する方法を示しています。 入力パルス%I0 に、ロジックを台無しにする余分なパルスを与えるという厄介な習慣があるとします。 TOF（Timer Off Delay）は、余分なパルスがクリーンアップされた出力%Q0 に到達するのを防ぎます。 これがどのように機能するかは、タイマーが入力を取得したときに、タイマーの出力が時間設定の間オンになっていることです。 ノーマルクローズ接点%TM0.Q を使用すると、タイマーの出力は、タイムアウトになるまで、それ以上の入力が出力に到達するのをブロックします。

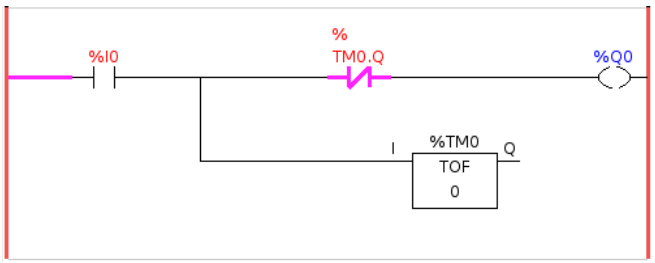


図 13-94

11.12.3 外部非常停止

外部非常停止の例は、/ config / classicladder / cl-estop フォルダーにあります。 pyVCP パネルを使用して、外部コンポーネントをシミュレートします。

外部の非常停止を LinuxCNC に接続し、外部の非常停止を内部の非常停止と連携させるには、クラシックラダーを介した接続がいくつか必要です。

まず、図のようにポンド記号を追加するか、次の行を削除してコメントアウトすることにより、メイン HAL ファイルの非常停止ループを開く必要があります。

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

次に、次の 2 行を追加して、custom.hal ファイルに ClassicLadder を追加します。

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

次に、構成を実行し、ここに示すようにラダーを構築します。

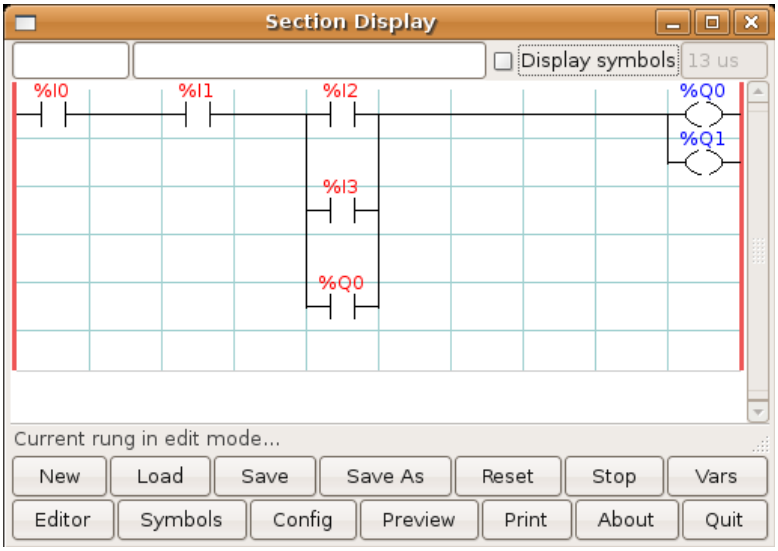


図 13-95

ラダーを作成したら、[名前を付けて保存]を選択し、ラダーを estop.clp として保存します。

次に、custom.hal ファイルに次の行を追加します。

```
# Load the ladder
loadusr classicladder --nogui estop.clp
```

I / O の割り当て

- %I0 = pyVCP パネルからの入力シミュレートされた非常停止（チェックボックス）
- %I1 = LinuxCNC の非常停止からの入力
- %I2 = LinuxCNC の非常停止リセットパルスからの入力
- %I3 = pyVCP パネルのリセットボタンからの入力
- %Q0 =有効にするための LinuxCNC への出力
- %Q1 =外部ドライバボードのイネーブルピンへの出力（ボードにディセーブルピンがある場合は N / C 出力を使用）

次に、custom_postgui.hal ファイルに次の行を追加します

```
# E-Stop example using pyVCP buttons to simulate external components
# The pyVCP checkbutton simulates a normally closed external E-Stop
net ext-estop classicladder.0.in-00 <= pyvcp.py-estop
```

```
# Request E-Stop Enable from LinuxCNC
net estop-all-ok iocontrol.0.emc-enable-in <= classicladder.0.out-00
# Request E-Stop Enable from pyVCP or external source
net ext-estop-reset classicladder.0.in-03 <= pyvcp.py-reset
# This line resets the E-Stop from LinuxCNC
net emc-reset-estop iocontrol.0.user-request-enable =>
classicladder.0.in-02
# This line enables LinuxCNC to unlatch the E-Stop in Classic Ladder
net emc-estop iocontrol.0.user-enable-out => classicladder.0.in-01
# This line turns on the green indicator when out of E-Stop
net estop-all-ok => pyvcp.py-es-status
```

次に、次の行を panel.xml ファイルに追加します。 デフォルトの html ビューアではなく、テキストエディタで開く必要があることに注意してください。

```
<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>
```

次に、構成を起動すると、次のようになります。

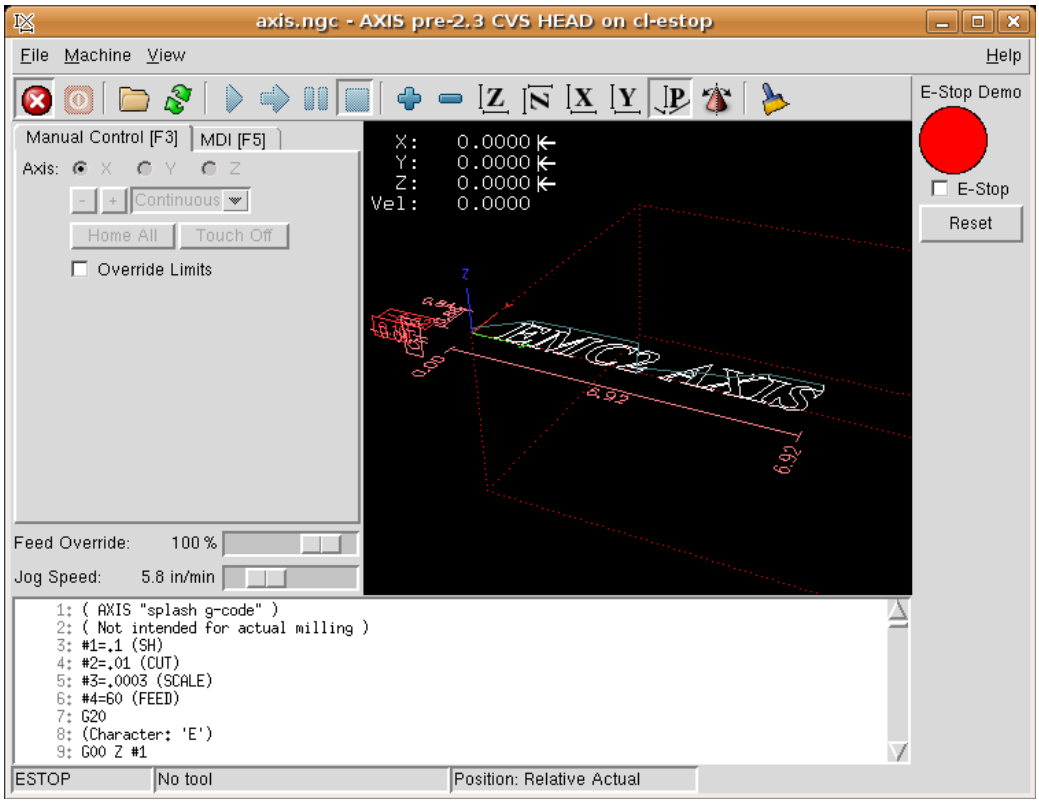


図 13-96

この例では、実際の例と同様に、AXIS E-Stop または外部リセットによって OFF モードになる前に、リモート E-Stop（チェックボックスでシミュレート）をクリアする必要があることに注意してください。 AXIS 画面の非常停止が押された場合は、もう一度押すとクリアする必要があります。 AXIS で非常停止を実行した後は、外部からリセットすることはできません。

11.12.4 タイマー/操作例タイマー/操作例

この例では、Operate ブロックを使用して、入力がオンかオフかに基づいてタイマープリセットに値を割り当てています。

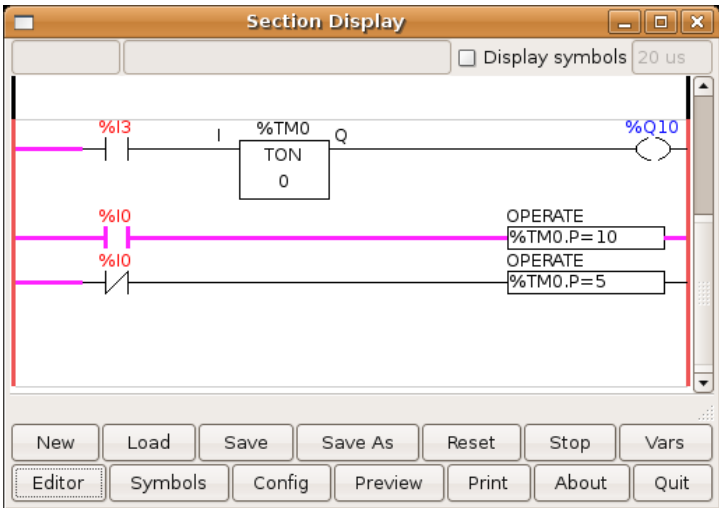


図 13-97

この場合、%I0 は true であるため、タイマープリセット値は 10 です。%I0 が false の場合、タイマープリセットは 5 になります。

12章 HAL

12.1 HAL の紹介

HAL は HARDWAREABSTRACTIONLAYER の略です。 最高レベルでは、これは、複雑なシステムを組み立てるために、多数のビルディングブロックをロードして相互接続できるようにする方法にすぎません。 ハードウェアの部分は、HAL が元々、さまざまなハードウェアデバイス用に LINUXCNC を簡単に構成できるように設計されていたためです。 ビルディングブロックの多くは、ハードウェアデバイスのドライバーです。 ただし、HAL はハードウェアドライバーを構成するだけではありません。

12.1.1 HAL は、従来のシステム設計手法に基づいています

HAL は、ハードウェア回路およびシステムの設計に使用されるのと同じ原則に基づいているため、最初にそれらの原則を調べると便利です。

すべてのシステム（CNC マシンを含む）は、相互接続されたコンポーネントで構成されています。 CNC マシンの場合、これらのコンポーネントは、メインコントローラー、サーボアンプまたはステッピングドライブ、モーター、エンコーダー、リミットスイッチ、押しボタンペンダント、おそらくスピンドルドライブ用の VFD、ツールチェンジャーを実行するための PLC などです。 これらの部品を選択、取り付け、配線して、完全なシステムを作成します。

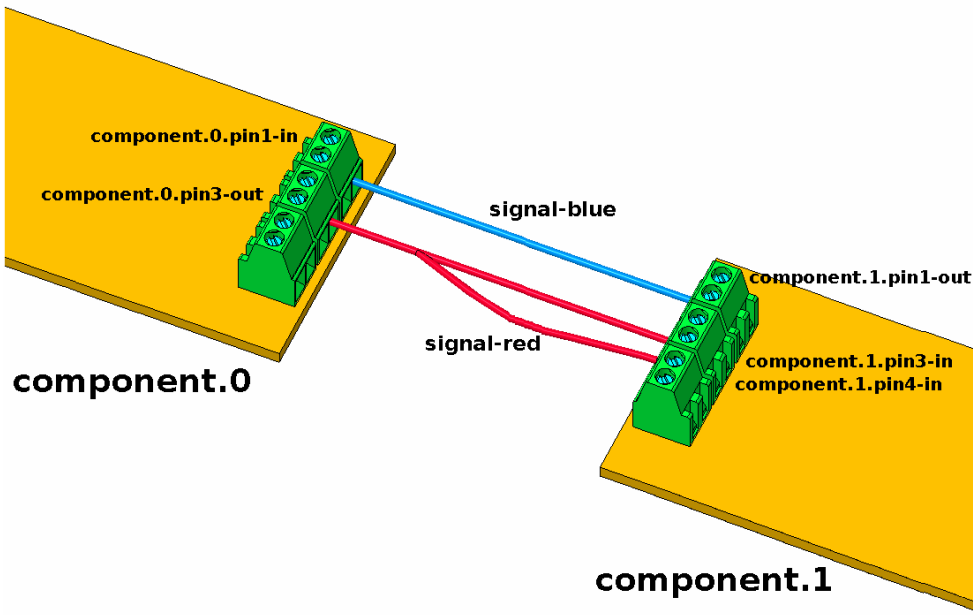


図 14-98 **HAL コンセプト-電気回路の接続**

図 1 は、次のような HAL コードで記述されます。

```
net signal-blue component.0.pin1-in component.1.pin1-out
```

```
net signal-red component.0.pin3-out component.1.pin3-in component.1.pin4-in
```

12.1.2 パーツの選択

機械メーカーは、個々の部品がどのように機能するかを心配する必要はありません。 彼はそれらをブラックボックスとして扱います。 設計段階で、ステッパーまたはサーボ、サーボアンプのブランド、リミットスイッチの種類、数など、使用する部品を決定します。使用する特定のコンポーネントに関するインテグレーターの決定は、その内容に基づいています。 コンポーネントが行い、仕様はデバイスの製造元から提供されます。 モーターのサイズとモーターが駆動する必要のある負荷は、モーターを実行するために給電されるアンプの選択に影響します。 アンプの選択は、アンプに必要なフィードバックの種類と、コントロールからアンプに送信する必要のある速度または位置信号に影響を与える可能性があります。

HAL の世界では、インテグレーターは必要な HAL コンポーネントを決定する必要があります。 通常、すべてのインターフェイスカードにはドライバが必要です。 ステップパルスのソフトウェア生成、PLC 機能、およびその他のさまざまなタスクには、追加のコンポーネントが必要になる場合があります。

12.1.3 相互接続設計

ハードウェアシステムの設計者は、部品を選択するだけでなく、それらの部品をどのように相互接続するかも決定します。 各ブラックボックスには端子があり、単純なスイッチの場合は 2 つ、サーボドライブまたは PLC の場合は数十になります。 それらは一緒に配線する必要があります。 モーターはサーボアンプに接続し、リミットスイッチはコントローラーに接続します。 機械製造業者が設計に取り組むとき、彼はすべての部品がどのように相互接続されるべきかを示す大きな配線図を作成します。

HAL を使用する場合、コンポーネントは信号によって相互接続されます。 設計者は、必要な信号とそれらが接続するものを決定する必要があります。

12.1.4 実装

配線図が完成したら、マシンを構築します。 部品を入手して取り付ける必要があります。その後、配線図に従って相互接続します。 物理システムでは、各相互接続は、適切な端子に切断して接続する必要がある 1 本のワイヤです。

HAL は、HAL システムの構築に役立つ多くのツールを提供します。 一部のツールでは、1 本のワイヤーを接続（または切断）できます。 他のツールを使用すると、システムに関するすべての部品、配線、およびその他の情報の完全なリストを保存できるため、1 つのコマンドで再構築できます。

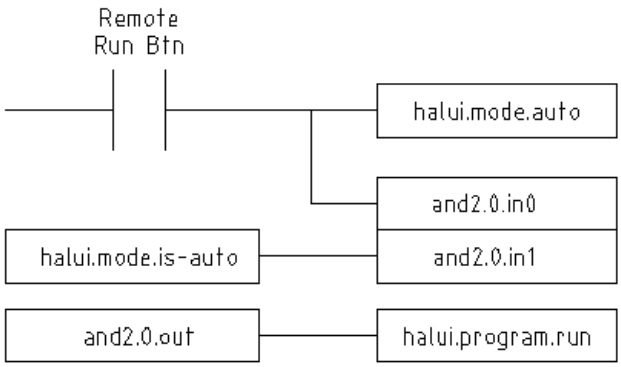
12.1.5 テスト

初めて正しく動作するマシンはほとんどありません。 テスト中、ビルダーはメーターを使用して、リミットスイッチが機能しているかどうかを確認したり、サーボモーターに流れる DC 電圧を測定したりする場合があります。 彼は、ドライブのチューニングをチェックしたり、電気ノイズを探したりするために、オシロスコープを接続する場合があります。 彼は、配線図を変更する必要がある問題を見つけるかもしれません。 おそらく、パーツを別の方法で接続するか、まったく別のものに交換する必要があります。

HAL は、電圧計、オシロスコープ、信号発生器、およびシステムのテストと調整に必要なその他のツールに相当するソフトウェアを提供します。 システムの構築に使用したのと同じコマンドを使用して、必要に応じて変更を加えることができます。

12.1.6 概要

このドキュメントは、この種のハードウェアシステム統合を行う方法をすでに知っているが、ハードウェアを LINUXCNC に接続する方法を知らない人を対象としています。 HALUI の例のドキュメントの「リモートスタートの例」セクションを参照してください。



上記の従来のハードウェア設計は、メインコントロールの端で終わります。 コントロールの外側には、必要なことをすべて実行するために相互に接続された、比較的単純なボックスの束があります。 内部では、コントロールは大きな謎です。1 つの巨大なブラックボックスが機能することを願っています。

HAL は、この従来のハードウェア設計手法を大きなブラックボックスの内部に拡張します。これにより、デバイスドライバーやコントローラーの一部の内部部品が、外部ハードウェアと同じように相互接続したり、交換したりできる小さなブラックボックスになります。 これにより、システムの配線図に、大きなブラックボックスだけでなく、内部コントローラーの一部を表示できます。 そして最も重要なことは、インテグレーターが残りのハードウェアで使用するのと同じ方法を使用してコントローラーをテストおよび変更できるようにすることです。

モーター、アンプ、エンコーダーなどの用語は、ほとんどのマシンインテグレーターによく知られています。 エンコーダをコンピュータのサーボ入力ボードに接続するために非常に柔軟な 8 導体シールドケーブルを使用することについて話すとき、読者はそれが何であるかをすぐに理解

し、両端をどのような種類のコネクタで構成する必要があるかという質問に導かれます 。 HAL にも同じような考え方が不可欠ですが、特定の思考の流れが軌道に乗るには少し時間がかかる場合があります。 HAL ワードの使用は、最初は少し奇妙に思えるかもしれませんが、ある接続から次の接続へと機能するという概念は同じです。

配線図をコントローラーの内部に拡張するというこのアイデアが、HAL のすべてです。 ハードウェアブラックボックスを相互接続するというアイデアに慣れている場合は、HAL を使用してソフトウェアブラックボックスを相互接続することにほとんど問題はありません。

12.1.7 HAL の概念

このセクションは、主要な HAL 用語を定義する用語集ですが、これらの用語がアルファベット順に配置されていないため、従来の用語集とは少し異なります。 それらは、HAL の方法での関係またはフローによって配置されます。

COMPONENT

ハードウェア設計について話すとき、個々の部品をパーツ、ビルディングブロック、ブラックボックスなどと呼びました。HAL に相当するものはコンポーネントまたは HAL コンポーネントです。 （このドキュメントでは、他の種類のコンポーネントと混同される可能性がある場合に HAL コンポーネントを使用しますが、通常はコンポーネントのみを使用します。）HAL コンポーネントは、明確に定義された入力、出力、および動作を備えたソフトウェアであり、インストールして 必要に応じて相互接続します。

PARAMETER

多くのハードウェアコンポーネントには、他のコンポーネントに接続されていないが、アクセスする必要がある調整があります。 たとえば、サーボアンプには、チューニング調整を可能にするトリムポットや、メーターまたはスコープを取り付けてチューニング結果を表示できるテストポイントが含まれていることがよくあります。 HAL コンポーネントには、パラメーターと呼ばれるそのような項目を含めることもできます。 パラメータには 2 つのタイプがあります。入力パラメータはトリムポットと同等です。これらはユーザーが調整できる値であり、一度設定すると固定されたままになります。 出力パラメータはユーザーが調整することはできません。これらは、内部信号を監視できるテストポイントに相当します。

PIN

ハードウェアコンポーネントには、それらを相互接続するために使用される端子があります。 HAL に相当するものは、ピンまたは HAL ピンです。 （混乱を避けるために、必要に応じて HAL ピンが使用されます。）すべての HAL ピンには名前が付けられ、ピン名はそれらを相互接続するときに使用されます。 HAL ピンは、コンピューター内にのみ存在するソフトウェアエンティティです。

PHYSICAL_PIN

多くのI / Oデバイスには、パラレルポートコネクタのピンなど、外部ハードウェアに接続する実際の物理ピンまたは端子があります。 混乱を避けるために、これらは物理ピンと呼ばれます。 これらは現実の世界に突き出ているものです。

SIGNAL

物理マシンでは、実際のハードウェアコンポーネントの端子はワイヤで相互接続されています。 ワイヤに相当するHALは、信号またはHAL信号です。 HAL信号は、機械製造業者の要求に応じてHALピンを相互に接続します。 HAL信号は、（マシンの実行中でも）自由に切断および再接続できます。

TYPE

実際のハードウェアを使用する場合、24ボルトのリレー出力をサーボアンプの+/- 10Vアナログ入力に接続することはありません。 HALピンにも同じ制限があり、タイプに基づいています。 ピンと信号の両方にタイプがあり、信号は同じタイプのピンにのみ接続できます。 現在、次の4種類があります。

- ビット-単一のTRUE / FALSEまたはON / OFF値
- FLOAT-64ビットの浮動小数点値で、解像度は約53ビット、ダイナミックレンジは1000ビットを超えます。
- U32-32ビットの符号なし整数。有効な値は0〜4,294,967,295です。
- S32-32ビットの符号付き整数。有効な値は-2,147,483,647〜 +2,147,483,647です。

FUNCTION

実際のハードウェアコンポーネントは、入力にすぐに作用する傾向があります。 たとえば、サーボアンプへの入力電圧が変化すると、出力も自動的に変化します。 ただし、ソフトウェアコンポーネントは自動的に機能しません。 各コンポーネントには、そのコンポーネントが実行することになっていることを実行するために実行する必要がある特定のコードがあります。 場合によっては、そのコードは単にコンポーネントの一部として実行されます。 ただし、ほとんどの場合、特にリアルタイムコンポーネントでは、コードを特定の順序で特定の間隔で実行する必要があります。 たとえば、入力データに対して計算を実行する前に入力を読み取る必要があり、計算が完了するまで出力を書き込む必要はありません。 このような場合、コードは1つ以上の関数の形でシステムで利用できるようになります。 各関数は、特定のアクションを実行するコードのブロックです。 システムインテグレーターは、スレッドを使用して、特定の順序で特定の時間間隔で実行される一連の関数をスケジュールできます。

THREAD

スレッドは、リアルタイムタスクの一部として特定の間隔で実行される関数のリストです。スレッドが最初に作成されるとき、特定の時間間隔（期間）がありますが、機能はありません。 関数はスレッドに追加でき、スレッドが実行されるたびに順番に実行されます。

例として、HAL_PARPORT という名前の PARPORT コンポーネントがあるとします。 そのコンポーネントは、物理ピンごとに 1 つ以上の HAL ピンを定義します。 ピンについては、そのコンポーネントのドキュメントセクションで説明されています。名前、各ピンと物理ピンの関係、反転されているか、極性を変更できるかなどです。ただし、それだけでは HAL ピンから物理ピンにデータを取得できません。 。 それを行うにはコードが必要であり、そこで関数が登場します。 パーポートコンポーネントには、少なくとも 2 つの機能が必要です。1 つは物理入力ピンを読み取って HAL ピンを更新する機能、もう 1 つは HAL ピンからデータを取得して物理出力ピンに書き込む機能です。 これらの関数は両方とも、PARPORT ドライバーの一部です。

12.1.8 HAL コンポーネント

各 HAL コンポーネントは、明確に定義された入力、出力、および動作を備えたソフトウェアであり、必要に応じてインストールおよび相互接続できます。 このセクションでは、使用可能なコンポーネントのいくつかと、それぞれの機能について簡単に説明します。 各コンポーネントの完全な詳細は、このドキュメントの後半にあります。

12.1.9 HAL フックを備えた外部プログラム

MOTION

NML 1 モーションコマンドを受け入れ、HAL と相互作用するリアルタイムモジュール

IOCONTROL

NML I / O コマンドを受け入れ、HAL と対話するユーザースペースモジュール

CLASSICLADDER

すべての I / O に HAL を使用する PLC

HALUI

HAL と対話し、NML コマンドを送信するユーザースペースプログラムであり、外部のノブとスイッチを使用して完全なユーザーインターフェイスとして機能することを目的としています。

12.1.10 内部コンポーネント

STEPGEN

位置ループ付きのソフトウェアステップパルスジェネレータ。 STEPGEN のセクションを参照してください

ENCODER

ソフトウェアベースのエンコーダーカウンター。 セクションエンコーダを参照してください

PID

比例/積分/微分制御ループ。 セクション PID を参照してください

SIGGEN

テスト用のサイン/コサイン/トライアングル/方形波ジェネレータ。 セクション SIGGEN を参照してください

SUPPLY

テスト用の簡単なソース

12.1.11 ハードウェアドライバー

HAL_AX5214H

AXIOM MEASUREMENT＆CONTROLAX5241H デジタル I / O ボード用のドライバー

HAL_GM

GENERAL MECHATRONICS GM6-PCI BOARD

HAL_M5I20

MESA ELECTRONICS 5I20 BOARD

HAL_MOTENC

VITAL SYSTEMS MOTENC-100 BOARD

HAL_PARPORT

PC パラレルポート

HAL_PPMC

PICO SYSTEMS FAMILY OF CONTROLLERS (PPMC, USC AND UPC)

HAL_STG

SERVO TO GO CARD (VERSION 1 & 2)

HAL_VTI

VIGILANT TECHNOLOGIES PCI ENCDAC-4 CONTROLLER

12.1.12 ツールとユーティリティ

HALCMD

構成と調整のためのコマンドラインツール。 セクション HALCMD を参照してください

HALMETER

HAL 信号用の便利なマルチメータ。 セクション HALMETER を参照してください。

HALSCOPE

HAL 信号用のフル機能のデジタルストレージオシロスコープ。 HALSCOPE のセクションを参照してください。

これらの各ビルディングブロックについては、後の章で詳しく説明します。

12.1.13 HAL のタイミングの問題

HAL が基づいていると私たちが言ったブラックボックス間の物理的な配線モデルとは異なり、2 つのピンを HAL 信号で接続するだけでは、物理的なケースの動作にははるかに及ばない。

真のリレーロジックは、相互に接続されたリレーで構成されており、接点が開閉すると、電流が即座に流れます（または停止します）。 他のコイルは状態などを変えるかもしれません、そしてそれはすべてただ起こります。 しかし、PLC スタイルのラダーロジックでは、そのようには機能しません。 通常、ラダーを 1 回通過する場合、各ラングは表示された順序で評価され、パスごとに 1 回だけ評価されます。 完璧な例は、コイルと直列の NC 接点を備えた単一のラングラダーです。 接点とコイルは同じリレーに属しています。

これが従来のリレーであった場合、コイルがオンになるとすぐに、接点が開き始め、コイルの電源がオフになります。 つまり、接点が再び閉じるなどです。リレーがブザーになります。

PLC の場合、PLC がラングの評価を開始するときにコイルがオフで接点が閉じていると、そのパスが終了するとコイルがオンになります。 コイルをオンにすると接点供給が開くという事実は、次のパスまで無視されます。 次のパスで、PLC は接点が開いていることを確認し、コイルの電源を切ります。 そのため、リレーはオンとオフをすばやく切り替えますが、PLC がラングを評価する頻度によって決定されます。

HAL では、関数はラングを評価するコードです。 実際、CLASSICLADDER の HAL 対応のリアルタイムバージョンは、まさにそれを行うための関数をエクスポートします。 一方、スレッドは特定の時間間隔で関数を実行するものです。 PLC にすべてのラングを 10 ミリ秒ごと、または 1 秒ごとに評価させることを選択できるのと同じように、異なる周期で HAL スレッドを定義できます。

あるスレッドを別のスレッドと区別するのは、スレッドが行うことではありません。つまり、どの関数がスレッドに接続されているかによって決まります。 本当の違いは、単にスレッドが実行される頻度です。

LINUXCNC では、50US のスレッドと 1MS のスレッドがある場合があります。 これらは BASE_PERIOD と SERVO_PERIOD に基づいて作成され、実際の時間は INI ファイルの値によって異なります。

次のステップは、各スレッドが何をする必要があるかを決定することです。 これらの決定の一部は、（ほぼ）どの LINUXCNC システムでも同じです。たとえば、MOTION-COMMAND-HANDLER は常にサーボスレッドに追加されます。

他の接続は積分器によって行われます。 これには、STG ドライバーのエンコーダー読み取りおよび DAC 書き込み関数をサーボスレッドにフックすることや、STEPGEN の関数をベーススレッドにフックすること、およびステップをポートに書き込むための PARPORT 関数が含まれる場合があります。

12.2 HAL Basics

このドキュメントは、HAL の基本へのリファレンスを提供します。

12.2.1 HAL コマンド

詳細については、HALCMD の MAN ページを参照してください。ターミナルウィンドウで MANHALCMD を実行します。

HAL 構成を確認し、ピンとパラメーターのステータスを確認するには、AXIS の[マシン]メニューの[HAL 構成]ウィンドウを使用します。 ピンのステータスを監視するには、[監視]タブを開き、監視する各ピンをクリックすると、監視ウィンドウに追加されます。

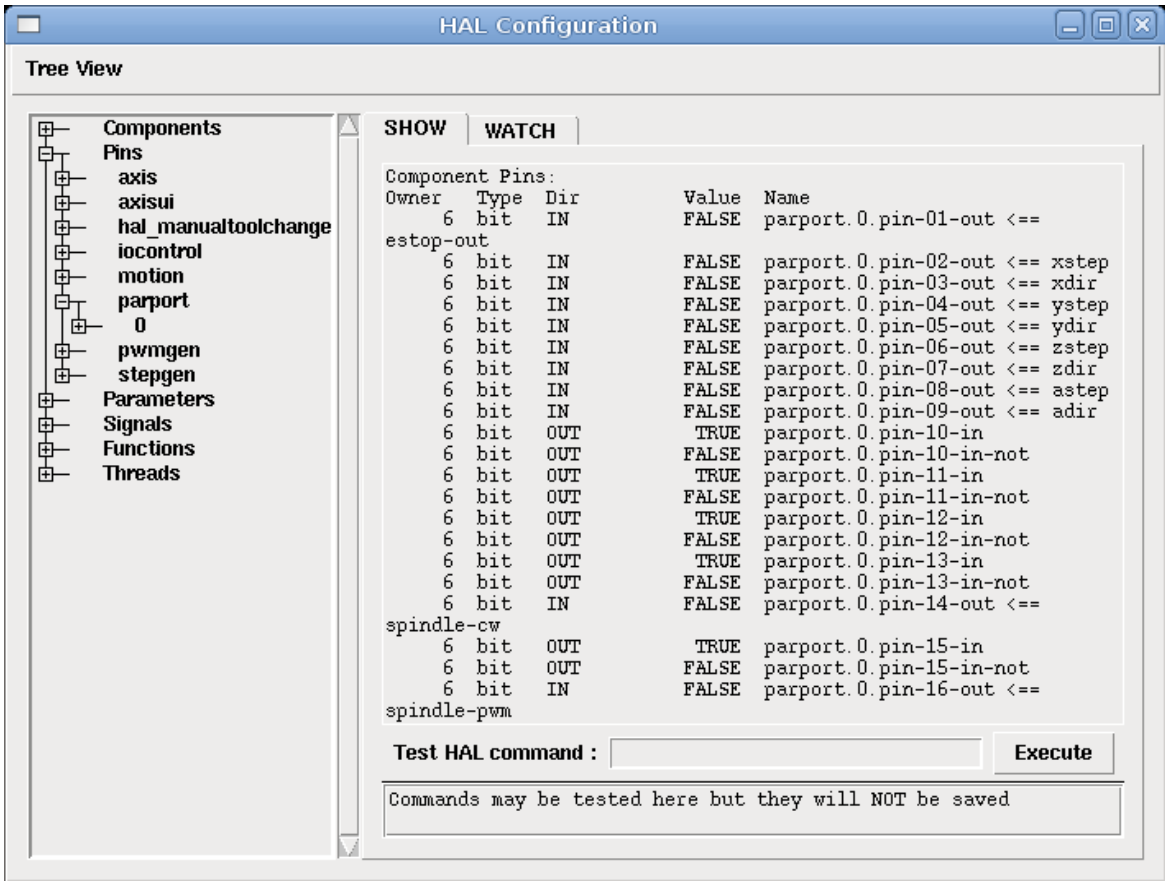


図 14-99

12.2.2 LOADRT

コマンド LOADRT は、リアルタイムの HAL コンポーネントをロードします。 スレッドの速度で更新するには、リアルタイムコンポーネント関数をスレッドに追加する必要があります。 ユーザースペースコンポーネントをリアルタイムスペースにロードすることはできません。

構文と例：

```
loadrt <component> <options>

loadrt mux4 count=1
```

12.2.3 ADDF

関数 FUNCTNAME をスレッド THREADNAME に追加します。 デフォルトでは、ファイル内の順序で関数を追加します。 位置が指定されている場合、スレッド内のそのスポットに関数を追加します。 負の位置とは、ねじの端に対する位置を意味します。 たとえば、1 はスレッドの開始、-1 はスレッドの終了、-3 は終了から 3 番目です。

一部の関数は、パーポートの読み取りおよび書き込み関数のように、特定の順序でロードすることが重要です。 関数名は通常、コンポーネント名に数字を加えたものです。 次の例では、コンポーネント OR2 がロードされ、SHOW 関数が OR2 関数の名前を示しています。

```
$ halrun
halcmd: loadrt or2
halcmd: show function
Exported Functions:
Owner     CodeAddr     Arg      FP     Users   Name
00004     f8bc5000     f8f950c8        NO      0       or2.0
```

関数をスレッドの速度で更新するには、HAL リアルタイムコンポーネントからスレッドに関数を追加する必要があります。

通常、この例に示すように 2 つのスレッドがあります。 一部のコンポーネントは浮動小数点演算を使用するため、浮動小数点演算をサポートするスレッドに追加する必要があります。 FP は、浮動小数点演算がそのスレッドでサポートされているかどうかを示します。

```
$ halrun
halcmd: loadrt motmod base_period_nsec=55555 servo_period_nsec=1000000
num_joints=3
halcmd: show thread
Realtime Threads:
Period     FP     Name            (      Time,    Max-Time )
995976     YES    servo-thread    (      0,               0 )
55332      NO     base-thread     (      0,               0 )
```

- ベーススレッド（高速スレッド）：このスレッドは、ステップパルスの作成、パラレルポートの読み取りと書き込みなど、高速応答が必要なアイテムを処理します。 浮動小数点演算はサポートしていません。
- サーボスレッド（低速スレッド）：このスレッドは、モーションコントローラー、CLASSICLADDER、モーションコマンドハンドラーなど、より遅い応答を許容できる項目を処理し、浮動小数点演算をサポートします。

構文と例：

```
addf <function> <thread>

addf mux4.0 servo-thread
```

NOTE

コンポーネントが浮動小数点スレッドを必要とする場合、通常は低速のサーボスレッドです。

12.2.4 LOADUSR

コマンド LOADUSR は、ユーザースペースの HAL コンポーネントをロードします。 ユーザースペースプログラムは独自の個別のプロセスであり、オプションでピンとパラメーターを介して他の HAL コンポーネントと通信します。 リアルタイムコンポーネントをユーザースペースにロードすることはできません。 フラグは、次の 1 つ以上である可能性があります。

-W

-WN<NAME> コンポーネントの準備が整うのを待ちます。 コンポーネントは、コマンドの最初の引数と同じ名前であると見なされます。
指定された<NAME>を持つコンポーネントを待機します。 これは、コンポーネントに名前オプションがある場合にのみ適用されます。

-W プログラムが終了するのを待つ

-I プログラムの戻り値を無視する（-W を使用）

-N コンポーネントの有効なオプションである場合は、コンポーネントに名前を付けます。

構文と例：

```
loadusr <component> <options>
loadusr halui
loadusr -Wn spindle gs2_vfd -n spindle
```

英語では、LOADUSR が名前スピンドルコンポーネント GS2_VFD 名前スピンドルを待機することを意味します。

12.2.5 NET

コマンド NET は、信号と 1 つまたは複数のピンの間に接続を作成します。 信号が存在しない場合、NET は新しい信号を作成します。 これにより、NEWSIG コマンドを使用する必要がなくなります。 オプションの方向矢印<=、=>、および<=>を使用すると、NET コマンドラインを読み取るときにロジックを簡単にたどることができ、NET コマンドでは使用されません。 方向矢印は、ピン名からスペースで区切る必要があります。

構文と例：

```
net signal-name pin-name <optional arrow> <optional second pin-name>

net home-x joint.0.home-sw-in <= parport.0.pin-11-in
```

上記の例では、HOME-X は信号名、JOINT.0.HOME-SW-IN は DIRECTION IN ピン、<=はオプションの方向矢印、PARPORT.0.PIN-11-IN は DIRECTIONOUT ピンです。 。 これは紛らわしいように思われるかもしれませんが、パラレルポートピンの入力ラベルと出力ラベルは、HAL でのピンの処理方法ではなく、ピンの物理的な動作方法を示しています。

次の規則に従っている場合、ピンを信号に接続できます。

- IN ピンは常に信号に接続できます
- 信号に OUT ピンがない限り、IO ピンを接続できます
- OUT ピンは、信号に他の OUT ピンまたは IO ピンがない場合にのみ接続できます。

上記の規則に従っている限り、同じ信号名を複数のネットコマンドで使用して追加のピンを接続できます。

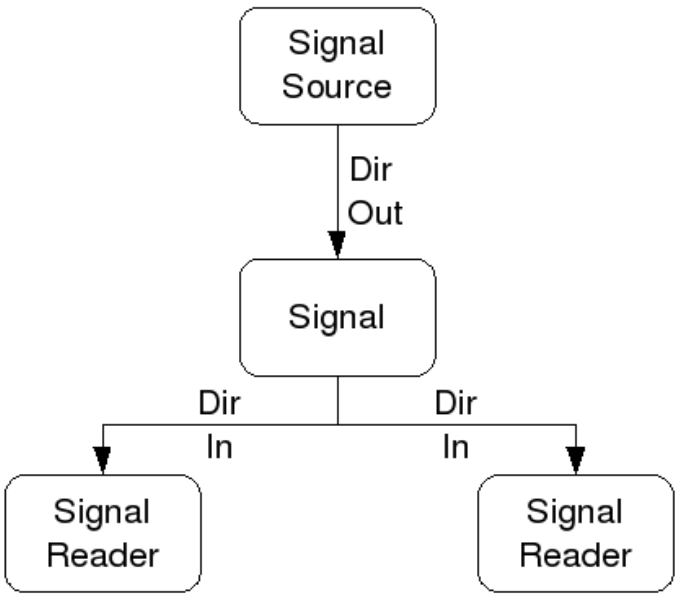


図 14-100

この例は、ソースが STEPGEN.0.OUT で、2 つのリーダー PARPORT.0.PIN-02-OUT と PARPORT.0.PIN-08-OUT がある信号 XSTEP を示しています。 基本的に、STEPGEN.0.OUT の値は信号 XSTEP に送信され、その値は次に PARPORT.0.PIN-02-OUT および PARPORT.0.PIN-08-OUT に送信されます。

```
#   signal    source          =>    destination          destination
Net xStep     stepgen.0.out   =>    parport.0.pin-02-out parport.0.pin-08-out
```

シグナル xSTEP には STEPGEN.0.OUT（ソース）の値が含まれているため、同じシグナルを再度使用して、値を別のリーダーに送信できます。 これを行うには、別の回線のリーダーで信号を使用するだけです。

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

I / O ピン ENCODER.N.INDEX-ENABLE のような I / O ピンは、コンポーネントで許可されているとおりに読み取りまたは設定できます。

12.2.6 SETP

コマンド SETP は、ピンまたはパラメータの値を設定します。 有効な値は、ピンまたはパラメータのタイプによって異なります。 データ型が一致しない場合はエラーになります。

一部のコンポーネントには、使用前に設定する必要のあるパラメーターがあります。 パラメータは、使用前または実行中に必要に応じて設定できます。 信号に接続されているピンには SETP を使用できません。

構文と例：

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

12.2.7 SETS

コマンド SETS は、信号の値を設定します。

構文と例：

```
sets <signal-name> <value>
net mysignal and2.0.in0 pyvcp.my-led
sets mysignal 1
```

次の場合はエラーになります。

- シグナル名が存在しません
- 信号にすでにライターがある場合
- 値が信号の正しいタイプでない場合

12.2.8 UNLINKP

コマンド UNLINKP は、接続された信号からピンのリンクを解除します。 コマンドを実行する前にピンに信号が接続されていなかった場合、何も起こりません。 UNLINKP コマンドは、トラブルシューティングに役立ちます。

構文と例：

```
unlinkp <pin-name>
```

```
unlinkp parport.0.pin-02-out
```

12.2.9 OBSOLETE COMMANDS

次のコマンドは減価償却されており、将来のバージョンから削除される可能性があります。 新しい設定では、NET コマンドを使用する必要があります。 これらのコマンドが含まれているため、古い構成でも引き続き機能します。

LINKSP コマンド LINKSP は、信号と 1 つのピンの間に接続を作成します。

構文と例：

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

LINKSP コマンドは NET コマンドに置き換えられました。

LINKPS コマンド LINKPS は、1 つのピンと 1 つの信号の間に接続を作成します。 これは LINKSP と同じですが、引数が逆になっています。

構文と例：

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

LINKPS コマンドは NET コマンドに置き換えられました。

NEWSIG コマンド NEWSIG は、<SIGNAME>という名前と<TYPE>のデータ型で新しい HAL 信号を作成します。 タイプはビット、S32、U32 または FLOAT である必要があります。 <SIGNAME>がすべて準備完了の場合はエラー。

構文と例：

```
newsig <signame> <type>
newsig Xstep bit
```

詳細については、HAL マニュアルまたは HALRUN のマニュアルページを参照してください。

12.2.10 HAL データ

12.2.11 Bit

ビット値はオンまたはオフです。

- ビット値= TRUE または 1 および FALSE または 0（TRUE、TRUE、TRUE はすべて有効です）

12.2.12 Float

FLOAT は浮動小数点数です。 つまり、小数点は必要に応じて移動できます。

- 浮動小数点値=約 53 ビットの解像度と 1000 ビットを超えるダイナミックレンジを持つ 64 ビット浮動小数点値。

浮動小数点数の詳細については、以下を参照してください。

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

12.2.13 s32

S32 番号は、負または正の値を持つことができる整数です。

- S32 値=整数-2147483648～2147483647

12.2.14 u32

U32 番号は、正の数のみの整数です。

- U32 値= 0 から 4294967295 までの整数

12.2.15 HAL ファイル

STEPPER CONFIG WIZARD を使用して構成を生成した場合、構成ディレクトリには最大 3 つの HAL ファイルがあります。

- MY-MILL.HAL（構成の名前が MY-MILL の場合）このファイルが最初にロードされるため、STEPPER CONFIGWIZARD を使用した場合は変更しないでください。
- CUSTOM.HAL このファイルは、GUI がロードされる前に次にロードされます。 これは、GUI がロードされる前にロードするカスタム HAL コマンドを配置する場所です。
- CUSTOM_POSTGUI.HAL このファイルは、GUI のロード後にロードされます。 ここに、GUI のロード後にロードするカスタム HAL コマンドを配置します。 PYVCP ウィジェットを使用する HAL コマンドはすべてここに配置する必要があります。

12.2.16 HAL パラメータ

各 HAL コンポーネントは、作成時に 2 つのパラメーターが自動的に追加されます。 これらのパラメーターを使用すると、コンポーネントの実行時間をスコープできます。

- TIME
- TMAX

時間は、関数の実行にかかった CPU サイクルの数です。

TMAX は、関数の実行にかかった CPU サイクルの最大数です。 TMAX は読み取り/書き込みパラメーターであるため、ユーザーはこれを 0 に設定して、関数の実行時の最初の初期化を取り除くことができます。

12.2.17 基本的なロジックコンポーネント

HAL には、いくつかのリアルタイムロジックコンポーネントが含まれています。 ロジックコンポーネントは、特定の入力に対する出力が何であるかを示す真理値表に従います。 通常、これらはビットマニピュレータであり、電気論理ゲートの真理値表に従います。

その他のコンポーネントについては、リアルタイムコンポーネントリストまたはマニュアルページを参照してください。

12.2.18 and2

AND2 コンポーネントは、2 つの入力とゲートです。 以下の真理値表は、入力の各組み合わせに基づく出力を示しています。

構文

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

関数

and2.n

Pins

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

真理値表

In0	In1	Out
False	False	False
True	False	False
False	Ture	False
True	True	True

12.2.19 not

NOT コンポーネントはビットインバーターです。

構文

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

関数

```
not.all
not.n
```

Pins

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

真理値表

In0	In1
True	False
False	True

関数

or2.n

Pins

12.2.20 or2

OR2 コンポーネントは 2 入力 OR ゲートです。

構文

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

関数

or2.n

Pins

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

真理値表

In0	In1	Out
True	False	True
True	True	True
False	Ture	True
False	False	False

12.2.21 xor2

XOR2 コンポーネントは、2 入力 XOR（排他的論理和）ゲートです。

構文

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

関数

```
xor2.n
```

Pins

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

真理値表

In0	In1	Out
True	False	True
True	True	False
False	Ture	True
False	False	False

12.2.22 ロジックの例

2 つの入力を 1 つの出力に接続する AND2 の例。

```
loadrt and2 count=1
addf and2.0 servo-thread
net my-sigin1 and2.0.in0 <= parport.0.pin-11-in
net my-sigin2 and2.0.in1 <= parport.0.pin-12-in
net both-on parport.0.pin-14-out <= and2.0.out
```

上記の例では、AND2 の 1 つのコピーがリアルタイム空間にロードされ、サーボスレッドに追加されます。 パラレルポートの次のピン 11 は、AND ゲートの IN0 ビットに接続されています。 次のピン 12 は、AND ゲートの IN1 ビットに接続されています。 最後に、AND2 出力ビットをパラレルポートのピン 14 に接続します。したがって、AND2 の真理値表に従って、ピン 11 とピン 12 がオンの場合、出力ピン 14 がオンになります。

12.3 HAL TWOPASS

12.3.1 TWOPASS

LINUXCNC 2.5 は、HAL ファイルのモジュール化と可読性に役立つ HAL 構成ファイルの TWOPASS 処理をサポートしています。（HAL ファイルは、HAL スタンザの LINUXCNC INI ファイルで[HAL] HALFILE = FILENAME として指定されます）。

通常、1 つ以上の HAL 構成ファイルのセットは、コンポーネントの複数のインスタンスを処理する可能性のあるカーネルモジュールをロードするために、単一の一意の LOADRT 行を使用する必要があります。 たとえば、セットアップの 3 つの異なる場所で 2 つの入力 AND ゲートコンポーネント（AND2）を使用する場合、次のように指定するためにどこかに 1 行が必要になります。

```
loadrt and2 count=3
```

その結果、コンポーネント AND2.0、AND2.1、および 2.2 が生成されます。

サポートされているコンポーネントに NAMES = OPTION を指定すると、構成が読みやすくなります。
例：

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

その結果、コンポーネント AA、AB、AC が生成されます。

コンポーネントを追加（または削除）するときは、コンポーネントに適用可能な単一の LOADRT ディレクティブを見つけて更新する必要があるため、コンポーネントとその名前を追跡することはメンテナンスの問題になる可能性があります。 TWOPASS 処理は、[HAL]セクションに INI ファイルパラメータを含めることで有効になります。

```
[HAL]
TWOPASS = anystring
```

ここで、「ANYSTRING」は NULL 以外の任意の文字列にすることができます。 この設定では、次のような複数の仕様を持つことができます。

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

これらのコマンドは、異なる HALFILE に表示される場合があります。 HALFILES は、INI ファイルに表示される順序で処理されます。

TWOPASS オプションは、デバッグ用の出力を追加するオプション（詳細）および一時ファイルの削除を防止するオプション（NODELETE）とともに指定できます。 オプションはコンマで区切ります。

例：

```
[HAL]
TWOPASS = on,verbose,nodelete
```

TWOPASS 処理では、すべての[HAL] HALFILES が最初に読み取られ、各モジュールの LOADRT ディレクティブの複数の出現が累積されます。 ユーザーコンポーネント（LOADUSR）は順番にロードされますが、最初のパスでは他の HAL コマンドは実行されません。

NOTE

ユーザーコンポーネントは、待機（-W）オプションを使用して、他のコマンドが実行される前にコンポーネントの準備ができていることを確認する必要があります。

最初のパスの後、リアルタイムモジュールは、COUNT =オプションを使用する場合は総数に等しい数で、または NAMES =オプションを使用する場合は指定されたすべての個別の名前で自動的にロード（LOADRT）されます。

次に、HALFILES で指定されている他のすべての HAL 命令を実行するために、2 番目のパスが作成されます。 コンポーネントの機能をスレッドの実行に関連付ける ADDF コマンドは、この 2 回目のパスで、他のコマンドと出現した順に実行されます。

COUNT =または NAMES =オプションのいずれかを使用できますが、これらは相互に排他的です。特定のモジュールに指定できるタイプは 1 つだけです。

TWOPASS 処理は、NAMES =オプションを使用する場合に最も効果的です。 このオプションを使用すると、ニーモニックまたは構成に関連する一意の名前を指定できます。 たとえば、微分成分を使用して各（X、Y、Z）座標の速度と加速度を推定する場合、COUNT =メソッドを使用すると、DDT.0、DDT.1、DDT.2 などの難解な成分名が得られます。

または、次のような NAMES =オプションを使用します。

```
loadrt ddt names=xvel,yvel,zvel
...
loadrt ddt names=xaccel,yaccel,zaccel
```

その結果、XVEL、YVEL、ZVEL、XACCEL、YACCEL、ZACCEL という名前のコンポーネントが作成されます。

ディストリビューションで提供される多くの COMP は、HALCOMPILE ユーティリティで作成され、NAMES =オプションをサポートします。 これらには、多くの HAL 構成の接着剤である一般的なロジックコンポーネントが含まれます。

HALCOMPILE ユーティリティを使用するユーザー作成のカンプは、NAMES =オプションも自動的にサポートします。 HALCOMPILE ユーティリティで生成されたコンプに加えて、他の多くのコンプが NAMES = OPTION をサポートしています。 NAMES =オプションをサポートするコンプには、AT_PID、ENCODER、ENCODER_RATIO、PID、SIGGEN、および SIM_ENCODER が含まれます。

12.3.2 Post GUI

一部の GUI は、GUI によって作成された HAL ピンを接続するために、GUI の開始後に処理されるハーフイルをサポートします。 TWOPASS 処理で POSTGUI ハーフイルを使用する場合は、POSTGUI ハーフイルによって追加されたコンポーネントのすべての LOADRT アイテムを、GUI の前に処理される別のハーフイルに含めます。 ADDF コマンドをファイルに含めることもできます。

例：

```
[HAL]
TWOPASS = on
HALFILE = file_1.hal
...
HALFILE = file_n.hal
HALFILE = file_with_all_loads_for_postgui.hal
...
POSTGUI_HALFILE = the_postgui_file.hal
```

12.3.3 .HAL ファイルの除外

TWOPASS 処理は、.HAL ファイルを同等の.TCL ファイルに変換し、HALTCL を使用して LOADRT および ADDF コマンドを検索し、それらの使用法を蓄積および統合します。 HAL コンポーネントジェネレーター（HALCOMPILE）によって受け入れられる単純な NAMES =（または COUNT =）パラメーターに準拠する LOADRT パラメーターが必要です。 特殊な HAL コンポーネントに含まれるより複雑なパラメータ項目は、適切に処理されない場合があります。

.HAL ファイルの任意の場所にマジックコメント行を含めることにより、.HAL ファイルを TWOPASS 処理から除外できます。 魔法のコメント行は、文字列#NOTWOPASS で始まる必要があります。 このマジックコメントで指定されたファイルは、-K（失敗した場合は続行）および-V（詳細）オプションを使用して HALCMD によって供給されます。

この除外規定は、問題を切り分けるため、または TWOPASS 処理を必要としない、またはその恩恵を受けない特別な HAL コンポーネントをロードするために使用できます。

通常、リアルタイムコンポーネントの LOADRT 順序は重要ではありませんが、特別なコンポーネントの LOADRT 順序は、そのような LOADRT ディレクティブを除外ファイルに配置することで強制できます。

NOTE

LOADRT ディレクティブの順序は通常重要ではありませんが、ADDF ディレクティブの順序は、サーボループコンポーネントの適切な動作にとって非常に重要であることがよくあります。

除外された.HAL ファイルの例：

```
$ cat twopass_excluded.hal
```

```
# The following magic comment causes this file to
# be excluded from twopass processing:
# NOTWOPASS
# debugging component with complex options:
loadrt mycomponent parm1="abc def" parm2=ghi
show pin mycomponent
# ordering special components
loadrt component_1
loadrt component_2
```

NOTE

マジックコメント内の大文字と小文字および空白は無視されます。 NAMES =または COUNT =パラメーター（通常は HALCOMPILE によって構築される）を使用するコンポーネントのロードは、TWOPASS 処理の利点を失うため、除外されたファイルでは使用しないでください。 シグナルを作成する HAL コマンド（NET）および実行順序を確立するコマンド（ADDF）は、除外されたファイルに配置しないでください。これは、ADDF コマンドの順序が重要になる可能性があるため、特に当てはまります。

12.3.4 例

シミュレータの TWOPASS の使用例は、次のディレクトリに含まれています。

```
configs/sim/axis/twopass/
configs/sim/axis/simtcl/
```

12.4 HAL チュートリアル

12.4.1 前書き

構成は理論からデバイス、つまり HAL デバイスに移行します。 コンピュータプログラミングを少し経験したことがある人のために、このセクションは HAL の HELLOWORLD です。 HALRUN を使用して、動作するシステムを作成できます。 これは、構成と調整のためのコマンドラインまたはテキストファイルツールです。 次の例は、そのセットアップと操作を示しています。

1.1.1.1 表記

HAL を実行していない限り、ターミナルコマンドはシステムプロンプトなしで表示されます。 ターミナルウィンドウは、メインの UBUNTU メニューバーの[アプリケーション/アクセサリ]にあります。

ターミナルコマンドの例

```
me@computer:~linuxcnc$ halrun
(will be shown like the following line)
```

```
halrun
(the halcmd: prompt will be shown when running HAL)
halcmd: loadrt debounce
halcmd: show pin
```

12.4.1.1 タブ補完

お使いのバージョンの HALCMD には、タブ補完が含まれている場合があります。 シェルのようにファイル名を完成させる代わりに、HAL 識別子を使用してコマンドを完成させます。 一意に一致するのに十分な文字を入力する必要があります。 HAL コマンドを開始した後、TAB キーを押してみてください。

タブ補完

```
halcmd: loa<TAB>
halcmd: load
halcmd: loadrt
halcmd: loadrt deb<TAB>
halcmd: loadrt debounce
```

12.4.1.2 RTAPI 環境

RTAPI は、REAL TIME APPLICATION PROGRAMMINGINTERFACE の略です。 多くの HAL コンポーネントはリアルタイムで動作し、すべての HAL コンポーネントはデータを共有メモリに格納するため、リアルタイムコンポーネントがデータにアクセスできます。 通常の LINUX は、HAL が必要とするリアルタイムプログラミングや共有メモリの種類をサポートしていません。 幸い、LINUX に必要な拡張機能を提供するリアルタイムオペレーティングシステム（RTOS）があります。 残念ながら、各 RTOS の動作は少し異なります。

これらの違いに対処するために、LINUXCNC チームは RTAPI を考案しました。これは、プログラムが RTOS と通信するための一貫した方法を提供します。 LINUXCNC の内部で作業したいプログラマーの場合は、API を理解するために LINUXCNC / SRC / RTAPI /RTAPI.H を学習することをお勧めします。 しかし、あなたが普通の人なら、RTAPI について知っておく必要があるのは、HAL で何かをする前に、RTAPI（および RTOS）をコンピュータのメモリにロードする必要があるということだけです。

12.4.2 簡単な例

1.1.1.1 コンポーネントのロード

このチュートリアルでは、LIVE CD が正常にインストールされ、RIP 2 を使用している場合は、RIP-ENVIRONMENT スクリプトを呼び出してシェルを準備したことを前提としています。

その場合、必要な RTOS および RTAPI モジュールをメモリにロードするだけです。 ターミナルウィンドウから次のコマンドを実行するだけです。

HAL の読み込み

```
cd linuxcnc
halrun
halcmd:
```

リアルタイム OS と RTAPI がロードされたら、最初の例に進むことができます。 プロンプトが HALCMD：として表示されていることに注意してください。 これは、後続のコマンドがシェルコマンドではなく、HAL コマンドとして解釈されるためです。

最初の例では、単純な信号発生器である SIGGEN と呼ばれる HAL コンポーネントを使用します。 SIGGEN コンポーネントの完全な説明は、このマニュアルの SIGGEN セクションにあります。 これはリアルタイムコンポーネントであり、LINUX カーネルモジュールとして実装されています。 SIGGEN をロードするには、HAL コマンド LOADRT を使用します。

SIGGEN を読込

```
halcmd: loadrt siggen
```

12.4.2.1 HAL の調査

モジュールがロードされたので、HAL の構成に使用されるコマンドラインツールである HALCMD を紹介します。 このチュートリアルでは、いくつかの HALCMD 機能を紹介します。 より完全な説明については、MAN HALCMD を試すか、このドキュメントの「HAL コマンド」セクションのリファレンスを参照してください。 最初の HALCMD 機能は SHOW コマンドです。 このコマンドは、HAL の現在の状態に関する情報を表示します。 インストールされているすべてのコンポーネントを表示するには：

コンポーネントを表示

```
halcmd: show comp
Loaded HAL Components:
ID Type Name PID State
3 RT siggen ready
2 User halcmd2177 2177 ready
```

HALCMD 自体は HAL コンポーネントであるため、常にリストに表示されます。 コンポーネントリストの HALCMD の後の番号は、プロセス ID です。 HALCMD の複数のコピーを同時に実行することが可能であるため（たとえば、異なるウィンドウで）、名前の末尾に PID を追加して一意にします。 このリストには、前の手順でインストールした SIGGEN コンポーネントも表示されます。 TYPE の下の RT は、SIGGEN がリアルタイムコンポーネントであることを

示しています。 [タイプ]の下の[ユーザー]は、それがユーザースペースコンポーネントであることを示します。

次に、SIGGEN が利用できるピンを見てみましょう。

ピンを表示

```
halcmd: show pin
Component Pins:
Owner   Type  Dir   Value  Name
3       float IN    1      siggen.0.amplitude
3       bit   OUT   FALSE  siggen.0.clock
3       float OUT   0      siggen.0.cosine
3       float IN    1      siggen.0.frequency
3       float IN    0      siggen.0.offset
3       float OUT   0      siggen.0.sawtooth
3       float OUT   0      siggen.0.sine
3       float OUT   0      siggen.0.square
3       float OUT   0      siggen.0.triangle
```

このコマンドは、現在の HAL のすべてのピンを表示します。 複雑なシステムには、数十または数百のピンが含まれる可能性があります。 しかし、現在、ピンは 9 つしかありません。 これらのピンのうち、8 つは浮動小数点で、1 つはビット（ブール値）です。 6 つは SIGGEN コンポーネントからデータを実行し、3 つは設定をコンポーネントに転送するために使用されます。 コンポーネントに含まれているコードをまだ実行していないため、一部のピンの値はゼロです。

次のステップは、パラメーターを確認することです。

パラメータを表示

```
halcmd: show param
Parameters:
Owner   Type  Dir   Value  Name
3       s32   RO    0      siggen.0.update.time
3       s32   RW    0      siggen.0.update.tmax
```

SHOW PARAM コマンドは、HAL 内のすべてのパラメーターを表示します。 現在、各パラメータには、コンポーネントがロードされたときに指定されたデフォルト値があります。 DIR というラベルの付いた列に注意してください。 -W というラベルの付いたパラメーターは、コンポーネント自体によって変更されることのない書き込み可能なパラメーターです。 代わりに、ユーザーがコンポーネントを制御するために変更することを目的としています。 これを行う方法については後で説明します。 R-というラベルの付いたパラメーターは読み取り専用パラメーターです。 コンポーネントによってのみ変更できます。 最後に、RW と

いうラベルの付いたパラメーターは読み取り/書き込みパラメーターです。 つまり、コンポーネントによって変更されますが、ユーザーが変更することもできます。 注：パラメータ SIGGEN.0.UPDATE.TIME および SIGGEN.0.UPDATE.TMAX はデバッグ用であり、これには含まれません。

ほとんどのリアルタイムコンポーネントは、1 つ以上の関数をエクスポートして、含まれているリアルタイムコードを実際に実行します。 SIGGEN がエクスポートした関数を見てみましょう。

関数を表示

```
halcmd: show funct
Exported Functions:
Owner         CodeAddr        Arg             FP       Users    Name
00003         f801b000                 fae820b8                YES      0
    siggen.0.update
```

SIGGEN コンポーネントは単一の関数をエクスポートしました。 浮動小数点が必要です。 現在、どのスレッドにもリンクされていないため、ユーザーはゼロです。

12.4.2.2 リアルタイムコードを実行させる

関数 SIGGEN.0.UPDATE に含まれているコードを実際に実行するには、リアルタイムスレッドが必要です。 新しいスレッドを作成するために使用されるスレッドと呼ばれるコンポーネント。 1 ミリ秒（1,000US または 1,000,000NS）の周期で TEST-THREAD というスレッドを作成しましょう。

```
halcmd: loadrt threads name1=test-thread period1=1000000
```

それがうまくいったかどうか見てみましょう：

スレッドを表示

```
halcmd: show thread
Realtime Threads:
Period        FP       Name    (        Time,    Max-Time )
999855        YES      test-thread (     0,       0        )
```

しました。 ハードウェアの制限により、周期は正確に 1,000,000 NS ではありませんが、ほぼ正しい速度で実行され、浮動小数点関数を処理できるスレッドがあります。 次のステップは、関数をスレッドに接続することです。

機能の追加

```
halcmd: addf siggen.0.update test-thread
```

これまで、HAL を確認するためだけに HALCMD を使用してきました。 ただし、今回は ADDF（関数の追加）コマンドを使用して、HAL 内の何かを実際に変更しました。 HALCMD に関数 SIGGEN.0.UPDATE をスレッド TEST-THREAD に追加するように指示しました。スレッドリストをもう一度見ると、成功したことがわかります。

```
halcmd: show thread

Realtime Threads:
Period      FP      Name                ( Time, Max-Time )
999855      YES     test-thread         ( 0, 0 )
1 siggen.0.update
```

SIGGEN コンポーネントが信号の生成を開始する前に、もう 1 つのステップが必要です。 HAL が最初に開始されたとき、スレッドは実際には実行されていません。 これは、リアルタイムコードが開始する前にシステムを完全に構成できるようにするためです。 構成に満足したら、次のようにリアルタイムコードを開始できます。

```
halcmd: start
```

これで信号発生器が動作しています。 その出力ピンを見てみましょう：

```
halcmd: show pin
Component Pins:
Owner     Type    Dir     Value    Name
3    float    IN      1          siggen.0.amplitude
3    bit      OUT     FALSE      siggen.0.clock
3    float    OUT     -0.1640929 siggen.0.cosine
3    float    IN      1          siggen.0.frequency
3    float    IN      0          siggen.0.offset
3    float    OUT     -0.4475303 siggen.0.sawtooth
3    float    OUT     0.9864449  siggen.0.sine
3    float    OUT     -1         siggen.0.square
3    float    OUT     -0.1049393 siggen.0.triangle
```

そしてもう一度見てみましょう：

```
halcmd: show pin
Component Pins:
Owner     Type    Dir     Value    Name
3    float    IN      1          siggen.0.amplitude
3    bit      OUT     FALSE      siggen.0.clock
3    float    OUT     0.0507619  siggen.0.cosine
3    float    IN      1          siggen.0.frequency
3    float    IN      0          siggen.0.offset
3    float    OUT     -0.516165  siggen.0.sawtooth
```

```
3    float    OUT    0.9987108 siggen.0.sine
3    float    OUT    -1        siggen.0.square
3    float    OUT    0.03232994 siggen.0.triangle
```

2 つの SHOWPIN コマンドをすばやく続けて実行したところ、出力がゼロではなくなっていることがわかります。 サイン、コサイン、のこぎり波、および三角形の出力は絶えず変化しています。 方形出力も機能していますが、サイクルごとに+1.0 から-1.0 に切り替わるだけです。

12.4.2.3 パラメータの変更

HAL の真の力は、物事を変えることができるということです。 たとえば、SETP コマンドを使用してパラメータの値を設定できます。信号発生器の振幅を 1.0 から 5.0 に変更しましょう。

SET PIN

```
halcmd: setp siggen.0.amplitude 5
```

パラメータとピンをもう一度確認してください

```
halcmd: show param
Parameters:
Owner        Type    Dir       Value    Name
3    s32     RO      1754      siggen.0.update.time
3    s32     RW      16997     siggen.0.update.tmax

halcmd: show pin
Component Pins:
Owner        Type    Dir       Value    Name
3    float   IN      5         siggen.0.amplitude
3    bit     OUT     FALSE     siggen.0.clock
3    float   OUT     0.8515425 siggen.0.cosine
3    float   IN      1         siggen.0.frequency
3    float   IN      0         siggen.0.offset
3    float   OUT     2.772382  siggen.0.sawtooth
3    float   OUT     -4.926954 siggen.0.sine
3    float   OUT     5         siggen.0.square
3    float   OUT     0.544764  siggen.0.triangle
```

パラメータ SIGGEN.0.AMPLITUDE の値が 5 に変更され、ピンの値が大きくなっていることに注意してください。

12.4.2.4 HAL 構成の保存

これまで HALCMD で行ったことのほとんどは、SHOW コマンドで物事を表示することだけでした。 しかし、2 つのコマンドは実際に物事を変えました。 HAL を使用してより複雑な

システムを設計するときは、多くのコマンドを使用して、必要な方法で構成します。 HAL には象の記憶があり、シャットダウンするまでその構成を保持します。 しかし、次回はどうですか？ システムを使用するたびに、一連のコマンドを手動で入力する必要はありません。 1 つのコマンドで HAL 全体の構成を保存できます。

SAVE

```
halcmd: save
# components
loadrt threads name1=test-thread period1=1000000
loadrt siggen
# pin aliases
# signals
# nets
# parameter values
setp siggen.0.update.tmax 14687
# realtime thread/function links
addf siggen.0.update test-thread
```

SAVE コマンドの出力は、一連の HAL コマンドです。 空の HAL から開始してこれらすべてのコマンドを実行すると、SAVE コマンドが発行されたときに存在していた構成が取得されます。 これらのコマンドを後で使用するために保存するには、出力をファイルにリダイレクトするだけです。

SAVE TO A FILE

```
halcmd: save all saved.hal
```

12.4.2.5 HALRUN を終了します

HAL セッションの終了が終了したら、HALCMD：プロンプトで EXIT と入力します。 これにより、システムプロンプトに戻り、HAL セッションが閉じられます。 HAL セッションをシャットダウンせずに、単にターミナルウィンドウを閉じないでください。

EXIT HAL

```
halcmd: exit
```

12.4.2.6 HAL 構成の復元

SAVED.HAL に保存されている HAL 構成を復元するには、これらの HAL コマンドをすべて実行する必要があります。 これを行うには、ファイルからコマンドを読み取る-F <ファイル名>と、コマンドの実行後に HALCMD プロンプトを表示する-I（大文字の I）を使用します。

RUN A SAVED FILE

```
halrun -I -f saved.hal
```

SAVED.HAL には開始コマンドがないことに注意してください。 もう一度発行する必要があります（または、SAVED.HAL を編集してそこに追加します）。

12.4.2.7 メモリから HAL を削除する

HAL セッションの予期しないシャットダウンが発生した場合、別のセッションを開始する前に HAL をアンロードする必要がある場合があります。 これを行うには、ターミナルウィンドウで次のコマンドを入力します。

REMOVING HAL

```
halrun -U
```

12.4.3 HAL メーター

グラフィカルインターフェイスを使用せずに、非常に複雑な HAL システムを構築できます。 しかし、あなたの仕事の結果を見ることには満足のいくものがあります。 HAL 用の最初で最も単純な GUI ツールは HALMETER です。 これは、便利な FLUKE マルチメーター（または古いタイマーの場合は SIMPSON アナログメーター）と同等の HAL である非常に単純なプログラムです。

再び SIGGEN コンポーネントを使用して、ハルメーターをチェックします。 前の例を終えたばかりの場合は、保存したファイルを使用して SIGGEN をロードできます。 そうでない場合は、以前と同じようにロードできます。

```
halrun
halcmd: loadrt siggen
halcmd: loadrt threads name1=test-thread period1=1000000
halcmd: addf siggen.0.update test-thread
halcmd: start
halcmd: setp siggen.0.amplitude 5
```

この時点で、SIGGEN コンポーネントがロードされて実行されています。 ハルメーターを始める時が来ました。

開始ハルメーター

```
halcmd: loadusr halmeter
```

最初に表示されるウィンドウは、[プローブするアイテムの選択]ウィンドウです。

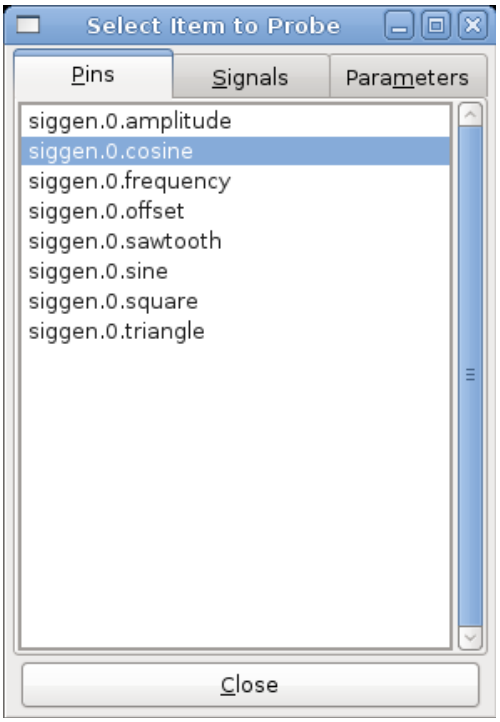


図 14-101

このダイアログには 3 つのタブがあります。 最初のタブには、システム内のすべての HAL ピンが表示されます。 2 つ目はすべての信号を表示し、3 つ目はすべてのパラメーターを表示します。 最初にピン SIGGEN.0.COSINE を確認したいので、それをクリックしてから[閉じる]ボタンをクリックします。 プローブ選択ダイアログが閉じ、メーターは次の図のようになります。

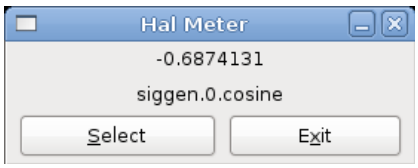


図 14-102

メーターの表示を変更するには、[選択]ボタンを押して[プローブするアイテムの選択]ウィンドウに戻ります。

SIGGEN が余弦波を生成すると、値が変化するのがわかります。 HALMETER は、1 秒間に約 5 回表示を更新します。 ハルメーターをシャットダウンするには、終了ボタンをクリックするだけです。

一度に複数のピン、信号、またはパラメーターを確認する場合は、より多くのハルメーターを開始できます。 ハルメーターウィンドウは意図的に非常に小さくしたので、一度にたくさんのウィンドウを画面に表示できます。

12.4.4 Stepgen の例

これまで、ロードした HAL コンポーネントは 1 つだけでした。 しかし、HAL の背後にある全体的な考え方は、複雑なシステムを構成するために、多数の単純なコンポーネントをロードして接続できるようにすることです。 次の例では、2 つのコンポーネントを使用します。

この新しい例の作成を開始する前に、白紙の状態から始めたいと思います。 前の例の 1 つを終了したばかりの場合は、すべてのコンポーネントを削除し、RTAPI および HAL ライブラリを再ロードする必要があります。

```
halcmd: exit
```

1.1.1.1 コンポーネントのインストール

次に、ステップパルスジェネレータコンポーネントをロードします。 このコンポーネントの詳細な説明については、インテグレータマニュアルの STEPGEN セクションを参照してください。 この例では、STEPGEN の速度制御タイプを使用します。 今のところ、詳細をスキップして、次のコマンドを実行するだけです。

```
halrun
halcmd: loadrt stepgen step_type=0,0 ctrl_type=v,v
halcmd: loadrt siggen
halcmd: loadrt threads name1=fast fp1=0 period1=50000 name2=slow period2=1000000
```

最初のコマンドは 2 つのステップジェネレーターをロードし、どちらもステッピングタイプ 0 を生成するように構成されています。2 番目のコマンドは旧友の SIGGEN をロードし、3 番目のコマンドは 2 つのスレッドを作成します。 1 ミリ秒。 高速スレッドは浮動小数点関数をサポートしていません。

以前と同様に、HALCMDSHOW を使用して HAL を確認できます。 今回は、以前よりもはるかに多くのピンとパラメーターがあります。

```
halcmd: show pin
Component Pins:
Owner   Type  Dir   Value  Name
4   float   IN    1      siggen.0.amplitude
4   bit     OUT   FALSE  siggen.0.clock
4   float   OUT   0      siggen.0.cosine
4   float   IN    1      siggen.0.frequency
4   float   IN    0      siggen.0.offset
4   float   OUT   0      siggen.0.sawtooth
4   float   OUT   0      siggen.0.sine
4   float   OUT   0      siggen.0.square
4 float OUT 0 siggen.0.triangle
```

```
3 s32 OUT 0 stepgen.0.counts
3 bit OUT FALSE stepgen.0.dir
3 bit IN FALSE stepgen.0.enable
3 float OUT 0 stepgen.0.position-fb
3 bit OUT FALSE stepgen.0.step
3 float IN 0 stepgen.0.velocity-cmd
3 s32 OUT 0 stepgen.1.counts
3 bit OUT FALSE stepgen.1.dir
3 bit IN FALSE stepgen.1.enable
3 float OUT 0 stepgen.1.position-fb
3 bit OUT FALSE stepgen.1.step
3 float IN 0 stepgen.1.velocity-cmd
halcmd: show param
Parameters:
Owner Type Dir Value Name
4 s32 RO 0 siggen.0.update.time
4 s32 RW 0 siggen.0.update.tmax
3 u32 RW 0x00000001 stepgen.0.dirhold
3 u32 RW 0x00000001 stepgen.0.dirsetup
3 float RO 0 stepgen.0.frequency
3 float RW 0 stepgen.0.maxaccel
3 float RW 0 stepgen.0.maxvel
3 float RW 1 stepgen.0.position-scale
3 s32 RO 0 stepgen.0.rawcounts
3 u32 RW 0x00000001 stepgen.0.steplen
3 u32 RW 0x00000001 stepgen.0.stepspace
3 u32 RW 0x00000001 stepgen.1.dirhold
3 u32 RW 0x00000001 stepgen.1.dirsetup
3 float RO 0 stepgen.1.frequency
3 float RW 0 stepgen.1.maxaccel
3 float RW 0 stepgen.1.maxvel
3 float RW 1 stepgen.1.position-scale
3 s32 RO 0 stepgen.1.rawcounts
3 u32 RW 0x00000001 stepgen.1.steplen
3 u32 RW 0x00000001 stepgen.1.stepspace
3 s32 RO 0 stepgen.capture-position.time
3 s32 RW 0 stepgen.capture-position.tmax
3 s32 RO 0 stepgen.make-pulses.time
3 s32 RW 0 stepgen.make-pulses.tmax
3 s32 RO 0 stepgen.update-freq.time
```

```
3 s32 RW 0 stepgen.update-freq.tmax
```

12.4.4.1 ピンと信号の接続

私たちが持っているのは、2 ステップのパルス発生器と信号発生器です。 次に、2 つのコンポーネントを接続するための HAL 信号を作成します。 2 ステップのパルス発生器が機械の X 軸と Y 軸を駆動していると仮定します。 テーブルを円で動かしたい。 これを行うには、X 軸にコサイン信号を送信し、Y 軸にサイン信号を送信します。 SIGGEN モジュールはサインとコサインを作成しますが、モジュールを相互に接続するためのワイヤーが必要です。 HAL では、ワイヤは信号と呼ばれます。 それらを 2 つ作成する必要があります。 これらは任意の名前で呼び出すことができます。この例では、X-VEL と Y-VEL になります。 信号 X-VELIS は、信号発生器の余弦出力から第 1 ステップパルス発生器の速度入力まで実行することを目的としています。 最初のステップは、信号を信号発生器の出力に接続することです。 信号をピンに接続するには、NET コマンドを使用します。

NET COMMAND

```
halcmd: net X-vel <= siggen.0.cosine
```

NET コマンドの効果を確認するために、信号を再度表示します。

```
halcmd: show sig
Signals:
Type          Value    Name    (linked to)
float         0        X-vel <== siggen.0.cosine
```

信号が 1 つまたは複数のピンに接続されている場合、SHOW コマンドは、信号名の直後のピンを一覧表示します。 矢印はデータフローの方向を示しています。この場合、データはピン SIGGEN.0.COSINE から信号 X-VEL に流れます。 次に、X-VEL をステップパルスジェネレータの速度入力に接続しましょう。

```
halcmd: net X-vel => stepgen.0.velocity-cmd
```

Y 軸信号 Y-VEL を接続することもできます。 これは、信号発生器の正弦波出力から 2 番目のステップのパルス発生器の入力まで実行することを目的としています。 次のコマンドは、2 つの NET コマンドが X-VEL に対して実行したことを 1 行で実行します。

```
halcmd: net Y-vel siggen.0.sine => stepgen.1.velocity-cmd
```

それでは、信号とそれに接続されているピンを最後に見てみましょう。

```
halcmd: show sig
Signals:
Type          Value    Name    (linked to)
float         0        X-vel <== siggen.0.cosine
==> stepgen.0.velocity-cmd
```

```
Float           0         Y-vel <== siggen.0.sine
==> stepgen.1.velocity-cmd
```

SHOW SIG コマンドを使用すると、データが HAL をどのように流れるかが正確にわかります。 たとえば、X-VEL 信号はピン SIGGEN.0.COSINE から送信され、ピン STEPGEN.0.VELOCITY-CMD に送信されます。

12.4.4.2 リアルタイム実行の設定-スレッドと関数

ワイヤーを流れるデータについて考えると、ピンと信号がかなり理解しやすくなります。 スレッドと関数はもう少し難しいです。 関数には、実際に物事を成し遂げるコンピューター命令が含まれています。 スレッドは、必要なときにこれらの命令を実行するために使用される方法です。 まず、私たちが利用できる機能を見てみましょう。

```
halcmd: show funct
Exported Functions:
Owner     CodeAddr    Arg         FP     Users   Name
00004     f9992000    fc731278    YES    0       siggen.0.update
00003     f998b20f    fc7310b8    YES    0       stepgen.capture-position
00003     f998b000    fc7310b8    NO     0       stepgen.make-pulses
00003     f998b307    fc7310b8    YES    0       stepgen.update-freq
```

一般に、各コンポーネントのドキュメントを参照して、その機能が何をするかを確認する必要があります。 この場合、関数 SIGGEN.0.UPDATE を使用して、信号発生器の出力を更新します。 実行されるたびに、サイン、コサイン、トライアングル、およびスクエアの出力の値が計算されます。 スムーズな信号を作成するには、特定の間隔で実行する必要があります。

他の 3 つの機能は、ステップパルスジェネレータに関連しています。

最初の STEPGEN.CAPTURE_POSITION は、位置フィードバックに使用されます。 生成されたステップパルスをカウントする内部カウンタの値をキャプチャします。 ステップの欠落がないと仮定すると、このカウンターはモーターの位置を示します。

ステップパルスジェネレータの主な機能は STEPGEN.MAKE_PULSES です。 MAKE_PULSES が実行されるたびに、ステップを実行する時間かどうかが決定され、実行される場合は、それに応じて出力が設定されます。 スムーズなステップパルスの場合、可能な限り頻繁に実行する必要があります。 MAKE_PULSES は非常に高速に実行する必要があるため、高度に最適化されており、わずかな計算しか実行しません。 他とは異なり、浮動小数点演算は必要ありません。

最後の関数 STEPGEN.UPDATE-FREQ は、周波数コマンドが変更された場合にのみ実行する必要があるスケーリングやその他の計算を実行する役割を果たします。

この例でこれが意味するのは、SIGGEN.0.UPDATE を適度な速度で実行して、サイン値とコサイン値を計算することです。 SIGGEN.0.UPDATE を実行した直後に、STEPGEN.UPDATE_FREQ を実行して、新しい値をステップパルスジェネレーターにロードします。 最後に、スムーズなパルスのために、STEPGEN.MAKE_PULSES をできるだけ速く実行する必要があります。 位置フィードバックを使用しないため、STEPGEN.CAPTURE_POSITION を実行する必要はまったくありません。

関数をスレッドに追加して実行します。 各スレッドは特定の速度で実行されます。 利用可能なスレッドを見てみましょう。

```
halcmd: show thread
Realtime Threads:
Period      FP      Name            ( Time, Max-Time )
996980      YES     slow            ( 0, 0 )
49849       NO      fast            ( 0, 0 )
```

2 つのスレッドは、スレッドをロードしたときに作成されました。 最初のものは低速で、ミリ秒ごとに実行され、浮動小数点関数を実行できます。 SIGGEN.0.UPDATE と STEPGEN.UPDATE_FREQ に使用します。 2 番目のスレッドは高速で、50 マイクロ秒ごとに実行され、浮動小数点をサポートしていません。 STEPGEN.MAKE_PULSES に使用します。 関数を適切なスレッドに接続するには、ADDF コマンドを使用します。 最初に関数を指定し、次にスレッドを指定します。

```
halcmd: addf siggen.0.update slow
halcmd: addf stepgen.update-freq slow
halcmd: addf stepgen.make-pulses fast
```

これらのコマンドを実行した後、SHOW THREAD コマンドを再度実行して、何が起こったかを確認できます。

```
halcmd: show thread
Realtime Threads:
Period      FP      Name            ( Time, Max-Time )
996980      YES     slow            ( 0, 0 )
1 siggen.0.update
2 stepgen.update-freq
49849       NO      fast            ( 0, 0 )
1 stepgen.make-pulses
```

これで、各スレッドの後に、関数が実行される順序で関数の名前が続きます。

12.4.4.3 パラメータの設定

HAL システムを開始する準備がほぼ整いました。ただし、まだいくつかのパラメータを調整する必要があります。デフォルトでは、SIGGEN コンポーネントは+1 から-1 までスイングする信号を生成します。この例では、テーブルの速度を+1 から-1 インチ/秒まで変化させる必要があります。ただし、ステップパルスジェネレータのスケーリングは完全に正しくありません。デフォルトでは、1.000 の入力で 1 秒あたり 1 ステップの出力周波数を生成します。1 秒あたり 1 ステップで、1 秒あたり 1 インチのテーブル移動が得られる可能性はほとんどありません。代わりに、1 インチあたり 5 回転の親ねじがあり、10 倍のマイクロステッピングで 1 回転あたり 200 ステップのステッパーに接続されていると仮定しましょう。したがって、ネジが 1 回転するのに 2000 ステップ、1 インチ移動するのに 5 回転かかります。つまり、全体的なスケーリングは 1 インチあたり 10000 ステップです。適切な出力を得るには、ステップパルスジェネレータへの速度入力に 10000 を掛ける必要があります。それがまさにパラメータ STEPGEN.N.VELOCITY-SCALE の目的です。この場合、X 軸と Y 軸の両方のスケーリングが同じであるため、両方のスケーリングパラメータを 10000 に設定します。

```
halcmd: setp stepgen.0.position-scale 10000
halcmd: setp stepgen.1.position-scale 10000
halcmd: setp stepgen.0.enable 1
halcmd: setp stepgen.1.enable 1
```

この速度スケーリングは、ピン STEPGEN.0.VELOCITY-CMD が 1.000 の場合、ステップジェネレーターが 10000 パルス/秒（10KHz）を生成することを意味します。 上記のモーターと親ねじを使用すると、軸は正確に毎秒 1.000 インチで移動します。 これは、HAL の重要な概念を示しています。スケーリングなどは可能な限り低いレベルで行われ、この場合はステップパルスジェネレーターで行われます。 内部信号 X-VEL は、1 秒あたりのインチ単位のテーブルの速度であり、SIGGEN などの他のコンポーネントは、スケーリングについてまったく認識していません（または気にしません）。 親ねじまたはモーターを変更した場合、ステップパルスジェネレーターのスケーリングパラメーターのみを変更します。

12.4.4.4 それを実行します！

これですべてが構成され、起動する準備が整いました。 最初の例と同じように、START コマンドを使用します。

```
halcmd: start
```

何も起こらないように見えますが、コンピューター内では、ステップパルスジェネレーターがステップパルスをクランクアウトし、毎秒 10KHz の順方向から 10KHz の逆方向に変化します。 このチュートリアルの後半では、これらの内部信号を取り出して実世界でモーターを実行する方法を説明しますが、最初にそれらを調べて、何が起こっているかを確認します。

12.4.5 Halscope

前の例では、いくつかの非常に興味深い信号が生成されます。 しかし、起こることの多くは、ハルメーターで見るには速すぎます。 HAL 内で何が起こっているかを詳しく調べるには、オシロスコープが必要です。 幸い、HAL には HALSCOPE と呼ばれるものがあります。

HALSCOPE には、カーネルモジュールとしてロードされるリアルタイム部分と、GUI と表示を提供するユーザー部分の 2 つの部分があります。 ただし、ユーザースペース部分はリアルタイム部分の読み込みを自動的に要求するため、これについて心配する必要はありません。 また、ディレクトリで初めて HALSCOPE を実行すると、AUTOSAVE.HALSCOPE ファイルを開くことができなかったという害のないメッセージが表示されることに注意してください。

HALSCOPE を起動

```
halcmd: loadusr halscope
halcmd: halscope: config file 'autosave.halscope' could not be opened
```

スコープの GUI ウィンドウが開き、すぐに次の図のようなリンクされていないリアルタイム関数ダイアログが表示されます。

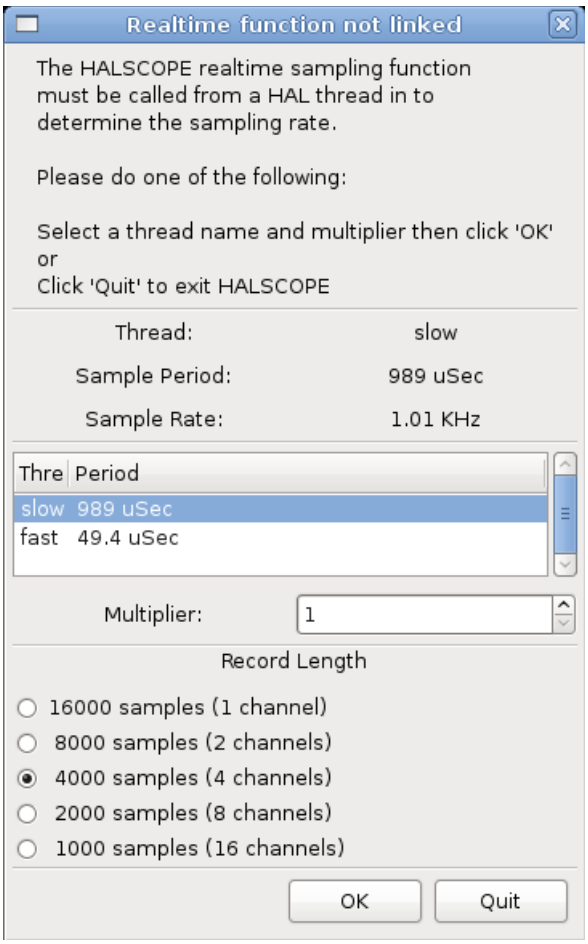


図 14-103

このダイアログでは、オシロスコープのサンプリングレートを設定します。 今のところ、ミリ秒に 1 回サンプリングしたいので、989 us スレッドをゆっくりクリックし、乗数を 1 のままにします。また、レコード長を 4000 サンプルのままにして、一度に最大 4 つのチャネルを使用できるようにします。 スレッドを選択して[OK]をクリックすると、ダイアログが消え、スコープウィンドウは次の図のようになります。

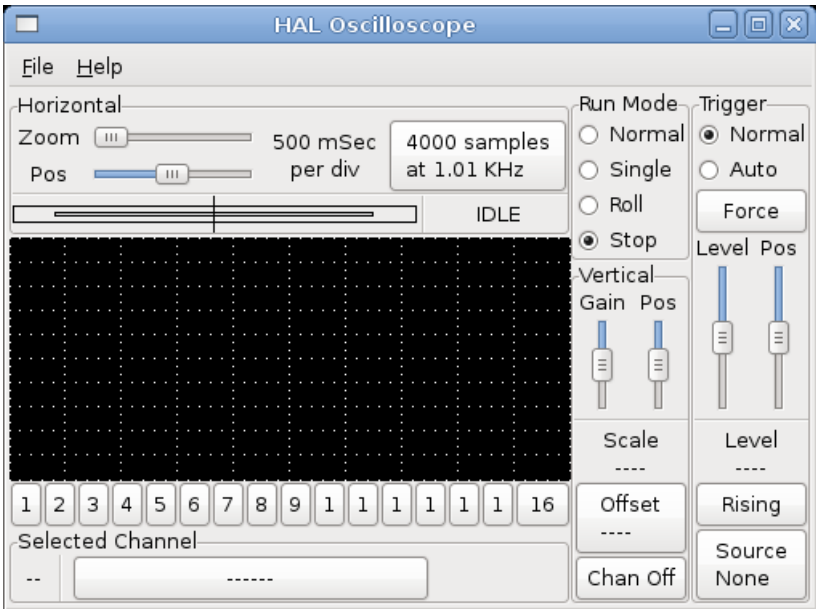


図 14-104

1.1.1.1 スコーププローブを接続する

この時点で、HALSCOPE を使用する準備が整いました。 サンプルレートとレコード長はすでに選択されているので、次のステップは何を見るべきかを決定することです。 これは、仮想スコーププローブを HAL にフックするのと同じです。 HALSCOPE には 16 のチャネルがありますが、一度に使用できる数はレコードの長さによって異なります。レコードに使用できるメモリは約 16,000 サンプルに固定されているため、チャネルが多いほどレコードが短くなります。

チャンネルボタンは、HALSCOPE 画面の下部にあります。 ボタン 1 をクリックすると、次の図に示すような[チャネルソースの選択]ダイアログが表示されます。 このダイアログは、HALMETER で使用されるダイアログと非常によく似ています。 前に定義したシグナルを確認したいので、[シグナル]タブをクリックすると、ダイアログに HAL 内のすべてのシグナルが表示されます（この例では 2 つのみ）。

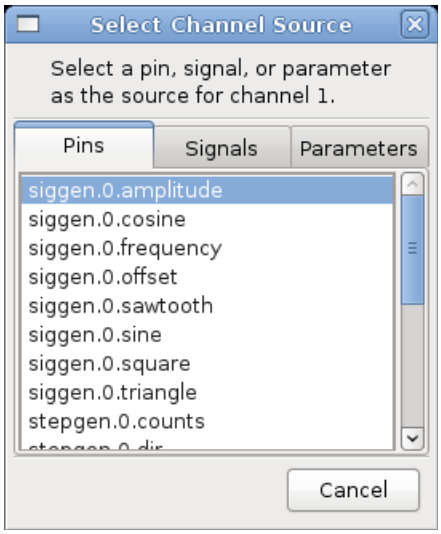


図 14-105

信号を選択するには、それをクリックするだけです。 この場合、チャネル 1 に信号 X-VEL を表示する必要があります。 [信号]タブをクリックしてから[X-VEL]をクリックすると、ダイアログが閉じ、チャネルが選択されます。

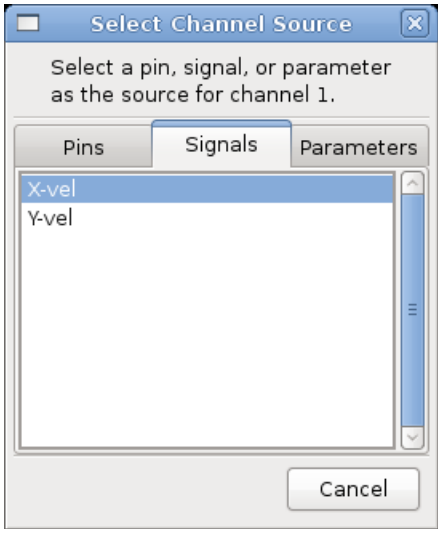


図 14-106

チャネル 1 ボタンが押されると、チャネル番号 1 と名前 X-VEL がボタンの行の下に表示されます。 そのディスプレイは常に選択されたチャンネルを示します-画面上に多くのチャンネルを置くことができますが、選択されたチャンネルは強調表示され、垂直位置やスケールなどのさまざまなコントロールは常に選択されたチャンネルで機能します。

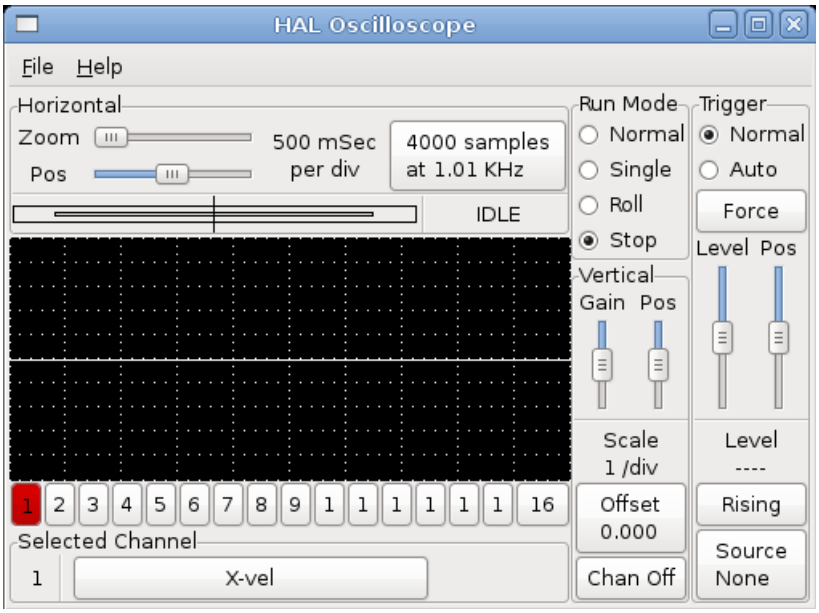


図 14-107

チャネル 2 に信号を追加するには、2 ボタンをクリックします。 ダイアログが表示されたら、[シグナル]タブをクリックし、[Y-VEL]をクリックします。 また、方形波と三角波の出力についても見ていきます。 これらのピンには信号が接続されていないため、代わりに[ピン]タブを使用します。 チャネル 3 の場合は SIGGEN.0.TRIANGLE を選択し、チャネル 4 の場合は SIGGEN.0.SQUARE を選択します。

12.4.5.1 最初の波形をキャプチャする

いくつかのプローブが HAL に接続されたので、次にいくつかの波形をキャプチャします。 スコープを開始するには、画面の[実行モード]セクション（右上）の[通常]ボタンをクリックします。 4000 サンプルのレコード長があり、1 秒あたり 1000 サンプルを取得しているため、HALSCOPE がバッファの半分を埋めるのに約 2 秒かかります。 その間、メイン画面のすぐ上にあるプログレスバーにバッファの充填が表示されます。 バッファが半分いっぱいになると、スコープはトリガーを待ちます。 まだ設定していないので、永遠に待ちます。 手動でトリガーするには、右上の[トリガー]セクションにある[強制]ボタンをクリックします。 バッファがいっぱいになる残りの部分が表示されたら、キャプチャされた波形が画面に表示されます。 結果は次の図のようになります。

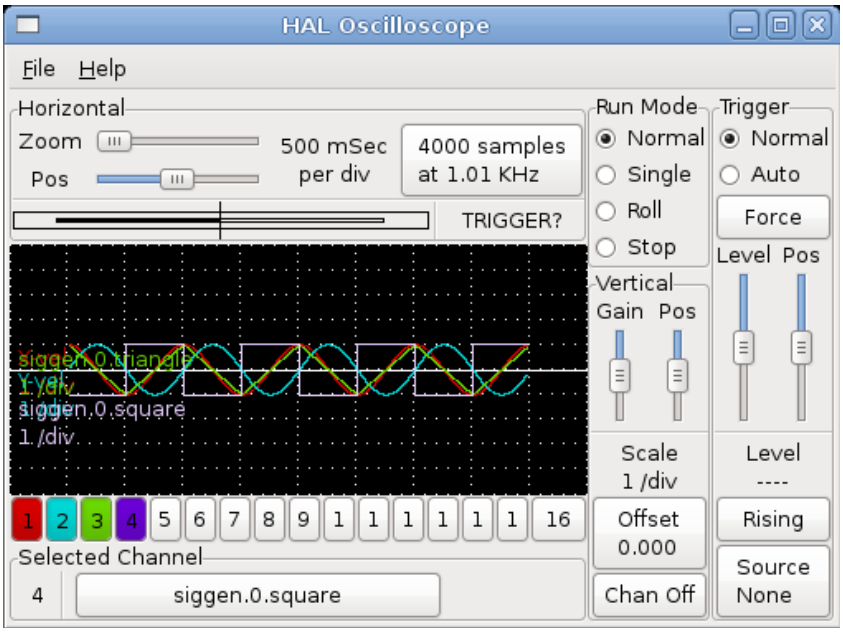


図 14-108

下部の[選択されたチャネル]ボックスは、紫色のトレースが現在選択されているチャネル 4 であり、ピン SIGGEN.0.SQUARE の値を表示していることを示しています。 チャネルボタン 1～3 をクリックして、他の 3 つのトレースを強調表示してみてください。

12.4.5.2 垂直方向の調整

4 つすべてが互いに重なり合っているため、トレースを区別するのはかなり困難です。 これを修正するには、画面の右側にあるボックスの垂直コントロールを使用します。 これらのコントロールは、現在選択されているチャネルに作用します。 ゲインを調整するときは、それが広い範囲をカバーしていることに注意してください。実際のスコープとは異なり、これは非常に小さい（ピコ単位）から非常に大きい（テラ単位）までの範囲の信号を表示できます。 位置コントロールは、表示されたトレースを画面の高さだけ上下に移動します。 より大きな調整には、オフセットボタンを使用する必要があります。

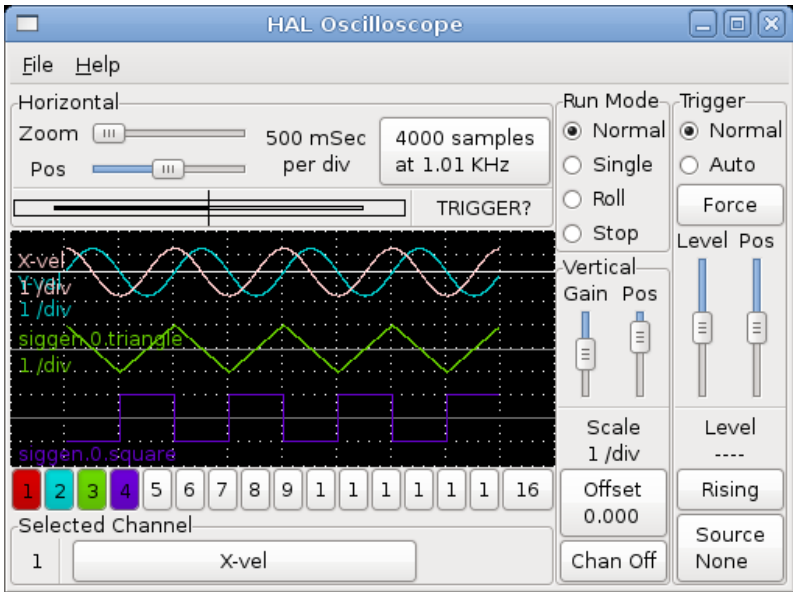


図 14-109

12.4.5.3 トリガー

[強制]ボタンを使用することは、スコープをトリガーするためのかなり満足のいく方法ではありません。 実際のトリガーを設定するには、右下の[ソース]ボタンをクリックします。[トリガーソース]ダイアログがポップアップ表示されます。これは、現在接続されているすべてのプローブのリストです。 トリガーに使用するプローブをクリックして選択します。この例では、次の図に示すように、チャネル 3 の三角波を使用します。

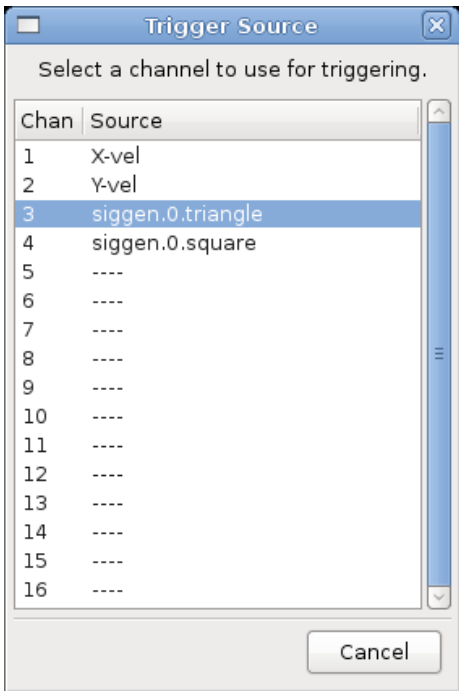


図 14-110

トリガーソースを設定した後、右端にある[トリガー]ボックスのスライダーを使用して、トリガーレベルとトリガー位置を調整できます。 レベルは画面の上から下に調整でき、ス

ライダーの下に表示されます。 位置は、レコード全体内のトリガーポイントの位置です。スライダーを完全に下げると、トリガーポイントはレコードの最後にあり、HALSCOPE はトリガーポイントの前に何が起こったかを表示します。 スライダーが完全に上にあるとき、トリガーポイントはレコードの先頭にあり、トリガーされた後に何が起こったかを表示します。 トリガーポイントは、画面上の進行状況ボックスに垂直線として表示されます。 トリガーレベル表示のすぐ下にあるボタンをクリックすると、トリガーの極性を変更できます。

垂直方向のコントロールとトリガーを調整したので、スコープの表示は次の図のようになります。

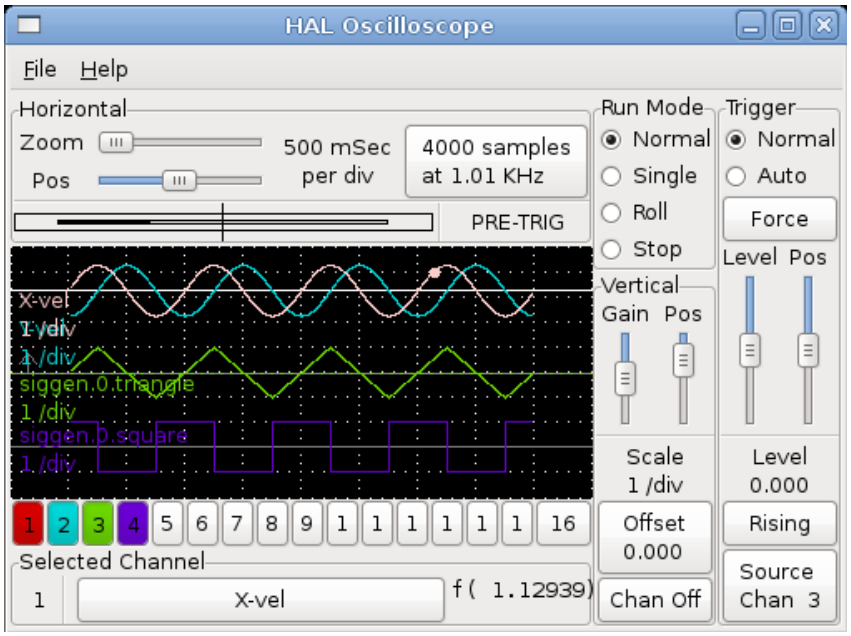


図 14-111

12.4.5.4 水平方向の調整

波形の一部を詳しく見るには、画面上部のズームスライダーを使用して波形を水平方向に拡大し、位置スライダーを使用してズームした波形のどの部分を表示するかを決定します。ただし、波形を単純に拡張するだけでは不十分な場合があり、サンプリングレートを上げる必要があります。 たとえば、この例で生成されている実際のステップパルスを確認したいと思います。 ステップパルスの長さはわずか 50us であるため、1KHz でのサンプリングは十分な速度ではありません。 サンプルレートを変更するには、サンプル数とサンプルレートを表示するボタンをクリックして、[サンプルレートの選択]ダイアログを表示します（図）。 この例では、50 us スレッドを高速でクリックします。これにより、約 20KHz のサンプルレートが得られます。 これで、約 4 秒に相当するデータを表示する代わりに、1 つのレコードは 20KHz で 4000 サンプル、つまり約 0.20 秒になります。

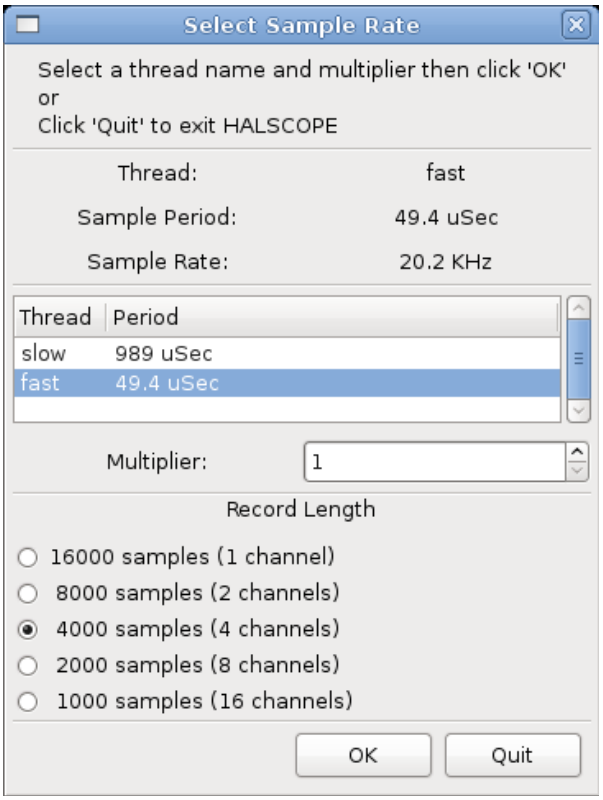


図 14-112

12.4.5.5 その他のチャンネル

それでは、ステップパルスを見てみましょう。 HALSCOPE には 16 チャンネルがありますが、この例では一度に 4 チャンネルしか使用していません。これ以上チャネルを選択する前に、いくつかをオフにする必要があります。チャネル 2 ボタンをクリックしてから、[垂直]ボックスの下部にある[チャンオフ]ボタンをクリックします。次に、チャネル 3 をクリックし、オフになっている場合はオフにして、チャネル 4 についても同じようにします。チャネルがオフになっていても、接続先を記憶しているため、トリガーソースとしてチャネル 3 を引き続き使用します。新しいチャネルを追加するには、チャネル 5 を選択し、ピン STEPGEN.0.DIR を選択してから、チャネル 6 を選択し、STEPGEN.0.STEP を選択します。次に、実行モード NORMAL をクリックしてスコープを開始し、水平ズームを 1 目盛りあたり 5 ミリ秒に調整します。速度コマンド（チャネル 1）がゼロに近づくと、ステップパルスが遅くなり、方向ピンの状態が変化して、ステップパルスが再び速くなります。速度コマンドの変化をよりよく確認するために、チャネル 1 のゲインを 1 目盛りあたり約 20 ミリに増やすことをお勧めします。結果は次の図のようになります。

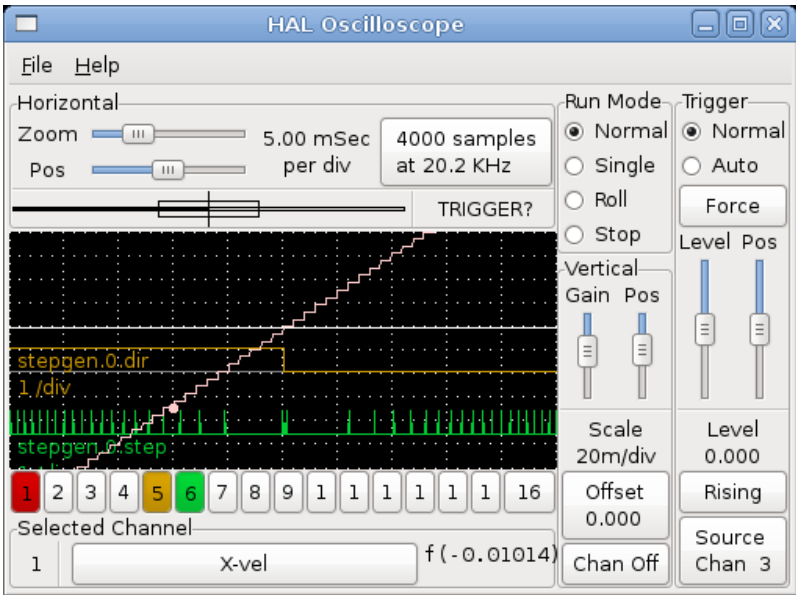


図 14-113

12.4.5.6 その他のサンプル

より多くのサンプルを一度に記録する場合は、リアルタイムで再起動し、キャプチャするサンプルの数を示す数値引数を使用して HALSCOPE をロードします。

```
halcmd: loadusr halscope 80000
```

SCOPE_RT コンポーネントがまだロードされていない場合、HALSCOPE はそれをロードし、合計 80000 のサンプルを要求するため、一度に 4 つのチャネルをサンプリングすると、チャネルごとに 20000 のサンプルが存在します。（SCOPE_RT がすでにロードされている場合、HALSCOPE への数値引数は効果がありません）。

12.5 一般的なリファレンス

12.5.1 一般的な命名規則

一貫した命名規則により、HAL がはるかに使いやすくなります。 たとえば、すべてのエンコーダドライバが同じものを提供した場合 ピンのセットと同じ名前を付けて、あるタイプのエンコーダドライバから別のタイプに簡単に変更できるようにします。 残念ながら、多くのオープンソースプロジェクトと同様に、HAL は設計されたものと単純に進化したものの組み合わせです。 結果として、 多くの矛盾があります。 このセクションでは、いくつかの規則を定義することでその問題に対処しようとしますが、おそらく すべてのモジュールがそれらに従うように変換されるまでしばらくお待ちください。

HALCMD およびその他の低レベルの HAL ユーティリティは、HAL 名を内部構造のない単一のエンティティとして扱います。 ただし、ほとんどのモジュールには暗黙の構造があります。 たとえ

ば、ボードには複数の機能ブロックがあり、各ブロックには複数のチャネルがあり、各チャネルには1つ以上のピンがあります。 これにより、ディレクトリツリーに似た構造になります。 HALCMD はツリー構造を認識しませんが、命名規則を適切に選択すると、関連するアイテムがグループ化されます（名前が並べ替えられるため）。 さらに、名前が必要な情報を提供する場合、そのような構造を認識するように高レベルのツールを設計できます。 これを行うには、すべての HAL コンポーネントが次のルールに従う必要があります。

- ドット（「。」）は、雇用のレベルを区切ります。 これは、ファイル名のスラッシュ（“ /”）に似ています。
- ハイフン（「-」）は、階層の同じレベルにある単語またはフィールドを区切ります。
- HAL コンポーネントでは、アンダースコアや「MIXEDCASE」を使用しないでください。
- 名前には小文字と数字のみを使用してください。

12.5.2 ハードウェアドライバーの命名規則

12.5.3 ピン/パラメータ名

ハードウェアドライバーは、次のように、ピンまたはパラメーター名を構成するために5つのフィールド（3つのレベル）を使用する必要があります。

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

個々のフィールドは次のとおりです。

<DEVICE-NAME>

ドライバーが使用することを目的としたデバイス。 これはほとんどの場合、ある種のインターフェースボードですが、他の可能性もあります。

<DEVICE-NUM>

コンピュータに複数のサーボボード、パラレルポート、またはその他のハードウェアデバイスをインストールすることができます。 デバイス番号は特定のデバイスを識別します。 デバイス番号は0から始まり、増分します。

<IO-TYPE>

ほとんどのデバイスは、複数のタイプの I / O を提供します。 単純なパラレルポートでさえ、デジタル入力とデジタル出力の両方を備えています。 より複雑なボードには、デジタル入力と出力、エンコーダカウンター、PWM またはステップパルスジェネレーター、アナログ-デジタルコンバーター、デジタル-アナログコンバーター、またはその他の独自の機能があります。 I / O タイプは、ピンまたはパラメータが関連付けられている I / O の種類を識別するために使用されます。 理想的には、同じ I / O タイプを実装するドライバーは、デバイスが大きく異なる場合でも、一貫したピンとパラメーターのセットと同じ動作を提供する必要

があります。 たとえば、デバイスに関係なく、HAL の内部から見た場合、すべてのデジタル入力は同じように動作する必要があります。

<CHAN-NUM>

事実上すべての I / O デバイスには複数のチャネルがあり、チャネル番号はそのうちの 1 つを識別します。 デバイス番号と同様に、チャネル番号はゼロから始まり、増分します。3 複数のデバイスがインストールされている場合、追加のデバイスのチャネル番号はゼロから始まります。 チャネル番号が 9 より大きい場合は、チャネル番号を 2 桁にし、ソート順を維持するために 10 未満の番号には先行ゼロを付ける必要があります。 一部のモジュールには、複数のチャネルに影響を与えるピンやパラメータがあります。 たとえば、PWM ジェネレーターには 4 つの独立した「デューティサイクル」入力を持つ 4 つのチャネルがありますが、4 つのチャネルすべてを制御する 1 つの「周波数」パラメーターがあります（ハードウェアの制限による）。 周波数パラメータは、チャネル番号として「0-3」を使用する必要があります。

<SPECIFIC-NAME>

個々の I / O チャネルには、HAL ピンが 1 つだけ関連付けられている場合がありますが、ほとんどの場合、複数のピンが関連付けられています。 たとえば、デジタル入力には 2 つのピンがあり、1 つは物理ピンの状態であり、もう 1 つは同じものが反転されています。 これにより、コンフィギュレータはアクティブハイ入力とアクティブロー入力のどちらかを選択できます。 ほとんどの IO タイプには、ドライバーが実装する必要のあるピンとパラメーターの標準セット（「正規インターフェース」と呼ばれる）があります。 カノニカルインターフェイスについては、カノニカルデバイスインターフェイスの章で説明しています。

例

MOTENC.0.ENCODER.2.POSITION

-最初の MOTENC ボード上の 3 番目のエンコーダチャネルの位置出力。

STG.0.DIN.03.IN

—最初の SERVO-TO-GO ボードの 4 番目のデジタル入力の状態。

PPMC.0.PWM.00-03.FREQUENCY

-最初の PICOSYSTEMS PPMC ボードの PWM チャネル 0〜3 に使用される搬送周波数。

12.5.4 関数名

ハードウェアドライバーには通常、ハードウェアを読み取って HAL ピンを更新するものと、HAL ピンからのデータを使用してハードウェアに書き込むものの 2 種類の HAL 機能しかありません。 それらは次のように名前を付ける必要があります。

<DEVICE-NAME>-<DEVICE-NUM>.<IO-TYPE>-<CHAN-NUM-RANGE>.READ|WRITE

<DEVICE-NAME>

ピンとパラメータに使用されるものと同じです。

<DEVICE-NUM>

関数がアクセスする特定のデバイス。

<IO-TYPE>

オプション。 関数は、ボード上のすべてのI / Oにアクセスする場合もあれば、特定のタイプにのみアクセスする場合もあります。 たとえば、エンコーダカウンタの読み取りとデジタルI / Oの読み取りのための独立した機能が存在する場合があります。 このような独立した関数が存在する場合、<IO-TYPE>フィールドはそれらがアクセスするI / Oのタイプを識別します。 単一の関数がボードによって提供されるすべてのI / Oを読み取る場合、<IO-TYPE>は使用されません。

NOTE

ドライバープログラマーへの注意：割り込み可能で独立したスレッドで機能できる場合を除いて、異なるI / Oタイプに個別の関数を実装しないでください。 エンコーダーの読み取りを中断し、デジタル入力を読み取り、エンコーダーの読み取りを再開すると問題が発生する場合は、すべてを実行する単一の関数を実装します。

<CHAN-NUM-RANGE>

オプション。 <IO-TYPE> I / Oがグループに分割され、さまざまな機能によってアクセスされる場合にのみ使用されます。

READ|WRITE

関数がハードウェアを読み取るか、ハードウェアに書き込むかを示します。

例

MOTENC.0.ENCODER.READ

関数がハードウェアを読み取るか、ハードウェアに書き込むかを示します。—最初のMOTENCボード上のすべてのエンコーダーを読み取ります。

GENERIC8255.0.DIN.09-15.READ

-最初の汎用8255ベースのデジタルI / Oボードの2番目の8ビットポートを読み取ります。

PPMC.0.WRITE

-最初のPICOSYSTEMS PPMCボードにすべての出力（ステップジェネレーター、PWM、DAC、およびデジタル）を書き込みます。

12.6 コアコンポーネント

MAN ページの MOTION（9）も参照してください。

12.6.1 モーション

これらのピンとパラメーターは、リアルタイム MOTMOD モジュールによって作成されます。 このモジュールは、LINUXCNC のモーションプランナーに HAL インターフェースを提供します。 基本的に、MOTMOD はウェイポイントのリストを取り込んで、モータードライブに供給されるジョイント位置の適切にブレンドされた制約制限付きストリームを生成します。

オプションで、デジタル I / O の数は NUM_DIO で設定されます。 アナログ I / O の数は NUM_AIO で設定されます。 デフォルトはそれぞれ 4 です。

AXIS.L および JOINT.N 'で始まるピン名とパラメーター名は、モーションコントローラー機能によって読み取られ、更新されます。

モーションは MOTMOD コマンドでロードされます。 キンはモーションの前にロードする必要があります。

```
loadrt motmod [base_period_nsec=period] [servo_period_nsec=period]
[traj_period_nsec=period] [num_joints=[0-9] ([num_dio=1-64] num_aio=1-16]) ([  -
unlock_joints_mask=0xNN])
```

- BASE_PERIOD_NSEC = 50000-基本タスク期間（ナノ秒単位）。 これは、マシンで最速のスレッドです。

NOTE

サーボベースのシステムでは、通常、BASE_PERIOD_NSEC が SERVO_PERIOD_NSEC よりも小さい理由はありません。 ソフトウェアステップ生成を備えたマシンでは、BASE_PERIOD_NSEC が 1 秒あたりの最大ステップ数を決定します。 長いステップ長とステップスペースの要件がない場合、絶対最大ステップレートは BASE_PERIOD_NSEC ごとに 1 ステップです。 したがって、上記の BASE_PERIOD_NSEC は、毎秒 20,000 ステップの絶対最大ステップレートを提供します。 50,000 NS（50 US）は、かなり控えめな値です。 使用可能な最小値は、レイテンシーテストの結果、必要なステップ長、およびプロセッサー速度に関連しています。 低すぎる BASE_PERIOD_NSEC を選択すると、「予期しないリアルタイム遅延」メッセージ、ロックアップ、または自発的な再起動が発生する可能性があります。

- SERVO_PERIOD_NSEC = 1000000-これは、ナノ秒単位のサーボタスク期間です。 この値は、BASE_PERIOD_NSEC の整数倍に丸められます。 この期間は、ステッピングモーターをベースにしたシステムでも使用されます。

  これは、新しいモーター位置が計算され、エラーがチェックされ、PID 出力値が更新されるなどの速度です。 ほとんどのシステムでは、この値を変更する必要はありません。 低レベルモーションプランナーの更新レートです。

- TRAJ_PERIOD_NSEC = 100000-これは、ナノ秒単位の TRAJECTORYPLANNER タスク期間です。この値は、SERVO_PERIOD_NSEC の整数倍に丸められます。 異常な運動学を備えたマシン（ヘキサポッドなど）を除いて、この値を SERVO_PERIOD_NSEC より大きくする理由はありません。

12.6.2 オプション

必要なデジタル I / O の数がデフォルトの 4 を超える場合は、MOTMOD のロード時に NUM_DIO オプションを使用して、最大 64 のデジタル I / O を追加できます。

必要なアナログ I / O の数がデフォルトの 4 を超える場合は、MOTMOD のロード時に NUM_AIO オプションを使用して、最大 16 のアナログ I / O を追加できます。

UNLOCK_JOINTS_MASK パラメーターは、ロッキングインデクサー（通常はロータリー）として使用されるジョイントのピンを作成するために使用されます。 マスクビットはジョイントを選択します。 マスクの LSB はジョイント 0 を選択します。例：UNLOCK_JOINTS_MASK = 0x38 はジョイント 3,4,5 を選択します

12.6.3 Pins

これらのピン、パラメーター、および関数は、リアルタイム MOTMOD モジュールによって作成されます。

- motion.adaptive-feed -（FLOAT、IN）適応送りが M52 P1 で有効になっている場合、指令された速度にこの値が乗算されます。 この効果は、NML レベルのフィードオーバーライド値と MOTION.FEED-HOLD で乗法的です。 LINUXCNC のバージョン 2.8 以降、負の適応フィード値を使用して G コードパスを逆に実行することが可能です。
- motion.analog-in-00 -（FLOAT、IN）これらのピン（構成されている場合は 00、01、02、03 以上）は M66 によって制御されます。
- motion.analog-out-00 -（FLOAT、OUT）これらのピン（構成されている場合は 00、01、02、03 以上）は M67 または M68 によって制御されます。
- motion.coord-error -（ビット、アウト）モーションがソフト制限を超えるなどのエラーに遭遇した場合は TRUE
- motion.coord-mode -（ビット、出力）モーションが TELEOP モードではなく、協調モードの場合は TRUE
- motion.current-vel -（FLOAT、OUT）ユーザー単位/秒での現在のツール速度。
- motion.digital-in-00 -（ビット、入力）これらのピン（構成されている場合は 00、01、02、03 以上）は M62-65 によって制御されます。
- motion.digital-out-00 -（ビット、出力）これらのピン（構成されている場合は 00、01、02、03 以上）は M62-65 によって制御されます。

- motion.distance-to-go -（FLOAT、OUT）現在の移動で残っている距離。
- motion.enable -（ビット、イン）このビットが FALSE に駆動されると、モーションが停止し、マシンはマシンオフ状態になり、オペレーターにメッセージが表示されます。 通常の動作の場合、このビットを TRUE に駆動します。
- motion.feed-hold - (bit, in) When Feed Stop Control is enabled with M53 P1, and this bit is TRUE, the feed rate is set to 0.
- motion.feed-inhibit -（BIT、IN）このビットが TRUE の場合、送り速度は 0 に設定されます。これは、スピンドル同期移動中、移動が終了するまで遅延します。
- motion.in-position -（ビット、出力）マシンが所定の位置にある場合は TRUE。
- motion.motion-enabled -（ビット、出力）マシンオン状態の場合は TRUE。
- motion.motion-type -（S32、OUT）これらの値は SRC / EMC / NML_INTF /MOTION_TYPES.H からのものです

  – 0：アイドル（モーションなし）

  – 1：トラバース

  – 2：線形フィード

  – 3：アークフィード

  – 4：工具交換

  – 5：プロービング

  – 6：回転軸のインデックス付け
- motion.on-soft-limit -（ビット、出力）マシンがソフト制限にある場合は TRUE。
- motion.probe-input -（ビット、入力）G38.N は、このピンの値を使用して、プローブがいつ接触したかを判別します。 プローブ接点が閉じている（接触している）場合は TRUE、プローブ接点が開いている場合は FALSE。
- motion.program-line -（S32、OUT）実行中の現在のプログラム行。 実行されていない場合、またはシングルステップ中に行間でゼロ。
- motion.requested-vel -（FLOAT、OUT）現在要求されている速度（ユーザー単位/秒）。 この値は、G コードファイルの F ワード設定であり、マシンの速度と加速の制限に対応するために削減される可能性があります。 このピンの値は、送りオーバーライドやその他の調整を反映していません。
- spindle.0.at-speed -（ビット、イン）モーションは、次の条件下で、このピンが TRUE になるまで一時停止します。各スピンドルの開始または速度変更後の最初の送りの移動前。 スピンドル同期移動のすべてのチェーンの開始前。 また、CSS モードの場合は、フィードへの移

行が急速に進むたびに。 この入力を使用して、カットを開始する前にスピンドルの速度が上がっていること、またはCSSモードの旋盤スピンドルが大径から小径のパスの後で大径で次のパスを開始する前に減速したことを確認できます。 多くのVFDは速度出力を備えています。 それ以外の場合は、要求されたスピンドル速度と実際のスピンドル速度を比較することにより、HALニアコンポーネントを使用してこの信号を簡単に生成できます。

- spindle.N.brake -（ビット、出力）スピンドルブレーキを適用する必要がある場合はTRUE。
- spindle.N.forward -（ビット、出力）スピンドルが前方に回転する必要がある場合はTRUE。
- spindle.N.index-enable -（ビット、I / O）スピンドル同期移動を正しく動作させるには、このピンをスピンドルエンコーダのインデックスイネーブルピンに接続する必要があります。
- spindle.N.inhibit -（ビット、イン）このビットがTRUEの場合、スピンドル速度は0に設定されます。
- spindle.N.on -（ビット、出力）スピンドルが回転する必要がある場合はTRUE。
- spindle.N.reverse -（ビット、アウト）スピンドルが後方に回転する必要がある場合はTRUE
- spindle.N.revs -（FLOAT、IN）スピンドル同期移動を正しく動作させるには、この信号をスピンドルエンコーダの位置ピンにフックする必要があります。 スピンドルエンコーダの位置は、スピンドルが時計回り（M3）方向に回転するたびにスピンドル回転数が1.0ずつ増加するようにスケーリングする必要があります。
- spindle.N.speed-in -（FLOAT、IN）実際のスピンドル速度のフィードバック（1秒あたりの回転数）。 これは、1回転あたりの送り動作（G95）で使用されます。 スピンドルエンコーダドライバに速度出力がない場合は、DDTコンポーネントを介してスピンドル位置を送信することにより、適切な速度出力を生成できます。 スピンドルエンコーダがない場合は、SPINDLE.N.SPEED-OUT-RPSをループバックできます。
- spindle.N.speed-out -（フロート、アウト）1分あたりの回転数でのコマンドスピンドル速度。 スピンドルフォワード（M3）の場合は正、スピンドルリバース（M4）の場合は負。
- spindle.N.speed-out-abs -（FLOAT、OUT）コマンドされたスピンドル速度（1分あたりの回転数）。 これは常に正の数になります。
- spindle.N.speed-out-rps -（FLOAT、OUT）コマンドされたスピンドル速度（1秒あたりの回転数）。 スピンドルフォワード（M3）の場合は正、スピンドルリバース（M4）の場合は負。
- spindle.N.speed-out-rps-abs -（FLOAT、OUT）コマンドされたスピンドル速度（1秒あたりの回転数）。 これは常に正の数になります。

- motion.teleop-mode -（ビット、出力）モーションが調整モードではなく、テレオプモードの場合は TRUE
- motion.tooloffset.x . . . motion.tooloffset.w -（FLOAT、OUT、軸ごとに 1 つ）は、有効なツールオフセットを示します。 ツールテーブル（G43 アクティブ）から取得することも、GCODE（G43.1 アクティブ）から取得することもできます。
- spindle.N.orient-angle-（FLOAT、OUT）M19 に必要なスピンドルの向き。 M19R ワードパラメータの値と[RS274NGC] ORIENT_OFFSETINI パラメータの値。
- spindle.N.orient-mode-（S32、OUT）必要なスピンドル回転モード M19。 デフォルトは 0 です。
- spindle.N.orient-（OUT、BIT）スピンドルオリエントサイクルの開始を示します。 M19 によって設定されます。 M3、M4、M5 のいずれかによってクリアされます。 スピンドルオリエントが TRUE のときにスピンドルオリエントフォールトがゼロでない場合、M19 コマンドはエラーメッセージで失敗します。
- spindle.N.is-oriented-（入力、ビット）スピンドル方向の確認ピン。 オリエントサイクルを完了します。 主軸方向がアサートされたときに主軸方向が TRUE の場合、主軸方向ピンがクリアされ、主軸ロックピンがアサートされます。 また、スピンドルブレーキピンがアサートされます。
- spindle.N.orient-fault-（S32、IN）オリエントサイクルの障害コード入力。 ゼロ以外の値を指定すると、オリエントサイクルが中止されます。
- spindle.N.lock-（ビット、出力）スピンドルは完全なピンを方向付けます。 M3、M4、M5 のいずれかによってクリアされます。

M19 オリエントスピンドルの HAL ピンの使用法概念的には、スピンドルは次のいずれかのモードにあります。

- 回転モード（デフォルト）
- 目的の方向モードを検索する
- オリエンテーション完了モード。

M19 が実行されると、主軸は目的の方向の検索に変わり、主軸の N.ORIENTHAL ピンがアサートされます。 目的の目標位置は、SPINDLE.N.ORIENT-ANGLE および SPINDLE.N.ORIENT-FWD ピンによって指定され、M19R および P パラメーターによって駆動されます。

HAL サポートロジックは、スピンドルを目的の位置に移動することにより、スピンドルに反応することが期待されます。 これが完了すると、HAL ロジックは、SPINDLE.N。が方向付けられたピンをアサートすることによってこれを確認することが期待されます。

次に、モーションは、SPINDLE.N.ORIENT ピンをディアサートし、SPINDLE.N.LOCKED ピンをアサートして、方向完了モードを示すことにより、これを確認します。 また、スピンドルを上げます。N。ブレーキピン。 これで、スピンドルはオリエンテーション完了モードになります。

SPINDLE.N.ORIENT が真であり、スピンドル.N。がまだアサートされていない場合、スピンドル.N.ORIENT-FAULT ピンの値がゼロ以外の場合、M19 コマンドは中止され、障害コードを含むメッセージが表示されます。 が表示され、モーションキューがフラッシュされます。 主軸は回転モードに戻ります。

また、M3、M4、または M5 コマンドはいずれも、目的の方向または方向完了モードの検索をキャンセルします。 これは、スピンドル指向ピンとスピンドルロックピンの両方をディアサートすることで示されます。

スピンドルオリエントモードピンは M19P ワードを反映し、次のように解釈されます。

- 0：最小の角度移動のために時計回りまたは反時計回りに回転します
- 1：常に時計回りに回転する
- 2：常に反時計回りに回転します

これは、スピンドルエンコーダ位置、スピンドルオリエント角度、およびスピンドルオリエントモードに基づいて PID コマンド値を提供するオリエント HAL コンポーネントとともに使用できます。

12.6.4 パラメーター

これらのパラメータの多くはデバッグの補助として機能し、いつでも変更または削除される可能性があります。

- MOTION-COMMAND-HANDLER.TIME-（S32、RO）
- MOTION-COMMAND-HANDLER.TMAX-（S32、RW）
- MOTION-CONTROLLER.TIME-（S32、RO）
- MOTION-CONTROLLER.TMAX-（S32、RW）
- MOTION.DEBUG-BIT-0-（ビット、RO）これはデバッグの目的で使用されます。
- MOTION.DEBUG-BIT-1-（ビット、RO）これはデバッグの目的で使用されます。
- MOTION.DEBUG-FLOAT-0-（FLOAT、RO）これはデバッグの目的で使用されます。
- MOTION.DEBUG-FLOAT-1-（FLOAT、RO）これはデバッグの目的で使用されます。
- MOTION.DEBUG-FLOAT-2-（FLOAT、RO）これはデバッグの目的で使用されます。
- MOTION.DEBUG-FLOAT-3-（FLOAT、RO）これはデバッグの目的で使用されます。

- MOTION.DEBUG-S32-0-（S32、RO）これはデバッグの目的で使用されます。
- MOTION.DEBUG-S32-1-（S32、RO）これはデバッグの目的で使用されます。
- MOTION.SERVO.LAST-PERIOD-（U32、RO）サーボスレッドの呼び出し間の CPU サイクル数。 通常、この数値を CPU 速度で割ると、時間が秒単位で示され、リアルタイムモーションコントローラーがタイミング制約を満たしているかどうかを判断するために使用できます。
- MOTION.SERVO.LAST-PERIOD-NS-（フロート、RO）

12.6.5 関数

通常、これらの機能は両方とも、示されている順序でサーボスレッドに追加されます。

- MOTION-COMMAND-HANDLER-ユーザースペースからのモーションコマンドを処理します
- モーションコントローラー-LINUXCNC モーションコントローラーを実行します

12.6.6 軸とジョイントのピンとパラメータ

これらのピンとパラメーターは、リアルタイム MOTMOD モジュールによって作成されます。 [トリビアルキネマティクスマシンでは、ジョイントと軸の間に 1 対 1 の対応があります。]これらは、モーションコントローラ機能によって読み取られ、更新されます。

ピンとパラメータの詳細については、モーションのマニュアルページ MOTION（9）を参照してください。

12.6.7 Iocontrol

IOCONTROL-NML I / O コマンドを受け入れ、ユーザースペースで HAL と対話します。

信号はユーザースペースでオンとオフが切り替えられます。厳密なタイミング要件がある場合、または単により多くの I / O が必要な場合は、代わりにモーションによって提供されるリアルタイム同期 I / O の使用を検討してください。

12.6.8 Pins

- IOCONTROL.0.COOLANT-FLOOD-（ビット、アウト）フラッドクーラントが要求された場合は TRUE。
- IOCONTROL.0.COOLANT-MIST-（ビット、アウト）ミストクーラントが要求された場合は TRUE。
- IOCONTROL.0.EMC-ENABLE-IN-（ビット、イン）外部非常停止条件が存在する場合は FALSE で駆動する必要があります。

- IOCONTROL.0.LUBE-（ビット、出力）潤滑油が指令された場合は TRUE。
- IOCONTROL.0.LUBE_LEVEL-（ビット、イン）潤滑油レベルが十分に高い場合は TRUE に駆動する必要があります。
- IOCONTROL.0.TOOL-CHANGE-（ビット、出力）ツールの変更が要求された場合は TRUE。
- IOCONTROL.0.TOOL-CHANGED-（ビット、イン）ツールの変更が完了したら TRUE を駆動する必要があります。
- IOCONTROL.0.TOOL-NUMBER-（S32、OUT）現在のツール番号。
- IOCONTROL.0.TOOL-PREP-NUMBER-（S32、OUT）RS274NGCT ワードからの次のツールの番号。
- IOCONTROL.0.TOOL-PREPARE-（ビット、出力）ツールの準備が要求された場合は TRUE。
- IOCONTROL.0.TOOL-PREPARED-（ビット、イン）ツールの準備が完了したら TRUE を駆動する必要があります。
- IOCONTROL.0.USER-ENABLE-OUT-（ビット、出力）内部非常停止条件が存在する場合は FALSE。
- IOCONTROL.0.USER-REQUEST-ENABLE-（ビット、出力）ユーザーが非常停止のクリアを要求した場合は TRUE。

12.6.9 INI の設定

多くの INI 設定が HAL 入力ピンとして利用可能になっています。

12.6.10 Pins

N はジョイント番号を示し、L は軸文字を示します

- INI.N.FERROR-（FLOAT、IN）[JOINT_N] FERROR
- INI.N.MIN_FERROR-（FLOAT、IN）[JOINT_N] MIN_FERROR
- INI.N.BACKLASH - (FLOAT, IN) [JOINT_N]BACKLASH
- INI.N.MIN_LIMIT - (FLOAT, IN) [JOINT_N]MIN_LIMIT
- INI.N.MAX_LIMIT - (FLOAT, IN) [JOINT_N]MAX_LIMIT
- INI.N.MAX_VELOCITY - (FLOAT, IN) [JOINT_N]MAX_VELOCITY
- INI.N.MAX_ACCELERATION - (FLOAT, IN) [JOINT_N]MAX_ACCELERATION
- INI.N.HOME - (FLOAT, IN) [JOINT_N]HOME
- INI.N.HOME_OFFSET - (FLOAT, IN) [JOINT_N]HOME_OFFSET

- INI.N.HOME_OFFSET - (S32, IN) [JOINT_N]HOME_SEQUENCE
- INI.L.MIN_LIMIT - (FLOAT, IN) [AXIS_L]MIN_LIMIT
- INI.L.MAX_LIMIT - (FLOAT, IN) [AXIS_L]MAX_LIMIT
- INI.L.MAX_VELOCITY - (FLOAT, IN) [AXIS_L]MAX_VELOCITY
- INI.L.MAX_ACCELERATION - (FLOAT, IN) [AXIS_L]MAX_ACCELERATION

note

軸ごとの MIN_LIMIT ピンと MAX_LIMIT ピンは、原点復帰後も継続的に受け入れられます。 軸ごとの FERROR ピンと MIN_FERROR ピンは、マシンがオンで、所定の位置にないときに尊重されます。 軸ごとの MAX_VELOCITY ピンと MAX_ACCELERATION ピンは、マシンがオンで MOTION_STATE がフリー（ホーミングまたはジョギング）のときにサンプリングされますが、プログラムの実行中（自動モード）または MDI モードのときはサンプリングされません。 したがって、プログラムの実行中にピン値を変更しても、プログラムが停止して MOTION_STATE が再び解放されるまで効果はありません。

- ini.traj_arc_blend_enable - (bit, in) [TRAJ]ARC_BLEND_ENABLE
- ini.traj_arc_blend_fallback_enable - (bit, in) [TRAJ]ARC_BLEND_FALLBACK_ENABLE
- ini.traj_arc_blend_gap_cycles - (float, in) [TRAJ]ARC_BLEND_GAP_CYCLES
- ini.traj_arc_blend_optimization_depth - (float, in) [TRAJ]ARC_BLEND_OPTIMIZATION_DEPTH
- ini.traj_arc_blend_ramp_freq - (float, in) [TRAJ]ARC_BLEND_RAMP_FREQ

note

TRAJ_ARC_BLEND ピンは継続的にサンプリングされますが、プログラムの実行中にピン値を変更しても、コマンドのキューイングのためにすぐには効果がない場合があります。

- ini.traj_default_acceleration - (float, in) [TRAJ]DEFAULT_ACCELERATION
- ini.traj_default_velocity - (float, in) [TRAJ]DEFAULT_VELOCITY
- ini.traj_max_acceleration - (float, in) [TRAJ]MAX_ACCELERATION

12.7 標準的なデバイスインターフェイス

12.7.1 前書き

次のセクションでは、「正規デバイス」によって提供されるピン、パラメーター、および機能を示します。 すべての HAL デバイスドライバーは、同じピンとパラメーターを提供し、同じ動作を実装する必要があります。

正規デバイスには、<IO-TYPE>フィールドと<SPECIFIC-NAME>フィールドのみが定義されていることに注意してください。 <DEVICE-NAME>、<DEVICE-NUM>、および<CHAN-NUM>フィールドは、実際のデバイスの特性に基づいて設定されます。

12.7.2 デジタル入力

正規のデジタル入力（I / O タイプフィールド：DIGIN）は非常に単純です。

12.7.3 Pins

- （ビット）IN-ハードウェア入力の状態。
- （ビット）IN-NOT-入力の反転状態。

12.7.4 パラメーター

- NONE

12.7.5 関数

- （機能）読み取り-ハードウェアを読み取り、HAL ピンではなく設定します。

12.7.6 デジタル出力

正規のデジタル出力（I / O タイプフィールド：DIGOUT）も非常に単純です。

12.7.7 Pins

- （ビット）出力-ハードウェア出力に書き込まれる（場合によっては反転される）値。

12.7.8 パラメーター

- （ビット）INVERT：TRUE の場合、ハードウェアに書き込む前に OUT が反転されます。

12.7.9 関数

- （機能）書き込み-読み取りと反転を行い、それに応じてハードウェア出力を設定します。

12.7.10 アナログ入力

標準的なアナログ入力（I / O タイプ：ADCIN）。 これは、アナログからデジタルへのコンバーターに使用されることが期待されています。 値の連続範囲への電圧。

12.7.11 Pins

- （FLOAT）値-スケールおよびオフセットパラメータに従ってスケーリングされたハードウェア読み取り値。 値=（（入力読み取り値、ハードウェアに依存する単位）*スケール）-オフセット

12.7.12 パラメーター

- （フロート）スケール-入力電圧（または電流）は、値に出力される前にスケールで乗算されます。
- （フロート）オフセット-これは、スケールマルチプライヤが適用された後、ハードウェア入力電圧（または電流）から差し引かれます。

- （FLOAT）BIT_WEIGHT：最下位ビット（LSB）の値。 これは事実上、入力読み取り値の粒度です。
- （FLOAT）HW_OFFSET：入力ピンに 0 ボルトが印加されたときに入力に存在する値。

12.7.13 関数

- （機能）読み取り-このアナログ入力チャネルの値を読み取ります。 これは、個々のチャネルの読み取りに使用される場合もあれば、すべてのチャネルが読み取られる場合もあります。

12.7.14 アナログ出力

正規のアナログ出力（I / O タイプ：ADCOUT）。 これは、多かれ少なかれ連続的な値の範囲を出力できるあらゆる種類のハードウェアを対象としています。 例としては、デジタル-アナログコンバーターまたは PWM ジェネレーターがあります。

12.7.15 Pins

- （FLOAT）VALUE-書き込まれる値。 ハードウェアに出力される実際の値は、スケールとオフセットのパラメーターによって異なります。
- （ビット）ENABLE：FALSE の場合、値ピンに関係なく、ハードウェアに 0 を出力します。

12.7.16 パラメーター

- （FLOAT）OFFSET-これは、ハードウェアが更新される前に値に追加されます
- （FLOAT）SCALE -これは、値ピンに 1 を入力すると、アナログ出力ピンが 1 ボルトを読み取るように設定する必要があります。
- （FLOAT）HIGH_LIMIT（オプション）-ハードウェアに出力する値を計算するときに、値+オフセットが HIGH_LIMIT より大きい場合、代わりに HIGH_LIMIT が使用されます。
- （FLOAT）LOW_LIMIT（オプション）-ハードウェアに出力する値を計算するときに、値+オフセットが LOW_LIMIT より小さい場合、代わりに LOW_LIMIT が使用されます。
- （FLOAT）BIT_WEIGHT（オプション）-ボルト（または電流出力の場合は mA）単位の最下位ビット（LSB）の値。
- （FLOAT）HW_OFFSET（オプション）-ハードウェアに 0 が書き込まれた場合に出力される実際の電圧（または電流）。

12.7.17 関数

（FUNCT）WRITE —これにより、計算された値がハードウェアに出力されます。 ENABLE が FALSE の場合、値、スケール、オフセットに関係なく、出力は 0 になります。 「0」の意味はハードウェアによって異なります。 たとえば、バイポーラ 12 ビット A / D は、ハードウェアピンから 0 ボルトを取得する D / A に 0x1FF（ミッドスケール）を書き込む必要がある場合があります。 ENABLE が TRUE の場合、スケール、オフセット、および値を読み取り、ADC（スケール*値）+オフセットに出力します。 ENABLE が FALSE の場合、0 を出力します。

12.8 HAL ツール

12.8.1 Halcmd

HALCMD は、HAL を操作するためのコマンドラインツールです。 HALCMD のかなり完全なマニュアルページがあります。これは、ソースまたはパッケージのいずれかから LINUXCNC をインストールした場合にインストールされます。 マンページは使用情報を提供します：

```
man halcmd
```

LINUXCNC を「RUN-IN-PLACE」用にコンパイルした場合は、マニュアルページを使用できるようにするために RIP-ENVIRONMENT スクリプトを入手する必要があります。

```
cd toplevel_directory_for_rip_build
. scripts/rip-environment
man halcmd
```

HAL チュートリアルには、HALCMD の使用例がいくつかあり、HALCMD の優れたチュートリアルです。

12.8.2 Halmeter

HALMETER は HAL 用の電圧計です。 ピン、信号、またはパラメーターを確認し、そのアイテムの現在の値を表示できます。 使い方はとても簡単です。 XWINDOWS シェルで HALMETER と入力して開始します。 HALMETER は GUI アプリケーションです。 [選択]と[終了]というラベルの付いた 2 つのボタンが付いた小さなウィンドウがポップアップ表示されます。 終了は簡単です-それはプログラムをシャットダウンします。 [選択]を選択すると、3 つのタブがある大きなウィンドウがポップアップ表示されます。 1 つのタブには、HAL で現在定義されているすべてのピンが一覧表示されます。 次はすべての信号をリストし、最後のタブはすべてのパラメーターをリストします。 タブをクリックしてから、ピン/信号/パラメータをクリックします。 次に、[OK]をクリックします。 リストが消え、小さなウィンドウに選択したアイテムの名前と値が表示されます。 表示は 1 秒間に約 10 回更新されます。 [OK]ではなく[承認]をクリックすると、小さなウィンドウに選択したアイテムの名前と値が表示されますが、大きなウィンドウは画面に表示されたままになります。 これは、さまざまなアイテムをすばやく確認したい場合に便利です。

複数のアイテムを監視する場合は、同時に多数のハルメーターを実行できます。 シェルウィンドウを拘束せずに HALMETER を起動する場合は、HALMETER&と入力してバックグラウンドで実行します。 コマンドラインに PIN | SIG | PAR [AM] <NAME>を追加することで、HALMETER に特定のアイテムの表示をすぐに開始させることもできます。 起動するとすぐに、ピン、信号、またはパラメータ<NAME>が表示されます。 （そのような項目がない場合は、通常どおり起動します。）最後に、表示する項目を指定する場合は、PIN | SIG | PARAM の前に-S を追加して、HALMETER に小さなウィンドウを使用するように指示できます。 アイテム名は値の下ではなくタイトルバーに表示さ

れ、ボタンはありません。 小さな画面スペースに多くのメーターが必要な場合に便利です。 詳細については、HALMETER チュートリアルのセクションを参照してください。

HALMETER は、ターミナルまたは AXIS からロードできます。 HALMETER は、値の表示において HALSHOW よりも高速です。 HALMETER には 2 つのウィンドウがあります。1 つは監視するピン、信号、またはパラメーターを選択するためのもので、もう 1 つは値を表示するためのものです。 複数のハルメーターを同時に開くことができます。 スクリプトを使用して複数の HALMETER を開く場合は、画面の左上隅を基準にして、それぞれの位置を-G XY で設定できます。 例えば：

```
loadusr halmeter pin hm2.0.stepgen.00.velocity-fb -g 0 500
```

その他のオプションについては、MAN ページを参照してください。 セクション HALMETER を参照してください。

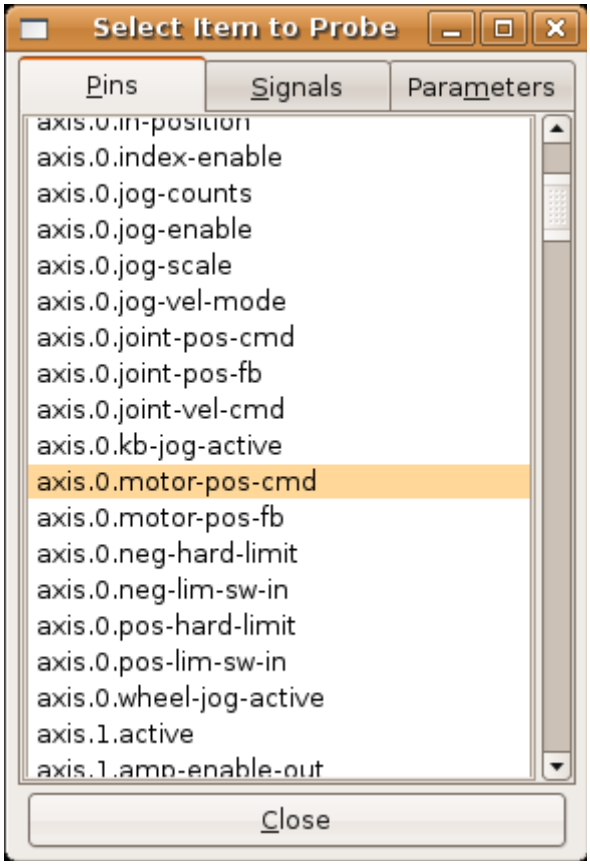
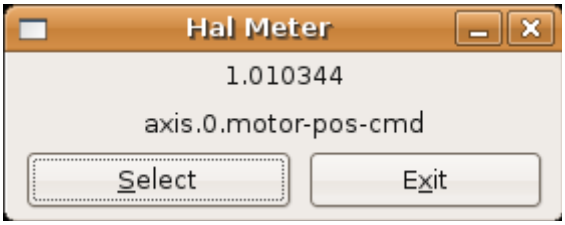


図 14-114



12.8.3 Halshow

HALSHOW（完全な使用法の説明）をコマンドラインから開始して、実行中の HAL の選択したコンポーネント、ピン、パラメーター、信号、機能、およびスレッドの詳細を表示できます。 [ウォッチ]タブでは、選択したピン、パラメーター、および信号アイテムが継続的に表示されます。 [ファイル]メニューには、1）ウォッチアイテムをウォッチリストに保存し、既存のウォッチリストをロードするためのボタンがあります。 ウォッチリストアイテムは、起動時に自動的にロードすることもできます。 コマンドラインで使用する場合：

halshow --help

Usage:

halshow [Options] [watchfile]

Options:

--help (this help)

--fformat format_string_for_float

--iformat format_string_for_int

NOTE

HALSHOW でウォッチファイルを作成するには、「ファイル/ウォッチリストの保存」を使用します。スタンドアロンで使用するには、LINUXCNC が実行されている必要があります。

LINUXCNC は、スタンドアロンで使用するために実行されている必要があります

[ファイル] / [ウォッチリストの保存]メニュー項目を使用して作成されたウォッチファイルは、トークン「PIN +」、「PARAM +」、「SIG = +」の後に適切なピン、パラメータ、または信号名が続く 1 行としてフォーマットされます。 トークンと名前のペアはスペース文字で区切られます。
例：PIN + JOIN.0.POS-HARD-LIMIT PIN + JOIN.1.POS-HARD-LIMIT SIG + ESTOP-LOOP

[ファイル/ウォッチリストの保存（複数行）]メニュー項目を使用して作成されたウォッチファイルは、上記のようにトークンと名前のペアで識別される項目ごとに個別の行でフォーマットされます。

例：PIN + JOIN.0.POS-HARD-LIMIT PIN + JOIN.1.POS-HARD-LIMIT SIG + ESTOP-LOOP

[ファイル] / [ウォッチリストのロード]メニュー項目を使用してウォッチファイルをロードすると、トークンと名前のペアが 1 行または複数行として表示される場合があります。 空白行と＃文字で始まる行は無視されます。

12.8.4 Halscope

HALSCOPE は、HAL 用のオシロスコープです。 これにより、ピン、信号、およびパラメーターの値を時間の関数としてキャプチャできます。 完全な操作手順は、最終的にはここにあります。 今のところ、基本を説明しているチュートリアルの章のセクション[SEC：TUTORIAL-HALSCOPE]を参照してください。

HALSCOPE の[ファイル]メニューセレクターには、構成を保存したり、以前に保存した構成を開いたりするためのボタンがあります。 HALSCOPE が終了すると、最後の構成が AUTOSAVE.HALSCOPE という名前のファイルに保存されます。

コマンドラインから HALSCOPE を起動するときに、構成ファイルを指定することもできます。 コマンドラインヘルプ（-H）の使用法：

```
halscope -h

Usage:

halscope [-h] [-i infile] [-o outfile] [num_samples]
```

12.8.5 Sim Pin

SIM_PIN は、書き込み可能なピン、パラメータ、または信号をいくつでも表示および更新するためのコマンドラインユーティリティです。 使用法：

```
Usage:
    sim_pin [Options] name1 [name2 ...] &
Options:
      --help            (this text)
      --title title_string     (window title, default: sim_pin)
Note: LinuxCNC (or a standalone Hal application) must be running
      A named item can specify a pin, param, or signal
      The item must be writable, e.g.:
      pin: IN or I/O (and not connected to a signal with a writer)
      param: RW
    signal: connected to a writable pin
    Hal item types bit,s32,u32,float are supported
    When a bit item is specifed, a pushbutton is created
    to manage the item in one of three manners specified
    by radio buttons:
      toggle: Toggle value when button pressed
      pulse: Pulse item to 1 once when button pressed
      hold: Set to 1 while button pressed
    The bit pushbutton mode can be specifed on the command
    line by formatting the item name:
        namei/mode=[toggle | pulse | hold]
    If the mode begins with an uppercase letter, the radio
```

```
    buttons for selecting other modes are not shown
```

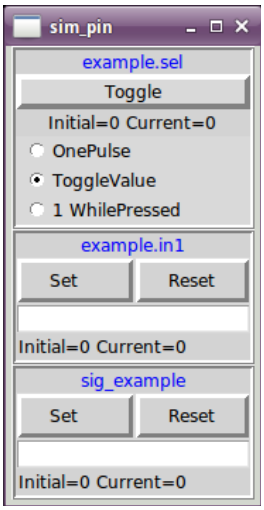
詳細については、MAN ページを参照してください。

```
man sim_pin
```

例（LINUXCNC を実行している場合）：

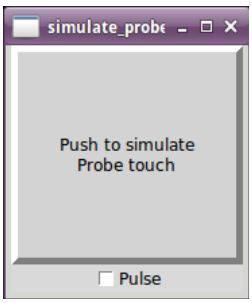
```
halcmd loadrt mux2 names=example; halcmd net sig_example example.in0

sim_pin example.sel example.in1 sig_example &
```



12.8.6 プローブのシミュレーション

SIMULATE_PROBE は、ピン MOTION.PROBE-INPUT のアクティブ化をシミュレートするための単純な GUI です。 使用法：



12.8.7 HAL ヒストグラム

HAL-HISTOGRAM は、HAL ピンのヒストグラムを表示するコマンドラインユーティリティです。

```
Usage:

hal-histogram --help | -?

or

hal-histogram [Options] [pinname]
```

Options:

--minvalue	minvalue	(minimum bin, default: 0)
--binsize	binsize	(binsize, default: 100)
--nbins	nbins	(number of bins, default: 50)
--logscale	0|1	(y axis log scale, default: 1)
--text	note	(text display, default: "")
--show		(show count of undisplayed nbins, default off)
--verbose		(progress and debug, default off)

Notes:

1) LinuxCNC (or another Hal application) must be running

2) If no pinname is specified, default is: motion-command-handler.time

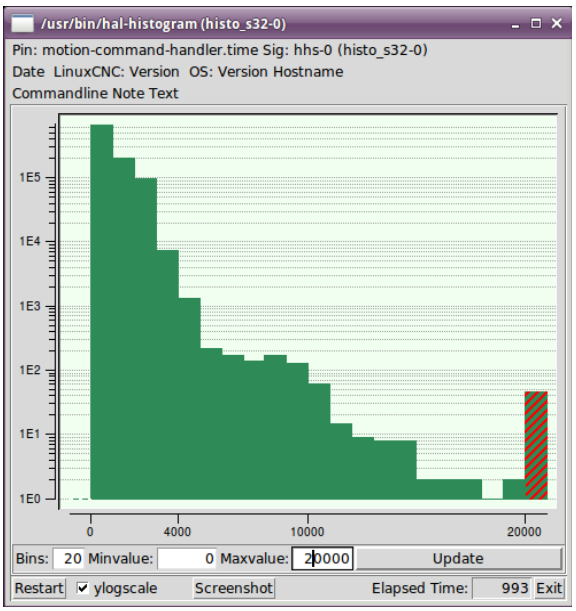
3) This app may be opened for 5 pins

4) pintypes float, s32, u32, bit are supported

5) The pin must be associated with a thread supporting floating point

For a base thread, this may require using:

loadrt motmod ... base_thread_fp=1



12.8.8 HAL レポート

HALREPORT は、実行中の LINUXCNC（または他の HAL）アプリケーションの HAL 接続に関するレポートを生成するコマンドラインユーティリティです。 レポートには、すべての信号接続が表示され、潜在的な問題にフラグが立てられます。 含まれる情報：

1. システムの説明とカーネルのバージョン。
2. 信号および接続されているすべての出力、IO、および入力ピン。
3. 各ピンの COMPONENT_FUNCTION、THREAD、および ADDF-ORDER。
4. 順序付けられていない機能を持つユーザースペースコンポーネントピン。
5. 未処理のコンポーネントの不明な機能の識別。
6. 出力のない信号。
7. 入力のない信号。
8. ADDF なしで機能します。
9. ドキュメントで非推奨/廃止としてマークされたコンポーネントの警告タグ。
10. エイリアス名を使用するピンの実名。

レポートはコマンドラインから生成され、出力ファイル（または、OUTFILENAME が指定されていない場合は STDOUT）に送信されます。

Usage:

halreport -h | --help (this help)

or

halreport [outfilename]

LINUXCNC の起動ごとにレポートを生成するには、HALREPORT と出力ファイル名を [APPLICATIONS] APP エントリとして INI ファイルに含めます。 例：

[APPLICATIONS]

APP = halreport /tmp/halreport.txt

関数の ADDF-ORDERING は、各サーボ周期で計算される関数のシーケンスが重要なサーボループで重要になる可能性があります。 通常、順序は次のとおりです。入力ピンの読み取り、モーションコマンドハンドラーおよびモーションコントローラー機能の実行、PID 計算の実行、最後に出力ピンの書き込み。

クリティカルパス内の各信号について、出力ピンの ADDF-ORDER は、接続先のクリティカル入力ピンの ADDF-ORDER よりも数値的に低くする必要があります。

スイッチ入力、ユーザースペースピンなどを処理するルーチン信号パスの場合、ADDF の順序は重要ではないことがよくあります。 さらに、ユーザースペースのピン値の変更のタイミングは、HAL スレッドで通常使用される間隔で制御または保証することはできません。

X 軸座標に対応する JOINT.0 を備えた TRIVKINS マシンで速度モードで動作する HOSTMOT2STEPGEN の PID ループを示すレポートファイルの抜粋の例：

```
SIG: pos-fb-0
OUT:    h.00.position-fb        hm2_7i92.0.read             servo-thread 001
(=hm2_7i92.0.stepgen.00.position-fb)
IN:     X_pid.feedback          X_pid.do-pid-calcs      servo-thread 004
IN:     joint.0.motor-pos-fb    motion-command-handler      servo-thread 002
.....                   motion-controller       servo-thread 003
...
SIG:    pos-cmd-0
OUT:    joint.0.motor-pos-cmd   motion-command-handler      servo-thread 002
.....                   motion-controller       servo-thread 003
IN:     X_pid.command               X_pid.do-pid-calcs      servo-thread 004
...
SIG: motor-cmd-0
OUT:    X_pid.output            X_pid.do-pid-calcs      servo-thread 004
IN:     h.00.velocity-cmd       hm2_7i92.0.write            servo-thread 008
(=hm2_7i92.0.stepgen.00.velocity-cmd)
```

上記の例では、HALFILE は HALCMD エイリアスを使用して、次のようなコマンドで HOSTMOT2FPGA ボードのピン名を簡略化します。

```
alias pin hm2_7i92.0.stepgen.00.position-fb h.00.position-fb
```

NOTE

疑わしいコンポーネント関数の検出は、1）サポートされていない（非推奨の）コンポーネント、2）複数の関数または従来とは異なる関数の命名を使用するユーザー作成コンポーネント、または 3）GUI に基づくプレフィックスなどの識別特性を欠く GUI 作成ユーザースペースコンポーネントで発生する可能性があります プログラム名。 疑わしい関数には疑問符「？」が付いています。

NOTE

既知のスレッド機能に関連付けることができないコンポーネントピンは、その機能を「不明」として報告します。

NOTE

HALREPORT は、接続の設計と検証を支援するために、実行中の HAL アプリケーションの接続レポートを生成します。 ピンタイプと現在の値は表示されていません。 この情報については、HALSHOW、HALMETER、HALSCOPE、または COMAND-LINEHALCMD プログラムで使用可能な SHOW コマンドなどのアプリケーションを使用してください。

12.9 HAL 表示

スクリプト HALSHOW は、実行中の HAL を回避する方法を見つけるのに役立ちます。 これは非常に特殊なシステムであり、動作中の HAL に接続する必要があります。 HALCMD インターフェイスライブラリを介して HAL が認識していることを報告する機能に依存しているため、スタンドアロンで実行することはできません。 それは発見に基づいています。 HALSHOW が異なる LINUXCNC 構成で実行されるたびに、それは異なります。

すぐにわかるように、HAL がそれ自体を文書化するこの機能は、効果的な CNC システムを作成するための 1 つの鍵です。

12.9.1 HAL 表示の開始

HALSHOW は、AXIS メニューの MACHINE / SHOW HALCONFIGURATION の下にあります。

HALSHOW は、SCRIPTS / HALSHOW の下の TKLINUXCNC メニューにあります。

HALSHOW は、ターミナルコマンドラインから起動して、整数および浮動小数点項目（ピンまたは信号）の形式を指定し、使用する保存済みウォッチリストファイルを識別できます。

```
$ halshow --help
Usage:
halshow [Options] [watchfile]
Options:
--help (this help)
--fformat format_string_for_float
--iformat format_string_for_int
Notes:
Create watchfile in halshow using: ’File/Save Watch List’
linuxcnc must be running for standalone usage
```

FLOAT の小数点数を制限し、現在のディレクトリで MY.HALSHOW という名前のファイルを使用する例：

```
$ halshow --fformat ".5f" ./my.halsh
```

12.9.2 HAL ツリー領域

図に示すように、ディスプレイの左側にはツリービューがあります。これは、一部のファイルブラウザで見られるようなものです。 右側は、ショーとウォッチ用のタブが付いたタブ付きノートブックです。

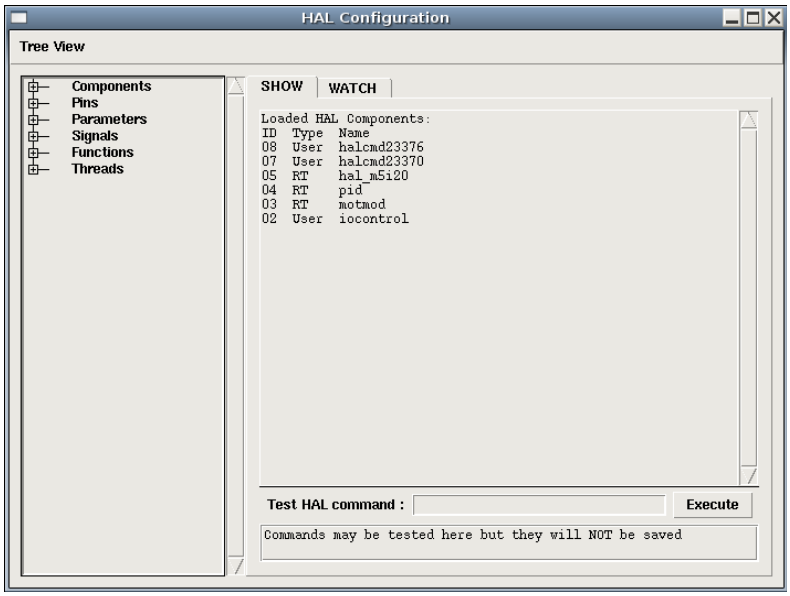


図 14-115

ツリーには、HAL の主要部分がすべて表示されます。 それぞれの前には、ボックス内の小さなプラス（+）またはマイナス（-）記号があります。 プラス記号をクリックすると、そのツリーノードが展開され、その下にあるものが表示されます。 そのボックスにマイナス記号が表示されている場合は、それをクリックするとツリーのそのセクションが折りたたまれます。

ディスプレイの左上端にある[ツリービュー]メニューを使用して、ツリーディスプレイを展開または折りたたむこともできます。 ツリービューの下に次のものがあります。ツリーを展開、ツリーを折りたたむ。 ピンの展開、パラメーターの展開、信号の展開。 および ERASEWATCH。（時計を消去すると、以前に設定したすべての時計が消去されます。1 つの時計だけを消去することはできません。）

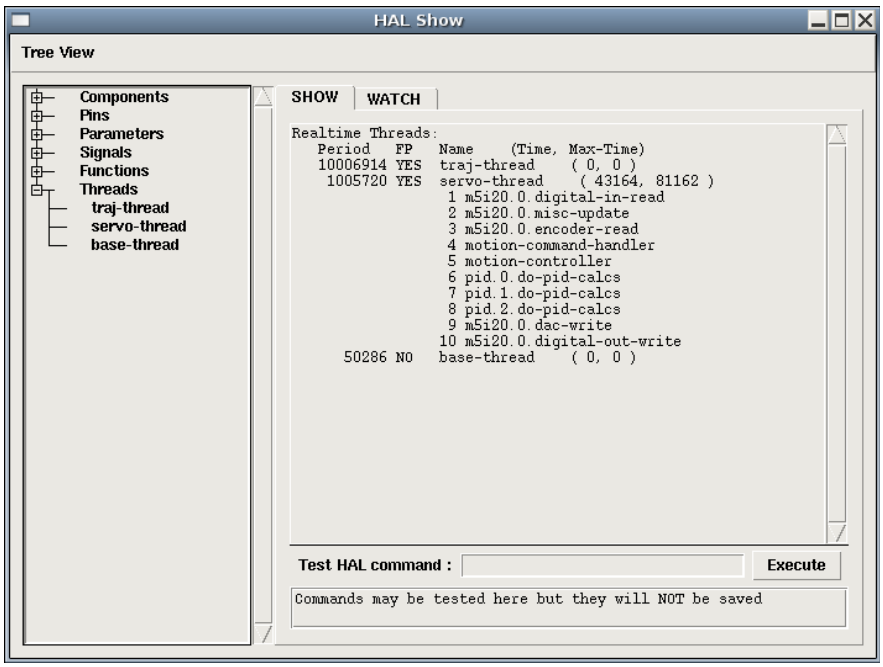


図 14-116

12.9.3 HAL 表示領域

ノード名、たとえば「コンポーネント」をクリックすると、HAL がそのノードのコンテンツについて知っているすべての情報が（[表示]タブの下に）表示されます。 図図 14.18 は、標準の M5I20 サーボカードを実行しているときに「コンポーネント」名をクリックした場合に表示されるのとまったく同じリストを示しています。 情報の表示は、従来のテキストベースの HAL 分析ツールに表示されるものとまったく同じです。 ここでの利点は、マウスクリックアクセスが可能であるということです。アクセスは、必要に応じて幅広く、または焦点を絞ることができます。

ツリー表示を詳しく見ると、HAL の 6 つの主要部分がすべて少なくとも 1 つのレベルに拡張できることがわかります。 これらのレベルが展開されると、右端のツリーノードをクリックしたときの応答にさらに集中できます。 複数の応答を示す HAL ピンとパラメーターがいくつかあることがわかります。 これは、HALCMD 自体の検索ルーチンの性質によるものです。 1 つのピンを検索すると、次のように 2 つのピンが表示される場合があります。

```
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
```

2 番目のピンの名前には、最初のピンの完全な名前が含まれています。

右側のショーエリアの下には、実行中の HAL で遊ぶことができるウィジェットのセットがあります。 ここで入力したコマンドと、それらが実行中の HAL に与える影響は保存されません。 これらは、LINUXCNC が稼働している限り存続しますが、LINUXCNC が稼働するとすぐになくなります。

「TESTHALCOMMAND：」というラベルの付いた入力ボックスは、HALCMD にリストされているすべてのコマンドを受け入れます。 これらには以下が含まれます：

- LOADRT, UNLOADRT (LOAD/UNLOAD REAL-TIME MODULE)
- LOADUSR, UNLOADUSR (ユーザースペースコンポーネントのロード/アンロード)
- ADDF、DELF（リアルタイムスレッドへの関数の追加/リアルタイムスレッドからの関数の削除）
- NET (2 つ以上のアイテム間の接続を作成します)
- SETP（パラメーター（またはピン）を値に設定）

この小さなエディタは、<ENTER>を押すか、実行ボタンを押すたびにコマンドを入力します。 これらのコマンドが適切に形成されていない場合、HALCMD からのエラーメッセージがこのエントリウィジェットの下に表示されます。 適切なコマンドを設定する方法がわからない場合は、HALCMD に関するドキュメントと使用している特定のモジュールをもう一度読む必要があります。

このエディタを使用して、差動モジュールを HAL に追加し、それを軸の位置に接続して、位置の変化率、つまり加速度を確認してみましょう。 まず、DDT という名前の HAL コンポーネントをロードし、それをサーボスレッドに追加してから、ジョイントの位置ピンに接続する必要があります。 それが完了すると、HALSCOPE で微分器の出力を見つけることができます。 じゃあ、行きましょう。 （はい、これを調べました。）

```
loadrt ddt
```

コンポーネントノードを見ると、どこかに DDT が表示されているはずです。

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

案の定です。 ID が 08 であることに注意してください。次に、関数で使用できる関数を確認する必要があります。

```
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
```

ここでは、所有者＃08 を探し、DDT.0 という名前の関数を確認します。 サーボスレッドに DDT.0 を追加できるはずです。これにより、サーボスレッドが更新されるたびに計算が実行されます。 もう一度 ADDF コマンドを検索すると、次のような 3 つの引数が使用されていることがわかります。

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

FUNCTNAME = DDT.0 はすでにわかっているので、ツリーのスレッドノードを展開してスレッド名を正しく取得しましょう。 ここでは、サーボスレッドとベーススレッドの 2 つのスレッドが表示されます。 スレッド内の DDT.0 の位置は重要ではありません。 したがって、関数 DDT.0 をサーボスレッドに追加します。

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

これは表示用であるため、位置を空白のままにして、スレッドの最後の位置を取得します。 次の図は、このコマンドが発行された後の HALSHOW の状態を示しています。

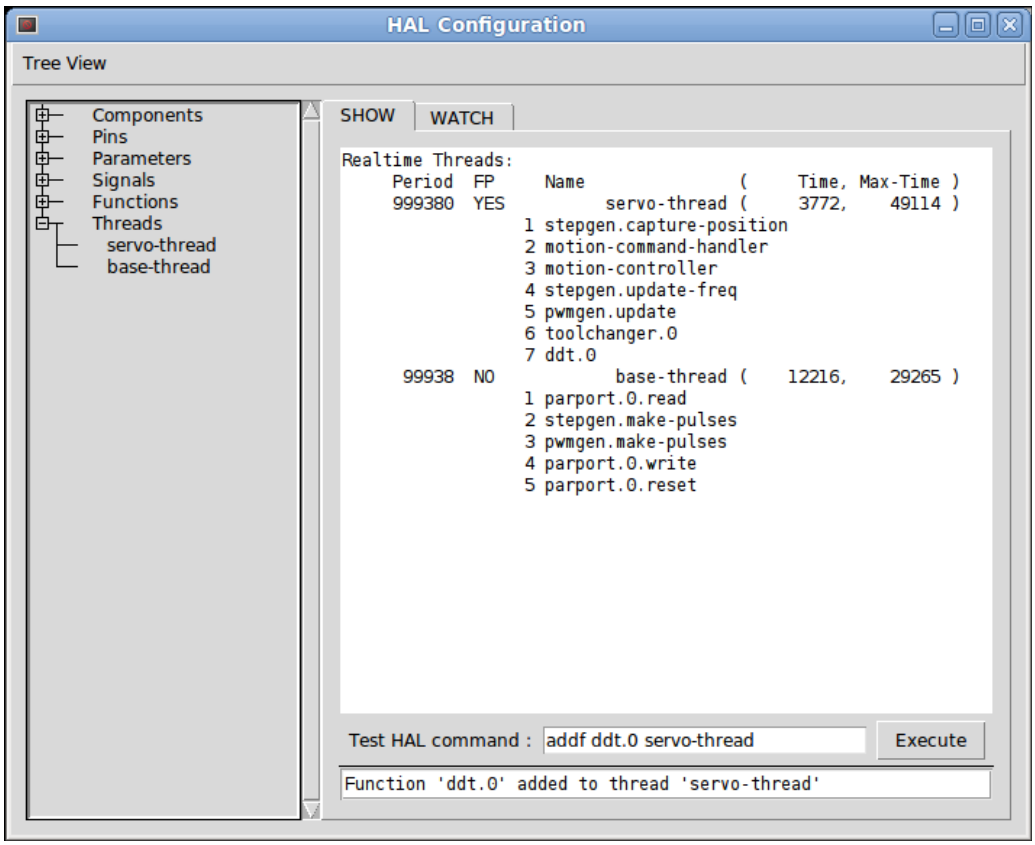


図 14-117

次に、DDT を何かに接続する必要があります。 しかし、どのピンが使用可能であるかをどうやって知るのでしょうか？ 答えはピンの下を見ることです。 そこで DDT を見つけて、これを確認します。

```
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
```

それは理解するのに十分簡単に見えますが、どの信号またはピンをそれに接続したいですか？ 軸ピン、STEPGEN ピン、または信号の場合があります。 これは、JOINT.0 を見るとわかります。

```
Component Pins:
Owner   Type  Dir   Value       Name
```

```
03              float     -W        0.00000e+00      joint.0.motor-pos-cmd ==> Xpos-cmd
```

したがって、XPOS-CMD は使用するのに適したシグナルであるように見えます。 次のコマンドを入力するエディターに戻ります。

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

ここで、ツリーノードを使用して XPOS-CMD シグナルを見ると、何をしたかがわかります。

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

この信号は JOINT.O.MOTOR-POS-CMD から送信され、DDT.0.IN と STEPGEN.0.POSITION-CMD の両方に送信されることがわかります。 ブロックを信号に接続することで、このモーションコマンドの通常のフローによる問題を回避しました。

HAL SHOW AREA は、HALCMD を使用して、実行中の HAL で何が起こっているかを検出します。それはあなたにそれが発見したものについての完全な情報を与えます。 また、小さなエディターパネルからコマンドを発行して、その HAL を変更すると更新されます。 この領域で利用可能なすべての情報なしで、別のセットを表示したい場合があります。 そこで、HAL ウォッチエリアが重要になります。

12.9.4 ウオッチタブ

ウォッチタブをクリックすると、空白のキャンバスが作成されます。 このキャンバスにシグナルとピンを追加して、それらの値を監視できます。5 ウォッチタブが表示されているときに、その名前をクリックしてシグナルまたはピンを追加できます。 次の図は、いくつかの「ビット」タイプの信号を含むこのキャンバスを示しています。 これらの信号には、最初の 3 つの軸のイネーブルアウトと 3 つの IOCONTROL「ESTOP」信号のうちの 2 つが含まれます。 LINUXCNC が ESTOP にないことを ESTOP 信号が示している場合でも、軸が有効になっていないことに注意してください。 TKLINUXCNC をざっと見ると、LINUXCNC の状態が ESTOPRESET であることがわかります。 アンプの有効化は、マシンの電源がオンになるまで TRUE になりません。

5-時計のディスプレイのリフレッシュレートは、HALMETER や HALSCOPE よりもはるかに低くなっています。信号のタイミングを適切に解決する必要がある場合は、これらのツールの方がはるかに効果的です。

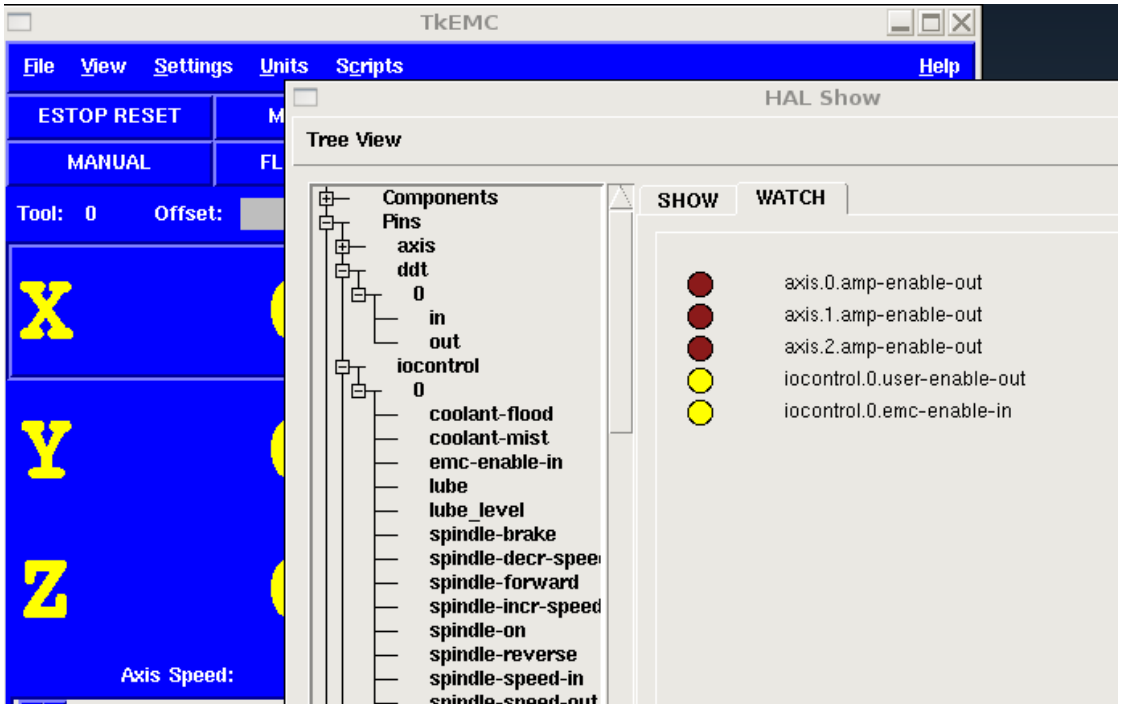


図 14-118

WATCH は、LED を表す色付きの円を使用してビットタイプ（バイナリ）値を表示します。 ビット信号またはピンが偽の場合は暗褐色で表示され、その信号が真の場合は淡黄色で表示されます。 ビットタイプ（バイナリ）信号ではないピンまたは信号を選択すると、ウォッチはそれを数値として表示します。

WATCH を使用すると、グラフィカルインターフェイスを使用しながら、スイッチをテストしたり、LINUXCNC に加えた変更の影響をすばやく確認したりできます。 時計のリフレッシュレートはステッパーパルスを表示するのに少し遅いですが、軸を非常にゆっくりと移動したり、距離を少しずつ移動したりする場合は、これらに使用できます。 LINUXCNC で IO_SHOW を使用したことがある場合は、HALSHOW の監視ページを設定して、IO_SHOW と同じようにパーポートを監視できます。

12.10 HAL コンポーネント

12.10.1 コマンドとユーザースペースコンポーネント

次のリストのすべてのコマンドには MAN ページがあります。 説明が拡張されているものもあれば、限定された説明があるものもあります。 また、以下にリストされているすべてのコンポーネントにはマニュアルページがあります。 このリストから、どのコンポーネントが存在するかがわかり、MANN 名を使用して追加情報を取得できます。 マニュアルページの情報を表示するには、ターミナルウィンドウで次のように入力します。

```
man axis (or perhaps 'man 1 axis' if your system requires it.)
```

AXIS

AXIS LINUXCNC（拡張マシンコントローラー）グラフィカルユーザーインターフェイス。

AXIS-REMOTE

AXIS リモートインターフェース。

COMP

LINUXCNC HAL コンポーネントをビルド、コンパイル、およびインストールします。

GLADEVCP

GLADE、GTK、HAL ウィジェットに基づく LINUXCNC の仮想コントロールパネル。

GS2

AUTOMATION DIRECT GS2VFD 用の HAL ユーザースペースコンポーネント。

HALCMD

コマンドラインから ENHANCEDMACHINE CONTROLLERHAL を操作します。

HAL_INPUT

USBHID デバイスを含む任意の LINUX 入力デバイスで HAL ピンを制御します。

HALMETER

HAL ピン、信号、およびパラメーターを観察します。

HALRUN

コマンドラインから ENHANCEDMACHINE CONTROLLERHAL を操作します。

HALSAMPLER

HAL からのリアルタイムのサンプルデータ。

HALSTREAMER

ファイルデータをリアルタイムで HAL にストリーミングします。

HALUI

HAL ピンを観察し、NML を介して LINUXCNC にコマンドを送信します。

IO

ACCEPTS NML I/O COMMANDS, INTERACTS WITH HAL IN USERSPACE.

IOCONTROL

NML I / O コマンドを受け入れ、ユーザースペースで HAL と対話します。

LINUXCNC

LINUXCNC（拡張マシンコントローラー）。.

PYVCP

LINUXCNC の仮想コントロールパネル。.

SHUTTLE

CONTOURDESIGN 製の SHUTTLEXPRESS および SHUTTLEPRO デバイスを使用して HAL ピンを制御します。

12.10.2 リアルタイムコンポーネントリスト

これらのいくつかは、MAN ページからの説明が拡張されています。 説明が限られているものもあります。 すべてのコンポーネントにはマニュアルページがあります。 このリストから、どのコ

ンポーネントが存在するかがわかり、MANN 名を使用して追加情報を取得できます。 マニュアルページの情報を表示するには、ターミナルウィンドウで次のように入力します。

```
man motion (or perhaps 'man 9 motion' if your system requires it.)
```

ドキュメントのメインページの MAN ページセクションまたは MAN9 のディレクトリリストも参照してください。

12.10.3 コア LINUXCNC コンポーネント

MOTION

NML モーションコマンドを受け入れ、HAL とリアルタイムで対話します。

AXIS

NML モーションコマンドを受け入れ、HAL とリアルタイムで対話します。

CLASSICLADDER

ラダーロジックに基づくリアルタイムソフトウェア PLC。 詳細については、CLASSICLADDER の章を参照してください。

GLADEVCP

GTK / GLADE で構築された仮想コントロールパネルを表示します。

THREADS

ハードリアルタイム HAL スレッドを作成します。

12.10.4 ロジックおよびビット単位のコンポーネント

and2

2 入力 AND ゲート。 OUT が真であるためには、両方の入力が真でなければなりません。 詳細

not

インバーター。 詳細

or2

2 入力 OR ゲート。 詳細

xor2

2 入力 XOR（排他的論理和）ゲート。 詳細

dbounce

ノイズの多いデジタル入力をフィルタリングします。 詳細。

debounce

ノイズの多いデジタル入力をフィルタリングします。 詳細。 説明 縁

edge

エッジ検出器。

flipflop

D タイプのフリップフロップ。

oneshot

ワンショットパルスジェネレータ。

logic

一般的な論理関数コンポーネント。

lut5

ルックアップテーブルに基づく 5 入力ロジック関数。 説明

match8

8 ビットのバイナリ一致検出器。

select8

8 ビットのバイナリ一致検出器。

12.10.5 算術およびフロートコンポーネント

ABS

入力信号の絶対値と符号を計算します。

BLEND

2 つの値の間で線形補間を実行します。

COMP

ヒステリシス付きの 2 入力コンパレータ。

CONSTANT

パラメータを使用して、ピンの値を設定します。

SUM2

2 つの入力（それぞれゲイン付き）とオフセットの合計。

COUNTER

入力パルスをカウントします（非推奨）。エンコーダコンポーネントを使用します。

UPDOWN

オプションの制限とラップアラウンド動作を使用して、カウントアップまたはカウントダウンします。

DDT

入力関数の導関数を計算します。

DEADZONE

しきい値内の場合は中心を返します。

HYPOT

3 入力斜辺（ユークリッド距離）計算機。

MULT2

2 つの入力の積。

MUX16

16 個の入力値から 1 つを選択します。

MUX2

2 つの入力値のいずれかから選択します。

MUX4

4 つの入力値から 1 つを選択します。

MUX8

8 つの入力値から 1 つを選択します。

NEAR

2 つの値がほぼ等しいかどうかを判断します。

OFFSET

入力にオフセットを追加し、フィードバック値からそれを減算します。

INTEG

インテグレーター。

INVERT

入力信号の逆数を計算します。

WCOMP

ウィンドウコンパレータ。

WEIGHTED_SUM

ビットのグループを整数に変換します。

BIQUAD

BIQUADIIR フィルター

LOWPASS

ローパスフィルタ

LIMIT1

出力信号が最小値と最大値の間に収まるように制限します。 6

LIMIT2

出力信号が最小値と最大値の間に収まるように制限します。 スルーレートを 1 秒あたり最大 v 未満に制限します。 7

LIMIT3

出力信号が最小値と最大値の間に収まるように制限します。 スルーレートを 1 秒あたり最大 v 未満に制限します。 二次導関数をメートル秒/秒の 2 乗あたりの MAXA 未満に制限します。 8

MAJ3

3 つの入力の大部分を計算します。

SCALE

入力にスケールとオフセットを適用します。

6 入力が位置の場合、これは位置が制限されていることを意味します。

7 入力が位置の場合、これは位置と速度が制限されていることを意味します。

8 入力が位置の場合、位置、速度、加速度が制限されていることを意味します。

12.10.6 型変換

CONV_BIT_S32

値をビットから S32 に変換します。

CONV_BIT_U32

値をビットから U32 に変換します。

CONV_FLOAT_S32

値を FLOAT から S32 に変換します。

CONV_FLOAT_U32

値を FLOAT から U32 に変換します。

CONV_S32_BIT

値を S32 からビットに変換します。

CONV_S32_FLOAT

値を S32 から FLOAT に変換します。

CONV_S32_U32

値を S32 から U32 に変換します。

CONV_U32_BIT

値を U32 からビットに変換します。

CONV_U32_FLOAT

値を U32 から FLOAT に変換します。

CONV_U32_S32

値を U32 から S32 に変換します。

12.10.7 ハードウェアドライバー

HAL_PPMC

アナログサーボ、PWM、ステッパーコントローラー用の PICOSYSTEMS ドライバー。

HM2_7I43

HOSTMOT2 を搭載した 7I43EPP ANYTHINGIO ボード用の MESAELECTRONICS ドライバー。（詳細については、MAN ページを参照してください）

HM2_PCI

HOSTMOT2 ファームウェアを備えた 5I20、5I22、5I23、4I65、および 4I68 ANYTHING I / O ボード用の MESAELECTRONICS ドライバー。（詳細については、MAN ページを参照してください）

HOSTMOT2

HOSTMOT2 ファームウェア用の MESAELECTRONICS ドライバー。

MESA_7I65

7I658 軸サーボカード用の MESAELECTRONICS ドライバー。（詳細については、MAN ページを参照してください）

PLUTO_SERVO

サーボ用のパラレルポート FPGA 用の PLUTO-P ドライバーとファームウェア。

PLUTO_STEP

ステッパー用のパラレルポート FPGA 用の PLUTO-P ドライバー。

THC

MESATHC カードまたはアナログから速度への入力を使用したトーチ高さ制御

SERPORT

8250 および 16550 シリアルポートのデジタル I / O ビット用のハードウェアドライバー。

12.10.8 キネマティクス

KINS

LINUXCNC のキネマティクス定義。

GANTRYKINS

1 つの軸を複数の関節にマッピングするキネマティクスモジュール。

GENHEXKINS

位置と方向に 6 つの自由度を与えます（XYZABC）。 モーターの位置はコンパイル時に定義されます。

GENSERKINS

最大 6 つの角度ジョイントを備えた一般的なシリアルリンクマニピュレータをモデル化できるキネマティクス。

MAXKINS

傾斜ヘッド（B 軸）とテーブルに取り付けられた水平ロータリー（C 軸）を備えた MAX という名前の卓上 5 軸ミルの運動学。 回転した座標系で UVW モーションを提供します。

ソースファイル MAXKINS.C は、他の 5 軸システムの開始点として役立つ場合があります。

TRIPODKINS

ジョイントは、3 つの事前定義された位置（モーター）からの制御点の距離を表し、位置に 3 つの自由度（XYZ）を与えます。

TRIVKINS

ジョイントと軸の間には 1：1 の対応があります。 ほとんどの標準的なフライス盤と旋盤は、些細な運動学モジュールを使用しています。

PUMAKINS

PUMA スタイルのロボットの運動学。

ROTATEKINS

X 軸と Y 軸は、ジョイント 0 と 1 に比べて 45 度回転しています。

SCARAKINS

スカラ型ロボットの運動学。

12.10.9 モーター制御

AT_PID

自動調整付きの比例/積分/微分コントローラー。

PID

比例/積分/微分コントローラー。 説明

PWMGEN

ソフトウェア PWM / PDM 生成。 説明

ENCODER

直交エンコーダ信号のソフトウェアカウント。 説明。

STEPGEN

ソフトウェアステップパルス生成。 説明。

12.10.10 BLDC および 3 相モーター制御

BLDC_HALL3

ホールセンサーを使用した 3 線式バイポーラ台形転流 BLDC モータードライバー。

CLARKE2

クラーク変換の 2 つの入力バージョン。

CLARKE3

クラーク（3 相からデカルト座標）変換。

CLARKEINV

逆クラーク変換。

12.10.11 その他

CHARGE_PUMP

一部のコントローラボードのチャージポンプ入力用に方形波を作成します。 チャージポンプは、ベーススレッド機能に追加する必要があります。 有効にすると、出力は 1 期間オンになり、1 期間オフになります。 出力の頻度を計算するには 1 /（秒単位の周期時間 x 2）= HZ。 たとえば、0.0001 秒である 100,000NS の基本期間があり、式が 1 /（0.0001 x 2）= 5,000HZ または 5KHZ である場合。

ENCODER_RATIO

2 つの軸を同期させる電子ギア。

ESTOP_LATCH

ESTOP ラッチ。

FEEDCOMP

入力に現在の速度と送り速度の比率を掛けます。

GEARCHANGE

2 つの速度範囲のいずれかから選択します。

ILOWPASS

他のアプリケーションが見つかるかもしれませんが、このコンポーネントは、MPG でジョギングしているときにスムーズな動きを作成するために作成されました。加速度の高い機械では、短いジョグはほとんどステップ関数のように動作します。 MPG エンコーダカウント出力と軸ジョグカウント入力の間に ILOWPASS コンポーネントを配置することにより、これをスムーズにすることができます。スケールを慎重に選択して、1 回のセッション中に約 2E9 /スケールパルスを超えないようにします。 MPG。必要な平滑化レベルに応じてゲインを選択します。 AXIS.N.JOG-SCALE 値をスケールごとに表示します。

JOYHANDLE

非線形のジョイパッドの動き、不感帯、スケールを設定します。

KNOB2FLOAT

カウント（おそらくエンコーダーから）を浮動小数点値に変換します。

MINMAX

入力から出力への最小値と最大値を追跡します。

SAMPLE_HOLD

サンプルアンドホールド。

SAMPLER

HAL からのリアルタイムのサンプルデータ。

SIGGEN

信号発生器。説明。

SIM_ENCODER

シミュレートされた直交エンコーダ。説明。

SPHEREPROBE

ふりをする半球を調べます。

STEPTEST

STEPCONF が使用して、軸の加速度と速度の値をテストできるようにします。

STREAMER

ファイルデータをリアルタイムで HAL にストリーミングします。

SUPPLY

パラメータからの値で出力ピンを設定します（非推奨）。

THREADTEST

スレッドの動作をテストするためのコンポーネント。

TIME

累積ランタイムタイマーは、アクティブな入力の HH：MM：SS をカウントします。

TIMEDELAY

時間遅延リレーに相当します。

TIMEDELTA

スレッドスケジューリングのタイミング動作を測定するコンポーネント。

TOGGLE2NIST

ボタンを切り替えてロジックをネストします。

TOGGLE

瞬間的な押しボタンからのプッシュオン、プッシュオフ。

TRISTATE_BIT

電子機器のトライステートバッファと同様に、有効な場合にのみ I / O ピンに信号を配置します。

TRISTATE_FLOAT

電子機器のトライステートバッファと同様に、有効な場合にのみ I / O ピンに信号を配置します。

WATCHDOG

ハートビートの 1〜32 個の入力を監視します。

12.10.12 HAL API コール

hal_add_funct_to_thread.3hal

hal_bit_t.3hal

hal_create_thread.3hal

hal_del_funct_from_thread.3hal

hal_exit.3hal

hal_export_funct.3hal

hal_float_t.3hal

hal_get_lock.3hal

hal_init.3hal

hal_link.3hal

hal_malloc.3hal

hal_param_bit_new.3hal

hal_param_bit_newf.3hal

hal_param_float_new.3hal

hal_param_float_newf.3hal

hal_param_new.3hal

hal_param_s32_new.3hal

hal_param_s32_newf.3hal

hal_param_u32_new.3hal

hal_param_u32_newf.3hal

hal_parport.3hal

hal_pin_bit_new.3hal

hal_pin_bit_newf.3hal

hal_pin_float_new.3hal

hal_pin_float_newf.3hal

hal_pin_new.3hal

hal_pin_s32_new.3hal

hal_pin_s32_newf.3hal

hal_pin_u32_new.3hal

hal_pin_u32_newf.3hal

hal_ready.3hal

hal_s32_t.3hal

hal_set_constructor.3hal

hal_set_lock.3hal

hal_signal_delete.3hal

hal_signal_new.3hal

hal_start_threads.3hal

hal_type_t.3hal

hal_u32_t.3hal

hal_unlink.3hal

intro.3hal

undocumented.3hal

12.10.13 RTAPI 呼び出し

EXPORT_FUNCTION.3rtapi

MODULE_AUTHOR.3rtapi

MODULE_DESCRIPTION.3rtapi

MODULE_LICENSE.3rtapi

RTAPI_MP_ARRAY_INT.3rtapi

RTAPI_MP_ARRAY_LONG.3rtapi

RTAPI_MP_ARRAY_STRING.3rtapi

RTAPI_MP_INT.3rtapi

RTAPI_MP_LONG.3rtapi

RTAPI_MP_STRING.3rtapi

intro.3rtapi

rtapi_app_exit.3rtapi

rtapi_app_main.3rtapi

rtapi_clock_set_period.3rtapi

rtapi_delay.3rtapi

rtapi_delay_max.3rtapi

rtapi_exit.3rtapi

rtapi_get_clocks.3rtapi

rtapi_get_msg_level.3rtapi

rtapi_get_time.3rtapi

rtapi_inb.3rtapi

rtapi_init.3rtapi

rtapi_module_param.3rtapi

RTAPI_MP_ARRAY_INT.3rtapi

RTAPI_MP_ARRAY_LONG.3rtapi

RTAPI_MP_ARRAY_STRING.3rtapi

RTAPI_MP_INT.3rtapi

RTAPI_MP_LONG.3rtapi

RTAPI_MP_STRING.3rtapi

rtapi_mutex.3rtapi

rtapi_outb.3rtapi

rtapi_print.3rtap

rtapi_prio.3rtapi

rtapi_prio_highest.3rtapi

rtapi_prio_lowest.3rtapi

rtapi_prio_next_higher.3rtapi

rtapi_prio_next_lower.3rtapi

rtapi_region.3rtapi

rtapi_release_region.3rtapi

rtapi_request_region.3rtapi

rtapi_set_msg_level.3rtapi

rtapi_shmem.3rtapi

rtapi_shmem_delete.3rtapi

rtapi_shmem_getptr.3rtapi

rtapi_shmem_new.3rtapi

rtapi_snprintf.3rtapi

rtapi_task_delete.3rtpi

rtapi_task_new.3rtapi

rtapi_task_pause.3rtapi

rtapi_task_resume.3rtapi

rtapi_task_start.3rtapi

rtapi_task_wait.3rtapi

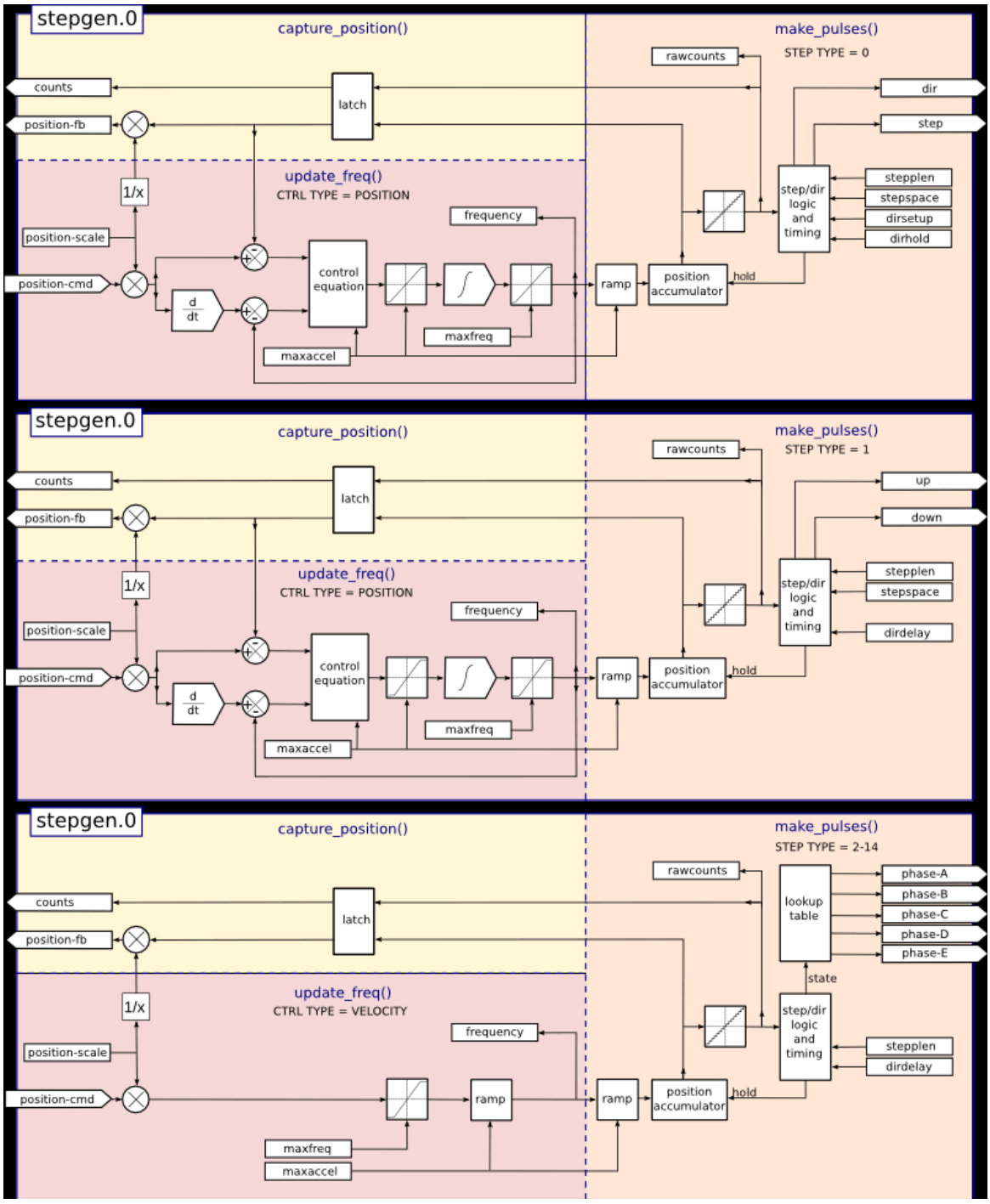
12.11 HAL コンポーネントの説明

12.11.1 Stepgen

このコンポーネントは、位置または速度コマンドに応答して、ソフトウェアベースのステップパルスの生成を提供します。 位置モードでは、事前に調整された位置ループが組み込まれているため、PID 調整は必要ありません。 速度モードでは、速度と加速度の制限に従いながら、指令された速度でモーターを駆動します。 これはリアルタイムコンポーネントのみであり、CPU 速度などに応じて、10kHz からおそらく 50kHz の最大ステップレートが可能です。 ステップパルスジェネレータのブロック図は 3 つのブロック図を示しており、それぞれがシングルステップパルスジェネレータです。 最初の図は、ステップタイプ 0（ステップと方向）用です。 2 つ目はステップタイプ 1（アップ/ダウン、または疑似 PWM）用で、3 つ目はステップタイプ 2〜14（さまざま

なステッピングパターン）用です。 最初の 2 つの図は位置モード制御を示し、3 番目の図は速度モードを示しています。 制御モードとステップタイプは個別に設定され、任意の組み合わせを選択できます。

。ステップパルスジェネレータブロック図位置モード



インストール

---

halcmd: loadrt stepgen step_type=<type-array> [ctrl_type=<ctrl_array>]

<TYPE-ARRAY>は、コンマで区切られた一連の 10 進整数です。 数値ごとにシングルステップパルスジェネレータがロードされ、数値の値によってステッピングタイプが決まります。

<CTRL_ARRAY>は、位置または速度モードを指定するための、コンマで区切られた一連の P 文字または V 文字です。 CTRL_TYPE はオプションです。省略した場合、すべてのステップジェネレーターは位置モードになります。

例えば：

```
halcmd: loadrt stepgen step_type=0,0,2 ctrl_type=p,p,v
```

3 ステップジェネレーターをインストールします。 最初の 2 つは、ステップタイプ 0（ステップと方向）を使用し、位置モードで実行します。 最後の 1 つは、ステップタイプ 2（直交）を使用し、速度モードで実行されます。 <CONFIG-ARRAY>のデフォルト値は 0,0,0 で、3 つのタイプ 0（ステップ/ディレクトリ）ジェネレーターがインストールされます。 ステップジェネレータの最大数は 8 です（STEPGEN.C の MAX_CHAN で定義されています）。 各ジェネレーターは独立していますが、すべてが同じ関数によって同時に更新されます。 以下の説明では、<CHAN>は特定のジェネレーターの番号です。 最初のジェネレーターは番号 0 です。

```
halcmd: unloadrt stepgen
```

ピン各ステップパルスジェネレータには、選択したステップタイプと制御タイプに応じて、これらのピンの一部のみがあります。

- （FLOAT）STEPGEN。<CHAN> .POSITION-CMD-位置単位での目的のモーター位置（位置モードのみ）。
- （FLOAT）STEPGEN。<CHAN> .VELOCITY-CMD-必要なモーター速度（位置単位/秒）（速度モードのみ）。
- （S32）STEPGEN。<CHAN> .COUNTS-COUNTS でのフィードバック位置。CAPTURE_POSITION（）によって更新されます。
- （FLOAT）STEPGEN。<CHAN> .POSITION-FB-位置単位でのフィードバック位置。CAPTURE_POSITION（）によって更新されます。
- （BIT）STEPGEN。<CHAN> .ENABLE-出力ステップを有効にします-FALSE の場合、ステップは生成されません。
- （BIT）STEPGEN。<CHAN> .STEP-ステップパルス出力（ステップタイプ 0 のみ）。
- （BIT）STEPGEN。<CHAN> .DIR-方向出力（ステップタイプ 0 のみ）。
- （BIT）STEPGEN。<CHAN> .UP-UP 疑似 PWM 出力（ステップタイプ 1 のみ）。
- （BIT）STEPGEN。<CHAN> .DOWN-DOWN 疑似 PWM 出力（ステップタイプ 1 のみ）。
- （BIT）STEPGEN。<CHAN> .PHASE-A-フェーズ A 出力（ステップタイプ 2〜14 のみ）。
- （BIT）STEPGEN。<CHAN> .PHASE-B-フェーズ B 出力（ステップタイプ 2〜14 のみ）。
- （BIT）STEPGEN。<CHAN> .PHASE-C-フェーズ C 出力（ステップタイプ 3〜14 のみ）。

- （BIT）STEPGEN。<CHAN> .PHASE-D-フェーズ D 出力（ステップタイプ 5〜14 のみ）。
- （BIT）STEPGEN。<CHAN> .PHASE-E-フェーズ E 出力（ステップタイプ 11〜14 のみ）。

パラメーター

- （FLOAT）STEPGEN。<CHAN> .POSITION-SCALE-位置単位あたりのステップ数。 このパラメーターは、出力とフィードバックの両方に使用されます。
- （FLOAT）STEPGEN。<CHAN> .MAXVEL-最大速度（位置単位/秒）。 0.0 の場合、効果はありません。
- （FLOAT）STEPGEN。<CHAN> .MAXACCEL-最大加速/減速速度（位置単位/秒の 2 乗）。 0.0 の場合、効果はありません。
- （FLOAT）STEPGEN。<CHAN> .FREQUENCY –現在のステップレート（1 秒あたりのステップ数）。
- （FLOAT）STEPGEN。<CHAN> .STEPLEN-ステップパルスの長さ（ステップタイプ 0 および 1）または特定の状態での最小時間（ステップタイプ 2〜14）、 ナノ秒単位で。
- （FLOAT）STEPGEN。<CHAN> .STEPSPACE-2 つのステップパルス間の最小間隔（ステップタイプ 0 および 1 のみ）（ナノ秒単位）。 STEPGEN DOUBLEFREQ 機能を有効にするには、0 に設定します。 DOUBLEFREQ を使用するには、パーポートリセット機能を有効にする必要があります。
- （FLOAT）STEPGEN。<CHAN> .DIRSETUP-方向変更から次のステップパルスの開始までの最小時間（ステップタイプ 0 のみ）（ナノ秒単位）。
- （FLOAT）STEPGEN。<CHAN> .DIRHOLD-ステップパルスの終了から方向変更までの最小時間（ステップタイプ 0 のみ）（ナノ秒単位）。
- （FLOAT）STEPGEN。<CHAN> .DIRDELAY-反対方向のステップまでの最小時間（ステップタイプ 1〜14 のみ）（ナノ秒単位）。
- （S32）STEPGEN。<CHAN> .RAWCOUNTS-MAKE_PULSES（）によって更新された生のフィードバックカウント。

位置モードでは、MAXVEL と MAXACCEL の値は、モーターが従うことができないステップパルス列の生成を回避するために内部位置ループによって使用されます。 モーターに適した値に設定すると、指令位置が瞬間的に大きく変化しても、新しい位置への台形の移動がスムーズになります。 このアルゴリズムは、位置誤差と速度誤差の両方を測定し、両方を同時にゼロにしようとする加速度を計算することによって機能します。 制御式ボックスの内容などの詳細については、コードを参照してください。

速度モードでは、MAXVEL はコマンドされた速度に適用される単純な制限であり、MAXACCEL は、コマンドされた速度が急激に変化した場合に実際の周波数を上昇させるために使用されます。 位置モードの場合と同様に、これらのパラメーターに適切な値を設定すると、モーターは生成されたパルス列に追従できます。

ステップタイプ 0 ステップタイプ 0 は、標準のステップおよび方向タイプです。 ステップタイプ 0 用に構成されている場合、ステップ信号と方向信号の正確なタイミングを決定する 4 つの追加パラメーターがあります。 次の図に、これらのパラメーターの意味を示します。 パラメータはナノ秒単位ですが、MAKE_PULSES（）を呼び出す脅威のスレッド期間の整数倍に切り上げられます。 たとえば、MAKE_PULSES（）が 16 US ごとに呼び出され、STEPLEN が 20000 の場合、ステップパルスは 2 x 16 = 32US の長さになります。 4 つのパラメーターすべてのデフォルト値は 1NS ですが、自動丸めはコードが最初に実行されたときに有効になります。 1 つのステップには高ステップレンと低ステップスペースが必要なため、最大周波数は 1,000,000,000 を（ステップレン+ステップスペース）で割ったものになります。 MAXFREQ がその制限より高く設定されている場合、自動的に下げられます。 MAXFREQ がゼロの場合、ゼロのままですが、出力周波数は制限されます。

パラレルポートドライバを使用する場合、ステップ周波数は、パーポートリセット機能と STEPGEN の DOUBLEFREQ 設定を使用して 2 倍にすることができます。

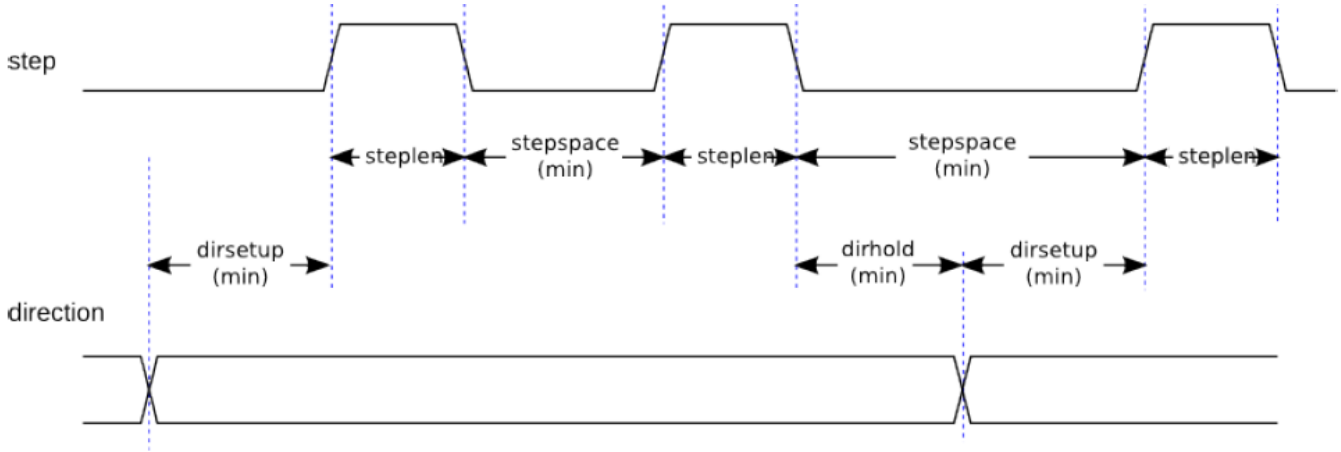


図 14-119

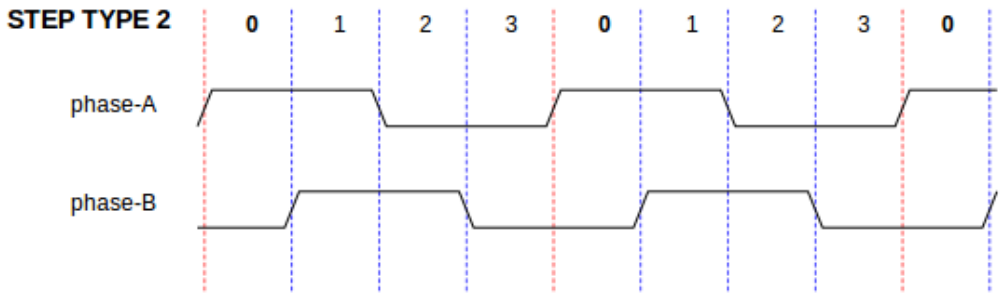
ステップタイプ 1 ステップタイプ 1 には、アップとダウンの 2 つの出力があります。 パルスは、進行方向に応じて、どちらかに現れます。 各パルスの長さはステップレンズであり、パルスは少なくともステップスペース NS だけ離れています。 最大周波数はステップタイプ 0 の場合と同じです。MAXFREQ が制限より高く設定されている場合、それは低くなります。 MAXFREQ がゼロの場合、ゼロのままですが、出力周波数は制限されます。

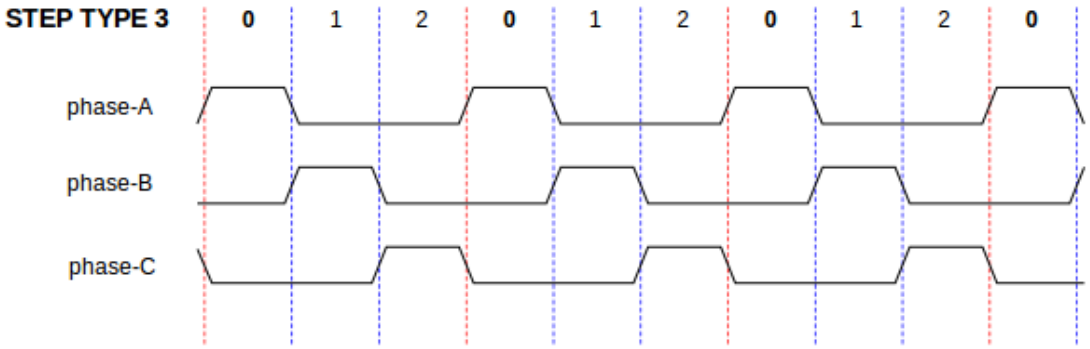
警告

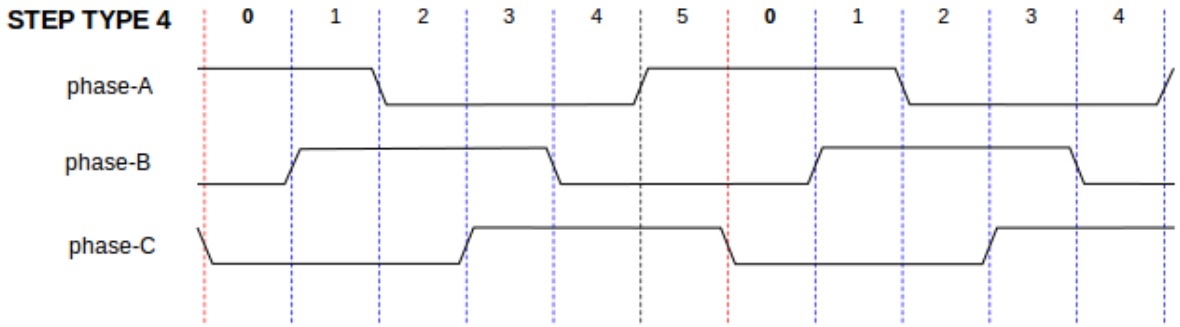
ステップタイプ 2～14 でパーポートリセット機能を使用しないでください。予期しない結果が発生する可能性があります。

ステップタイプ 2～14 ステップタイプ 2～14 は状態ベースであり、2～5 個の出力があります。 各ステップで、状態カウンターがインクリメントまたはデクリメントされます。 2 相および 3 相、4 相、および 5 相は、状態カウンターの関数としての出力パターンを示します。 最大周波数は 1,000,000,000 を STEPLEN で割ったものであり、他のモードと同様に、MAXFREQ は制限を超えると低下します。

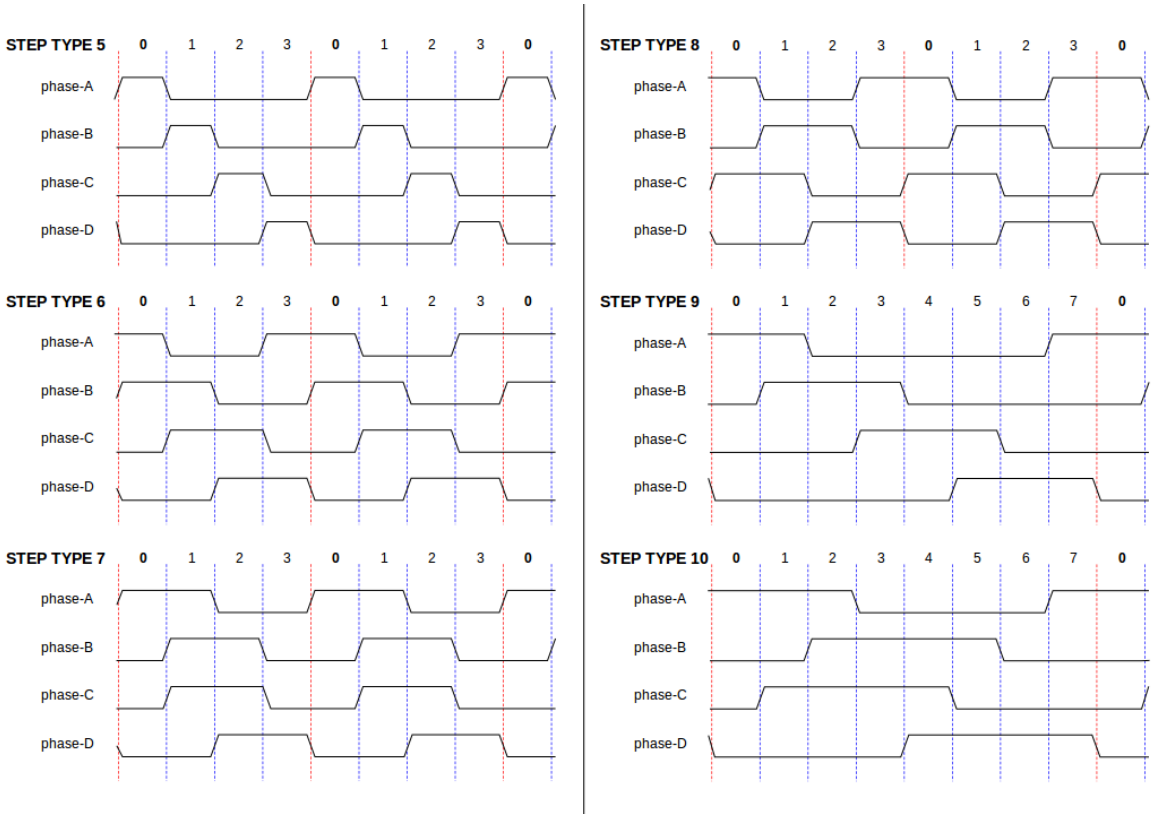
二相および三相ステップタイプ



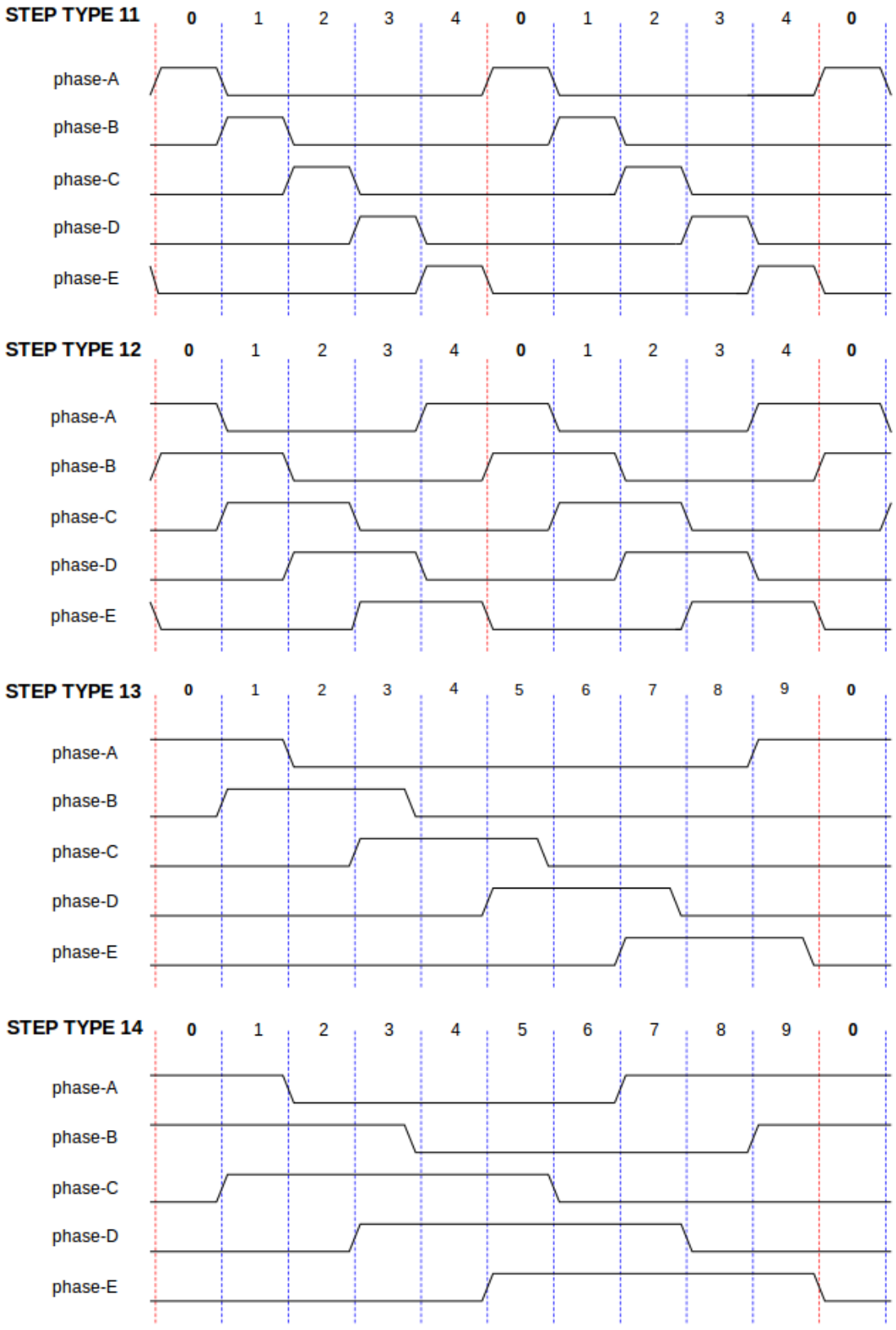




4 相ステップタイプ



5 段階のステップタイプ



関数コンポーネントは3つの関数をエクスポートします。 各関数は、すべてのステップパルスジェネレーターに作用します。異なるスレッドで異なるジェネレーターを実行することはサポートされていません。

- （機能）STEPGEN.MAKE-PULSES-パルスを生成およびカウントする高速関数（浮動小数点なし）。
- （機能）STEPGEN.UPDATE-FREQ –低速機能は、位置から速度への変換、スケーリング、および制限を行います。
- （機能）STEPGEN.CAPTURE-POSITION-フィードバック用の低速機能、ラッチの更新、および位置のスケーリング。

高速関数 STEPGEN.MAKE-PULSES は、コンピューターの機能に応じて 10〜50US の非常に高速なスレッドで実行する必要があります。 STEPLEN、STEPSPACE、DIRSETUP、DIRHOLD、および DIRDELAY はすべて、ナノ秒単位のスレッドの整数倍に切り上げられるため、そのスレッドの期間によって最大ステップ頻度が決まります。 他の 2 つの関数は、はるかに低いレートで呼び出すことができます。

12.11.2 PWMgen

このコンポーネントは、ソフトウェアベースの PWM（パルス幅変調）および PDM（パルス密度変調）波形の生成を提供します。 これはリアルタイムコンポーネントのみであり、CPU 速度などに応じて、かなり良好な解像度で数百ヘルツから、限られた解像度でおそらく 10KHz までの PWM 周波数が可能です。

インストール

```
loadrt pwmgen output_type=<config-array>
```

<CONFIG-ARRAY>は、コンマで区切られた一連の 10 進整数です。 数値ごとに 1 つの PWM ジェネレータがロードされ、数値の値によって出力タイプが決まります。 次の例では、3 つの PWM ジェネレーターをインストールします。 デフォルト値はありません。<CONFIG-ARRAY>が指定されていない場合、PWM ジェネレーターはインストールされません。 周波数ジェネレータの最大数は 8 です（PWMGEN.C の MAX_CHAN で定義されています）。 各ジェネレーターは独立していますが、すべてが同じ関数によって同時に更新されます。 以下の説明では、<CHAN>は特定のジェネレーターの番号です。 最初のジェネレーターは番号 0 です。

例

```
loadrt pwmgen output_type=0,1,2
```

削除

```
unloadrt pwmgen
```

出力タイプ PWM ジェネレータは 3 つの異なる出力タイプをサポートします。

- 出力タイプ 0-PWM 出力ピンのみ。 正のコマンドのみが受け入れられ、負の値はゼロとして扱われます（ゼロ以外の場合は、パラメーター MIN-DC の影響を受けます）。
- 出力タイプ 1-PWM / PDM および方向ピン。 正と負の入力は、正と負の PWM として出力されます。 方向ピンは、正のコマンドの場合は FALSE、負のコマンドの場合は TRUE です。 コントロールに CW と CCW の両方に正の PWM が必要な場合は、ABS コンポーネントを使用して、負の入力が入力されたときに PWM 信号を正の値に変換します。
- 出力タイプ 2-UP ピンと DOWN ピン。 正のコマンドの場合、PWM 信号はアップ出力に表示され、ダウン出力は FALSE のままです。 負のコマンドの場合、PWM 信号はダウン出力に表示され、アップ出力は FALSE のままです。 出力タイプ 2 は、ほとんどの H ブリッジの駆動に適しています。

ピン各 PWM ジェネレータには次のピンがあります。

- （FLOAT）PWMGEN。<CHAN> .VALUE-任意単位のコマンド値。 スケールパラメータによってスケーリングされます（以下を参照）。
- （BIT）PWMGEN。<CHAN> .ENABLE-PWM ジェネレータ出力を有効または無効にします。

各 PWM ジェネレーターには、選択した出力タイプに応じて、これらのピンのいくつかもあります。

- （BIT）PWMGEN。<CHAN> .PWM-PWM（または PDM）出力（出力タイプ 0 および 1 のみ）。
- （BIT）PWMGEN。<CHAN> .DIR-方向出力（出力タイプ 1 のみ）。
- （BIT）PWMGEN。<CHAN> .UP-正の入力値の PWM / PDM 出力（出力タイプ 2 のみ）。
- （BIT）PWMGEN。<CHAN> .DOWN-負の入力値の PWM / PDM 出力（出力タイプ 2 のみ）。

パラメーター

- （FLOAT）PWMGEN。<CHAN> .SCALE-値を任意の単位からデューティサイクルに変換するためのスケーリング係数。 たとえば、SCALE が 4000 に設定され、PWMGEN。<CHAN> .VALUE に渡される入力値が 4000 の場合、100％のデューティサイクル（常にオン）になります。 値が 2000 の場合、50％25Hz の方形波になります。
- （FLOAT）PWMGEN。<CHAN> .PWM-FREQ-必要な PWM 周波数（Hz）。 0.0 の場合、PWM の代わりに PDM を生成します。 内部制限よりも高く設定されている場合、UPDATE_FREQ（）を次に呼び出すと、内部制限に設定されます。 ゼロ以外で、ディザが FALSE の場合、UPDATE_FREQ（）を次に呼び出すと、MAKE_PULSES（）関数期間の最も近い整数倍に設定されます。
- （ビット）PWMGEN。<CHAN> .DITHER-PWM-TRUE の場合、ディザリングを有効にして、純粋な PWM では取得できない平均 PWM 周波数またはデューティサイクルを実現します。 FALSE の場合、PWM 周波数とデューティサイクルの両方が正確に達成できる値に丸められます。

- （FLOAT）PWMGEN。<CHAN> .MIN-DC-最小デューティサイクル、0.0〜1.0（この設定に関係なく、無効にするとデューティサイクルはゼロになります）。
- （FLOAT）PWMGEN。<CHAN> .MAX-DC-0.0〜1.0 の最大デューティサイクル。
- （FLOAT）PWMGEN。<CHAN> .CURR-DC-現在のデューティサイクル-すべての制限と丸めの後（読み取り専用）。

関数コンポーネントは 2 つの関数をエクスポートします。 各関数はすべての PWM ジェネレーターに作用します-異なるスレッドで異なるジェネレーターを実行することはサポートされていません。

- （FUNCT）PWMGEN.MAKE-PULSES-PWM 波形を生成する高速関数（浮動小数点なし）。 高速関数 PWMGEN.MAKE-PULSES は、コンピューターの機能に応じて 10〜50 US のベース（最速）スレッドで実行する必要があります。 そのスレッドの周期によって、最大 PWM キャリア周波数と、PWM または PDM 信号の分解能が決まります。 ベーススレッドが 50,000NS の場合、モジュールは 50US ごとに出力の状態を変更する時期かどうかを判断します。 50％のデューティサイクルと 25Hz の PWM 周波数では、これは出力が（1/25）秒ごとに状態を変更することを意味します/ 50US * 50％= 400 回の反復。 これは、800 の可能なデューティサイクル値があることも意味します（ディザリングなし）
- （FUNCT）PWMGEN.UPDATE-値をスケーリングおよび制限し、他のパラメーターを処理する低速関数。 これは、より複雑な数学を実行して、出力を高くする必要がある基本周期の数と、出力を低くする必要がある数を計算するモジュールの機能です。

12.11.3 Encoder

このコンポーネントは、直交エンコーダからの信号のソフトウェアベースのカウントを提供します。 これはリアルタイムコンポーネントのみであり、CPU 速度、遅延などに応じて、10KHz からおそらく最大 50KHz までの最大カウントレートが可能です。

ノイズとタイミングの変動を考慮して、ベーススレッドは 1/2 カウント速度である必要があります。 たとえば、スピンドルに 1 回転あたり 100 パルスのエンコーダがあり、最大 RPM が 3000 の場合、最大ベーススレッドは 25US になります。 1 回転あたり 100 パルスのエンコーダには 400 カウントがあります。 3000 RPM = 50 RPS（1 秒あたりの回転数）のスピンドル速度。 400 * 50 = 1 秒あたり 20,000 カウントまたはカウント間で 50US。

エンコーダカウンタのブロック図は、エンコーダカウンタの 1 つのチャネルのブロック図です。

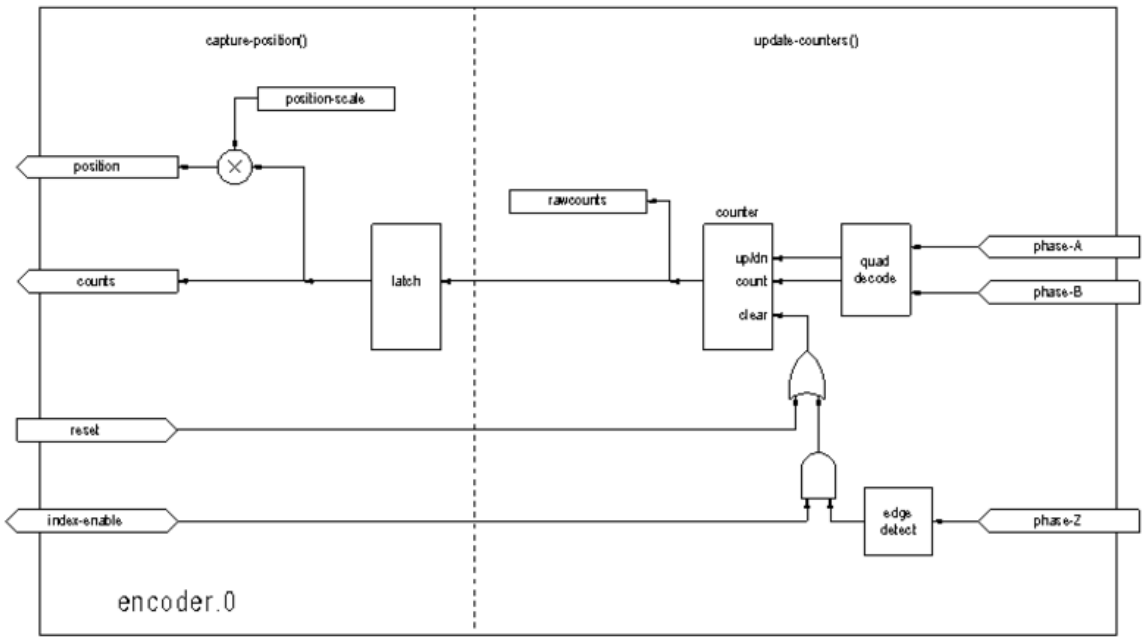


図 14-120

インストール

```
halcmd: loadrt encoder [num_chan=<counters>]
```

<COUNTERS>は、インストールするエンコーダーカウンターの数です。 NUMCHAN が指定されていない場合、3 つのカウンターがインストールされます。 カウンターの最大数は 8 です（ENCODER.C の MAX_CHAN で定義されています）。 各カウンターは独立していますが、すべてが同じ機能によって同時に更新されます。 以下の説明では、<CHAN>は特定のカウンターの番号です。 最初のカウンターは番号 0 です。

削除

```
halcmd: unloadrt encoder
```

ピン

- エンコーダー。<CHAN> .COUNTER-MODE（ビット、I / O）（デフォルト：FALSE）-カウンターモードを有効にします。 TRUE の場合、カウンタはフェーズ B の値を無視して、フェーズ A 入力の各立ち上がりエッジをカウントします。 これは、単一チャネル（非直交）センサーの出力をカウントするのに役立ちます。 FALSE の場合、直交モードでカウントされます。

- エンコーダー。<CHAN> .COUNTS（S32、OUT）-エンコーダーカウント内の位置。
- エンコーダー。<CHAN> .COUNTS-LATCHED（S32、OUT）-現時点では使用されていません。
- エンコーダ。<CHAN> .INDEX-ENABLE（ビット、I / O）-TRUE の場合、カウントと位置はフェーズ Z の次の立ち上がりエッジでゼロにリセットされます。同時に、インデックスイネーブルはゼロにリセットされ、次のことを示します。 立ち上がりエッジが発生しました。 インデックスイネーブルピンは双方向です。 INDEX-ENABLE が FALSE の場合、エンコーダーのフェーズ Z チャネルは無視され、カウンターは正常にカウントされます。 エンコーダドライバは、インデックスを有効にする TRUE を設定することはありません。 ただし、他のコンポーネントがそうする場合があります。
- エンコーダー。<CHAN> .LATCH-FALLING（ビット、入力）（デフォルト：TRUE）-現時点では使用されていません。
- エンコーダー。<CHAN> .LATCH-INPUT（ビット、入力）（デフォルト：TRUE）-現時点では使用されていません。
- エンコーダ。<CHAN> .LATCH-RISING（BIT、IN）-現時点では使用されていません。
- エンコーダー。<CHAN> .MIN-SPEED-ESTIMATE（FLOAT、IN）-速度が非ゼロとして推定され、位置補間が補間される最小の真の速度の大きさを決定します。 MIN-SPEED-ESTIMATE の単位は、速度の単位と同じです。 長さの単位あたりのカウントでのスケール係数。 このパラメータの設定が低すぎると、エンコーダパルスの到着が停止した後、速度が 0 になるまでに長い時間がかかります。
- エンコーダ。<CHAN> .PHASE-A（ビット、入力）-直交エンコーダ信号のフェーズ A。
- エンコーダ。<CHAN> .PHASE-B（ビット、入力）-直交エンコーダ信号のフェーズ B。
- エンコーダ。<CHAN> .PHASE-Z（ビット、入力）-直交エンコーダ信号のフェーズ Z（インデックスパルス）。
- エンコーダー。<CHAN> .POSITION（FLOAT、OUT）-スケーリングされた単位での位置（POSITION-SCALE を参照）。
- エンコーダー。<CHAN> .POSITION-INTERPOLATED（FLOAT、OUT）-エンコーダーカウント間で補間された、スケーリングされた単位での位置。 POSITIONINTERPOLATED は、最後に測定された速度に基づいて、エンコーダーカウント間を補間しようとします。 速度がほぼ一定で、最小速度の推定値を超えている場合にのみ有効です。 低速時、方向反転時、速度変更時は値が正しくないため、位置制御には使用しないでください。 ただし、低 PPR エンコーダ（1 回転あたり 1 パルスのエンコーダを含む）を旋盤のねじ切りに使用でき、他の用途もあります。
- エンコーダー。<CHAN> .POSITION-LATCHED（FLOAT、OUT）-現時点では使用されていません。

- エンコーダー。<CHAN> .POSITION-SCALE（FLOAT、I / O）-スケール係数（長さの単位あたりのカウント数）。 たとえば、位置スケールが 500 の場合、エンコーダの 1000 カウントは 2.0 単位の位置として報告されます。
- エンコーダー。<CHAN> .RAWCOUNTS（S32、IN）-UPDATE-COUNTERS によって決定された RAW カウント。 この値は、カウントや位置よりも頻繁に更新されます。 また、リセットやインデックスパルスの影響を受けません。
- エンコーダー。<CHAN> .RESET（BIT、IN）-TRUE の場合、カウントと位置を強制的にゼロにします。
- エンコーダー。<CHAN> .VELOCITY（FLOAT、OUT）-1 秒あたりのスケーリングされた単位での速度。 エンコーダは、位置出力を単純に微分する場合と比較して、量子化ノイズを大幅に低減するアルゴリズムを使用します。 真の速度の大きさが MIN-SPEED-ESTIMATE を下回る場合、速度出力は 0 です。
- エンコーダ。<CHAN> .X4-MODE（ビット、I / O）（デフォルト：TRUE）-4 回モードを有効にします。 TRUE の場合、カウンタは直交波形の各エッジをカウントします（フルサイクルごとに 4 カウント）。 FALSE の場合、フルサイクルごとに 1 回だけカウントされます。 カウンターモードでは、このパラメーターは無視されます。 1X モードは、一部のジョグホイールに役立ちます。

パラメーター

- エンコーダー。<CHAN> .CAPTURE-POSITION.TIME（S32、RO）
- エンコーダー。<CHAN> .CAPTURE-POSITION.TMAX（S32、RW）
- エンコーダー。<CHAN> .UPDATE-COUNTERS.TIME（S32、RO）
- エンコーダー。<CHAN> .UPDATE-COUNTER.TMAX（S32、RW）

関数コンポーネントは 2 つの関数をエクスポートします。 各関数は、すべてのエンコーダーカウンターに作用します。異なるスレッドで異なるカウンターを実行することはサポートされていません。

- （FUNCT）ENCODER.UPDATE-COUNTERS-パルスをカウントする高速関数（浮動小数点なし）。
- （FUNCT）ENCODER.CAPTURE-POSITION-ラッチとスケール位置を更新する低速機能。

12.11.4 PID

このコンポーネントは、比例/積分/微分制御ループを提供します。 これはリアルタイムコンポーネントのみです。 簡単にするために、この説明では位置ループについて説明していることを前提としていますが、このコンポーネントを使用して、速度、トーチの高さ、温度などの他のフィードバックループを実装できます。PID ループブロック図は、単一の PID ループのブロック図です。

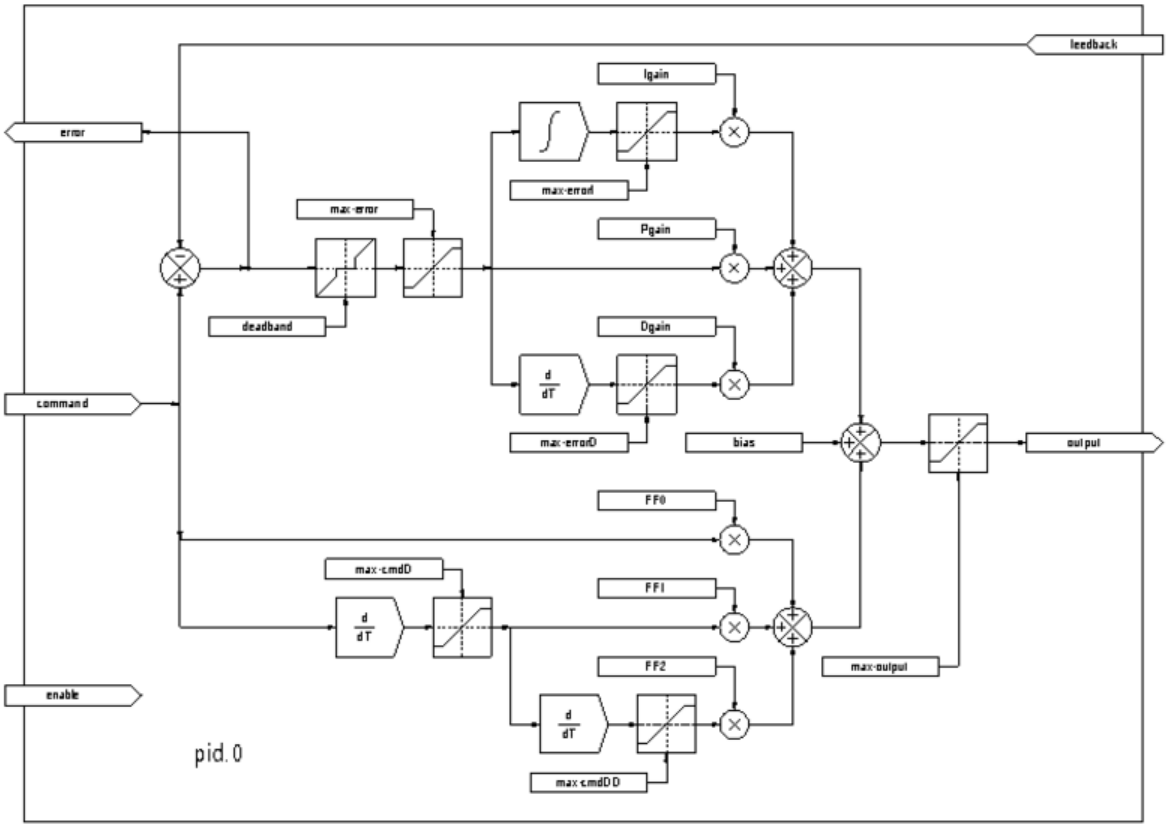


図 14-121

インストール

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

<LOOPS>は、インストールする PID ループの数です。 NUMCHAN が指定されていない場合、1 つのループがインストールされます。 ループの最大数は 16 です（PID.C の MAX_CHAN で定義されています）。 各ループは完全に独立しています。 以下の説明では、<LOOPNUM>は特定のループのループ番号です。 最初のループは番号 0 です。

DEBUG = 1 が指定されている場合、コンポーネントは、デバッグおよびチューニング中に役立つ可能性のあるいくつかの追加のピンをエクスポートします。 デフォルトでは、共有メモリスペースを節約し、ピンリストが乱雑にならないように、余分なピンはエクスポートされません。

削除

```
halcmd: unloadrt pid
```

ピン 3 つの最も重要なピンは

- FLOAT）PID。<LOOPNUM> .COMMAND-別のシステムコンポーネントによってコマンドされた目的の位置。 ••

- （FLOAT）PID。<LOOPNUM> .FEEDBACK-エンコーダーなどのフィードバックデバイスによって測定された現在の位置。
- （FLOAT）PID。<LOOPNUM> .OUTPUT-現在の位置から目的の位置に移動しようとする速度コマンド。

位置ループの場合、コマンドとフィードバックは位置単位です。 直線軸の場合、これはインチ、MM、メートル、または関連するものであれば何でもかまいません。 同様に、角度軸の場合、度、ラジアンなどになります。出力ピンの単位は、フィードバックをコマンドに一致させるために必要な変更を表します。 そのため、位置ループの場合、出力はインチ/秒、MM /秒、度/秒などの速度です。時間単位は常に秒であり、速度単位は位置単位と一致します。 コマンドとフィードバックがメートル単位の場合、出力はメートル/秒単位です。

各ループには、コンポーネントの一般的な動作を監視または制御するために使用される 2 つのピンがあります。

- （FLOAT）PID。<LOOPNUM> .ERROR-.COMMAND から.FEEDBACK を引いたものに等しい。
- （ビット）PID。<LOOPNUM> .ENABLE-ループを有効にするビット。 .ENABLE が FALSE の場合、すべての積分器がリセットされ、出力は強制的にゼロになります。 .ENABLE が TRUE の場合、ループは正常に動作します。

飽和を報告するために使用されるピン。 飽和は、PID ブロックの出力が最大または最小の制限にあるときに発生します。

- （ビット）PID。<LOOPNUM> .SATURATED-出力が飽和している場合は TRUE。
- FLOAT）PID。<LOOPNUM> .SATURATED_S-出力が飽和した時間。
- （S32）PID。<LOOPNUM> .SATURATED_COUNT-出力が飽和した時間。

ループの PID ゲイン、制限、およびその他の調整可能な機能はピンとして利用できるため、より高度な調整の可能性のために動的に調整できます。

- FLOAT）PID。<LOOPNUM> .PGAIN-比例ゲイン
- （FLOAT）PID。<LOOPNUM> .IGAIN-積分ゲイン
- （FLOAT）PID。<LOOPNUM> .DGAIN-微分ゲイン
- （FLOAT）PID。<LOOPNUM> .BIAS-出力の一定のオフセット
- （FLOAT）PID。<LOOPNUM> .FF0-ゼロ次フィードフォワード-コマンド（位置）に比例した出力。
- （FLOAT）PID。<LOOPNUM> .FF1-一次フィードフォワード-コマンドの導関数（速度）に比例する出力。

- （FLOAT）PID。<LOOPNUM> .FF2-2 次フィードフォワード-コマンドの 2 次導関数（加速）に比例する出力。
- （FLOAT）PID。<LOOPNUM> .DEADBAND-無視されるエラーの量
- （FLOAT）PID。<LOOPNUM> .MAXERROR-エラーの制限
- （FLOAT）PID。<LOOPNUM> .MAXERRORI-エラー積分器の制限
- （FLOAT）PID。<LOOPNUM> .MAXERRORD-誤差導関数の制限
- （FLOAT）PID。<LOOPNUM> .MAXCMDD-コマンド派生物の制限
- （FLOAT）PID。<LOOPNUM> .MAXCMDDD-コマンドの 2 次導関数の制限
- （FLOAT）PID。<LOOPNUM> .MAXOUTPUT-出力値の制限

コンポーネントのインストール時に DEBUG = 1 が指定された場合、4 つの追加ピンがエクスポートされます。

- （FLOAT）PID。<LOOPNUM> .ERRORI-エラーの積分。
- （FLOAT）PID。<LOOPNUM> .ERRORD-エラーの導関数。
- （FLOAT）PID。<LOOPNUM> .COMMANDD-コマンドの派生物。
- （FLOAT）PID。<LOOPNUM> .COMMANDDD-コマンドの 2 次導関数。

関数コンポーネントは、PID ループごとに 1 つの関数をエクスポートします。 この関数は、ループに必要なすべての計算を実行します。 各ループには独自の機能があるため、個々のループをさまざまなスレッドに含めて、さまざまな速度で実行できます。

- （FUNCT）PID。<LOOPNUM> .DO_PID_CALCS-単一の PID ループに対してすべての計算を実行します。

PID ループの出力を計算するために使用される正確なアルゴリズムを理解したい場合は、図の PID ループブロック図、EMC2 / SRC / HAL / COMPONENTS / PID.C の先頭にあるコメント、そしてもちろんコードを参照してください。 自体。 ループ計算は C 関数 CALC_PID（）にあります。

12.11.5 シミュレートされたエンコーダ

シミュレートされたエンコーダはまさにそれです。 これは、HAL ピンによって制御される速度で、インデックスパルスを使用して直交パルスを生成します。 テストに最も役立ちます。

インストール

```
halcmd: loadrt sim-encoder num_chan=<number>
```

<NUMBER>は、シミュレートするエンコーダーの数です。 指定しない場合、エンコーダーが 1 つ取り付けられます。 最大数は 8 です（SIM_ENCODER.C の MAX_CHAN で定義されています）。

削除

```
halcmd: unloadrt sim-encoder
```

ピン

- （FLOAT）SIM-ENCODER。<CHAN-NUM> .SPEED-シミュレートされたシャフトの速度コマンド。
- （ビット）SIM-ENCODER。<CHAN-NUM> .PHASE-A-直交出力。
- （ビット）SIM-ENCODER。<CHAN-NUM> .PHASE-B-直交出力。
- （ビット）SIM-ENCODER。<CHAN-NUM> .PHASE-Z-インデックスパルス出力。

.SPEED が正の場合、.PHASE-A が.PHASE-B よりも進みます。

パラメーター

- （U32）SIM-ENCODER。<CHAN-NUM> .PPR-回転あたりのパルス数。
- （FLOAT）SIM-ENCODER。<CHAN-NUM> .SCALE-速度のスケール係数。 デフォルトは 1.0 です。これは、速度が 1 秒あたりの回転数であることを意味します。 RPM の場合は 60 に、1 秒あたりの度数の場合は 360 に、1 秒あたりのラジアンの場合は 6.283185 に変更します。

1 回転あたりのパルス数は 1 回転あたりのカウント数と同じではないことに注意してください。パルスは完全な直交サイクルです。 ほとんどのエンコーダカウンタは、1 つの完全なサイクル中に 4 回カウントされます。

関数コンポーネントは 2 つの関数をエクスポートします。 各関数は、シミュレートされたすべてのエンコーダーに影響します。

- （FUNCT）SIM-ENCODER.MAKE-PULSES-直交パルスを生成する高速関数（浮動小数点なし）。
- （FUNCT）SIM-ENCODER.UPDATE-SPEED-速度の読み取り、スケーリングの実行、およびメイクパルスの設定を行う低速関数。

12.11.6 Debounce

デバウンスは、機械的なスイッチ接点によって作成されたグリッチをフィルタリングできるリアルタイムコンポーネントです。 また、短いパルスを拒否する他のアプリケーションでも役立つ場合があります。

インストール

```
halcmd: loadrt debounce cfg=<config-string>
```

<CONFIG-STRING>は、コンマで区切られた一連の 10 進整数です。 各番号は、同一のデバウンスフィルターのグループをインストールします。番号は、グループ内のフィルターの数を決定します。

例えば：

```
halcmd: loadrt debounce cfg=1,4,2
```

フィルタの 3 つのグループをインストールします。 グループ 0 には 1 つのフィルターが含まれ、グループ 1 には 4 つのフィルターが含まれ、グループ 2 には 2 つのフィルターが含まれます。 <CONFIG-STRING>のデフォルト値は「1」で、単一のフィルターを含む単一のグループをインストールします。 グループの最大数 8（DEBOUNCE.C の MAX_GROUPS で定義）。 グループ内のフィルターの最大数は、共有メモリスペースによってのみ制限されます。 各グループは完全に独立しています。 1 つのグループ内のすべてのフィルターは同一であり、それらはすべて同じ関数によって同時に更新されます。 以下の説明では、<G>はグループ番号、<F>はグループ内のフィルター番号です。 最初のフィルターはグループ 0、フィルター 0 です。

削除

```
halcmd: unloadrt debounce
```

ピン個々のフィルターには 2 つのピンがあります。

- （BIT）デバウンス。<G>。<F> .IN-グループ<G>のフィルター<F>の入力。
- BIT）デバウンス。<G>。<F> .OUT-グループ<G>のフィルター<F>の出力。

パラメーターフィルターの各グループには、1 つのパラメーターがあります 9。

- S32）DEBOUNCE。<G> .DELAY-グループ<G>内のすべてのフィルターのフィルター遅延。

フィルタの遅延は、スレッド期間の単位です。 最小遅延はゼロです。 ゼロ遅延フィルターの出力は、その入力に正確に従います。何もフィルターしません。 遅延が増加するにつれて、ますます長いグリッチは拒否されます。 遅延が 4 の場合、4 スレッド期間以下のすべてのグリッチは拒否されます。

関数フィルターの各グループには、そのグループ内のすべてのフィルターを同時に更新する 1 つの関数があります。 フィルタのさまざまなグループは、さまざまなスレッドからさまざまな期間に更新できます。

- （機能）DEBOUNCE。<G>-グループ<G>内のすべてのフィルターを更新します。

9　個々のフィルターには、内部状態変数もあります。 その変数をパラメーターとしてエクスポートできるコンパイル時スイッチがあります。 これはテストを目的としており、通常の状況では共有メモリを浪費するだけです。

12.11.7　Siggen

SIGGEN は、方形波、三角波、正弦波を生成するリアルタイムコンポーネントです。 これは主にテストに使用されます。

インストール

```
halcmd: loadrt siggen [num_chan=<chans>]
```

<CHANS>は、インストールする信号発生器の数です。 NUMCHAN が指定されていない場合、1 つの信号発生器がインストールされます。 ジェネレーターの最大数は 16 です（SIGGEN.C の MAX_CHAN で定義されています）。 各ジェネレーターは完全に独立しています。 以下の説明では、<CHAN>は特定の信号発生器の番号です（番号は 0 から始まります）。

削除

```
halcmd: unloadrt siggen
```

ピン各ジェネレータには 5 つの出力ピンがあります。

- （FLOAT）SIGGEN。<CHAN> .SINE-正弦波出力。
- （FLOAT）SIGGEN。<CHAN> .COSINE-コサイン出力。
- （FLOAT）SIGGEN。<CHAN> .SAWTOOTH-鋸歯状の出力。
- （FLOAT）SIGGEN。<CHAN> .TRIANGLE-三角波出力。
- （FLOAT）SIGGEN。<CHAN> .SQUARE-方形波出力。

5 つの出力はすべて、同じ周波数、振幅、およびオフセットを持っています。

出力ピンに加えて、3 つの制御ピンがあります。

- （FLOAT）SIGGEN。<CHAN> .FREQUENCY-周波数をヘルツで設定します。デフォルト値は 1Hz です。
- （FLOAT）SIGGEN。<CHAN> .AMPLITUDE-出力波形のピーク振幅を設定します。デフォルトは 1 です。
- (float) siggen.<chan>.offset - Sets DC offset of the output waveforms, default is 0.

たとえば、SIGGEN.0.AMPLITUDE が 1.0 で、SIGGEN.0.OFFSET が 0.0 の場合、出力は-1.0 から+1.0 にスイングします。 SIGGEN.0.AMPLITUDE が 2.5 で、SIGGEN.0.OFFSET が 10.0 の場合、出力は 7.5 から 12.5 にスイングします。

パラメータなし。 10

関数

- （FUNCT）SIGGEN。<CHAN> .UPDATE-5 つの出力すべての新しい値を計算します。

10 バージョン 2.1 より前は、周波数、振幅、およびオフセットがパラメーターでした。 他のコンポーネントによる制御を可能にするために、ピンに変更されました。

12.11.8 lut5

LUT5 コンポーネントは、ルックアップテーブルに基づく 5 入力ロジックコンポーネントです。

LUT5 は浮動小数点スレッドを必要としません。

インストール

```
loadrt lut5 [count=N|names=name1[,name2...]]
addf lut5.N servo-thread | base-thread
setp lut5.N.function 0xN
```

関数の計算上から始まる関数の 16 進数を計算するには、1 または 0 を入力して、その行が TRUE か FALSE かを示します。 次に、出力列のすべての数値を上から順に書き、右から左に書きます。 これは 2 進数になります。 UBUNTU のようなプログラムビューで電卓を使用して、2 進数を入力し、それを 16 進数に変換すると、それが関数の値になります。

Table 14.1: Look Up Table

Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Output
0	0	0	0	0	
0	0	0	0	1	
0	0	0	1	0	
0	0	0	1	1	
0	0	1	0	0	
0	0	1	0	1	
0	0	1	1	0	
0	0	1	1	1	
0	1	0	0	0	
0	1	0	0	1	
0	1	0	1	0	
0	1	0	1	1	
0	1	1	0	0	
0	1	1	0	1	
0	1	1	1	0	
0	1	1	1	1	
1	0	0	0	0	
1	0	0	0	1	
1	0	0	1	0	
1	0	0	1	1	
1	0	1	0	0	
1	0	1	0	1	
1	0	1	1	0	
1	0	1	1	1	
1	1	0	0	0	
1	1	0	0	1	
1	1	0	1	0	
1	1	0	1	1	
1	1	1	0	0	
1	1	1	0	1	
1	1	1	1	0	
1	1	1	1	1	

2 つの入力例次の表では、真になりたい各行の出力状態を選択しています。

Table 14.2: Look Up Table

Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Output
0	0	0	0	0	0
0	0	0	0	1	1

Table 14.2: (continued)

Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Output
0	0	0	1	0	0
0	0	0	1	1	1

この例の出力列を見ると、ビット 0 またはビット 0 で、ビット 1 がオンで、他には何もないときに出力をオンにする必要があります。 2 進数は B1010 です（出力を 90 度 CW 回転させます）。 この数値を電卓に入力し、表示を 16 進数に変更すると、関数に必要な数値は 0XA になります。 16 進プレフィックスは 0X です。

12.12 HAL の例

これらの例はすべて、STEPCONF ベースの構成から開始し、ベーススレッドとサーボスレッドの 2 つのスレッドがあることを前提としています。 STEPCONF ウィザードは、空の CUSTOM.HAL ファイルと CUSTOM_POSTGUI.HAL ファイルを作成します。 CUSTOM.HAL ファイルは構成 HAL ファイルの後にロードされ、CUSTOM_POSTGUI.HAL ファイルは GUI のロード後にロードされます。

12.12.1 2 つの出力を接続する

2 つの出力を入力に接続するには、OR2 コンポーネントを使用できます。 OR2 はこのように機能し、OR2 への入力がオンの場合、OR2 出力はオンになります。 OR2 への入力がどちらもオンになっていない場合、OR2 出力はオフになります。

たとえば、2 つの PYVCP ボタンを両方とも 1 つの LED に接続します。

.XML ファイル

```
<pyvcp>
    <button>
        <halpin>"button-1"</halpin>
        <text>"Button 1"</text>
        </button>
    <button>
```

```
        <halpin>"button-2"</halpin>
        <text>"Button 2"</text>
    </button>
    <led>
        <halpin>"led-1"</halpin>
        <size>50</size>
        <on_color>"green"</on_color>
        <off_color>"red"</off_color>
    </led>
</pyvcp>
```

POSTGUI.HAL ファイル

```
loadrt or2
addf or2.0 servo-thread
net button-1 or2.0.in0 <= pyvcp.button-1
net button-2 or2.0.in1 <= pyvcp.button-2
net led-1 pyvcp.led-1 <= or2.0.out
```

STEPCONF WIZARD で作成された AXIS シミュレーターでこの例を実行すると、ターミナルを開いて、ターミナルで HALCMD SHOW PIN OR2 と入力することにより、LOADRTOR2 で作成されたピンを確認できます。

```
halcmd show pin or2
Component Pins:
Owner     Type     Dir      Value    Name
    22    bit      IN       FALSE    or2.0.in0 <== button-1
    22    bit      IN       FALSE    or2.0.in1 <== button-2
    22    bit      OUT      FALSE    or2.0.out ==> led-1
```

HAL コマンドの SHOWPIN OR2 から、BUTTON-1 ピンが OR2.0.IN0 ピンに接続されていることがわかります。方向矢印から、ボタンが出力され、OR2.0.IN0 が入力であることがわかります。 。 OR2 からの出力は、LED の入力に**送られます**。

12.12.2 手動ツールチェンジ

この例では、独自の構成をローリングしていて、HAL MANUALTOOLCHANGE ウィンドウを追加したいと想定しています。 HAL 手動ツールチェンジは、プリセット可能なツールがあり、オフセットをツールテーブルに保存する場合に主に役立ちます。 ツールを変更するたびにタッチオフする必要がある場合は、G コードを分割するのが最善です。 HAL 手動ツール変更ウィンドウを使用するには、基本的に HAL_MANUALTOOLCHANGE コンポーネントをロードしてから、IOCONTROL ツール変更を HAL_MANUALTOOLCHANGE 変更に送信し、HAL_MANUALTOOLCHANGE 変更を IOCONTROL ツール変更に送信する必要があります。 工具交換が完了したことを示すために、外部入力用のピンが用意されています。

これは、HAL 手動ツールチェンジコンポーネントを使用した手動ツールチェンジの例です。

```
loadusr -W hal_manualtoolchange
net tool-change iocontrol.0.tool-change => hal_manualtoolchange.change
net tool-changed iocontrol.0.tool-changed <= hal_manualtoolchange.changed
net external-tool-changed hal_manualtoolchange.change_button <= parport.0.pin-12-in
net tool-number iocontrol.0.tool-prep-number => hal_manualtoolchange.number
net tool-prepare-loopback iocontrol.0.tool-prepare => iocontrol.0.tool-prepared
```

これは、HAL 手動ツールチェンジコンポーネントを使用しない手動ツールチェンジの例です。

```
net tool-number <= iocontrol.0.tool-prep-number
net tool-change-loopback iocontrol.0.tool-change => iocontrol.0.tool-changed
net tool-prepare-loopback iocontrol.0.tool-prepare => iocontrol.0.tool-prepared
```

12.12.3 速度を計算する

この例では、DDT、MULT2、および ABS を使用して、単一軸の速度を計算します。 リアルタイムコンポーネントの詳細については、マニュアルページまたはリアルタイムコンポーネントのセクション（セクション 14.10.2）を参照してください。

まず、構成をチェックして、すべての準備が整ったリアルタイムコンポーネントを使用していないことを確認します。 これを行うには、HAL 構成ウィンドウを開き、ピンセクションでコンポーネントを探します。 次に、それらがロードされている.HAL ファイルを見つけたら、カウントを増やして、インスタンスを正しい値に調整します。 CUSTOM.HAL ファイルに以下を追加します。

リアルタイムコンポーネントをロードします。

```
loadrt ddt count=1
loadrt mult2 count=1
```

```
loadrt abs count=1
```

関数をスレッドに追加して、更新されるようにします。

```
addf ddt.0 servo-thread
addf mult2.0 servo-thread
addf abs.0 servo-thread
```

接続します。

```
setp mult2.in1 60
net xpos-cmd ddt.0.in
net X-IPS mult2.0.in0 <= ddt.0.out
net X-ABS abs.0.in <= mult2.0.out
net X-IPM abs.0.out
```

この最後のセクションでは、MULT2.0.IN1 を 60 に設定して、DDT.0.OUT から取得したインチ/秒をインチ/分に変換します。

XPOS-CMD は、コマンドされた位置を DDT.0.IN に送信します。 DDT は、入力の変化の導関数を計算します。

DDT2.0.OUT に 60 を掛けて、IPM を求めます。

MULT2.0.OUT は絶対値を取得するために ABS に送信されます。

次の図は、X 軸が 15IPM でマイナス方向に移動している場合の結果を示しています。 ABS.0.OUT ピンまたは X-IPM 信号のいずれかから絶対値を取得できることに注意してください。

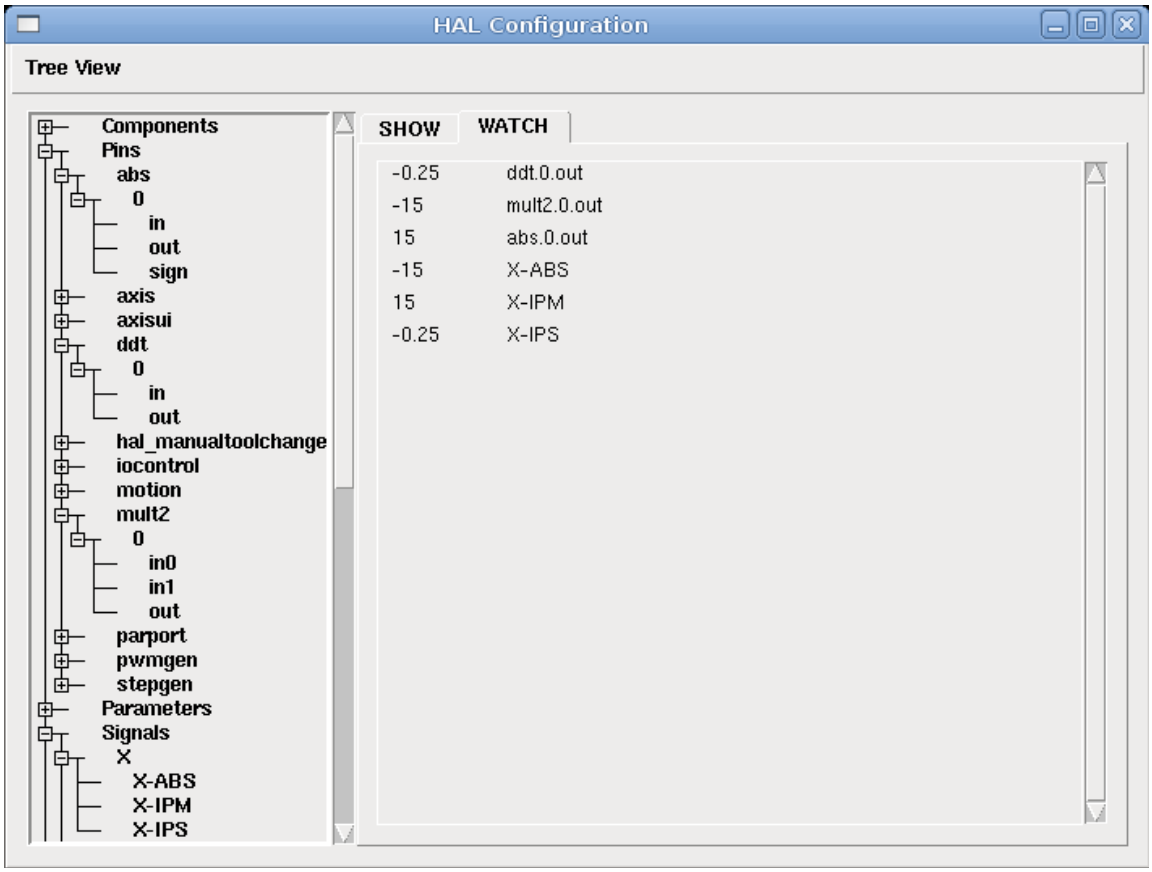


図 14-122

12.12.4 ソフトスタート

この例は、HAL コンポーネントのローパス、LIMIT2、または LIMIT3 を使用して、信号の変化速度を制限する方法を示しています。

この例では、旋盤のスピンドルを駆動するサーボモーターがあります。 サーボでコマンドされたスピンドル速度を使用した場合、現在の速度からコマンドされた速度にできるだけ速く移動しようとします。 これにより、問題が発生したり、ドライブが損傷したりする可能性があります。 変化の速度を遅くするために、PID の前にリミッターを介して SPINDLE.N.SPEED-OUT を送信して、PID コマンド値が新しい設定にゆっくりと変化するようにすることができます。

信号を制限する 3 つの組み込みコンポーネントは次のとおりです。

- LIMIT2 は、信号の範囲と 1 次導関数を制限します。
- LIMIT3 は、信号の範囲、1 次および 2 次導関数を制限します。
- ローパスは、指数関数的に重み付けされた移動平均を使用して入力信号を追跡します

これらの HAL コンポーネントの詳細については、マニュアルページを確認してください。

以下を SOFTSTART.HAL というテキストファイルに配置します。 LINUX に慣れていない場合は、ファイルをホームディレクトリに配置します。

```
loadrt threads period1=1000000 name1=thread
```

```
loadrt siggen
loadrt lowpass
loadrt limit2
loadrt limit3
net square siggen.0.square => lowpass.0.in limit2.0.in limit3.0.in
net lowpass <= lowpass.0.out
net limit2 <= limit2.0.out
net limit3 <= limit3.0.out
setp siggen.0.frequency .1
setp lowpass.0.gain .01
setp limit2.0.maxv 2
setp limit3.0.maxv 2
setp limit3.0.maxa 10
addf siggen.0.update thread
addf lowpass.0 thread
addf limit2.0 thread
addf limit3.0 thread
start
loadusr halscope
```

ターミナルウィンドウを開き、次のコマンドでファイルを実行します。

```
halrun -I softstart.hal
```

HAL オシロスコープが最初に起動したら、[OK]をクリックしてデフォルトのスレッドを受け入れます。

次に、信号をチャネルに追加する必要があります。 チャネル 1 をクリックし、[信号]タブから正方形を選択します。 チャネル 2〜4 について繰り返し、ローパス、LIMIT2、および LIMIT3 を追加します。

次に、トリガー信号を設定するには、[ソースなし]ボタンをクリックして、正方形を選択します。 ボタンがソースチャン 1 に変わります。次に、[実行モード]ラジオボタンボックスで[シングル]をクリックします。 これにより実行が開始され、実行が終了するとトレースが表示されます。

信号を分離して見やすくするには、チャンネルをクリックしてから、[垂直]ボックスの[位置]スライダーを使用して位置を設定します。

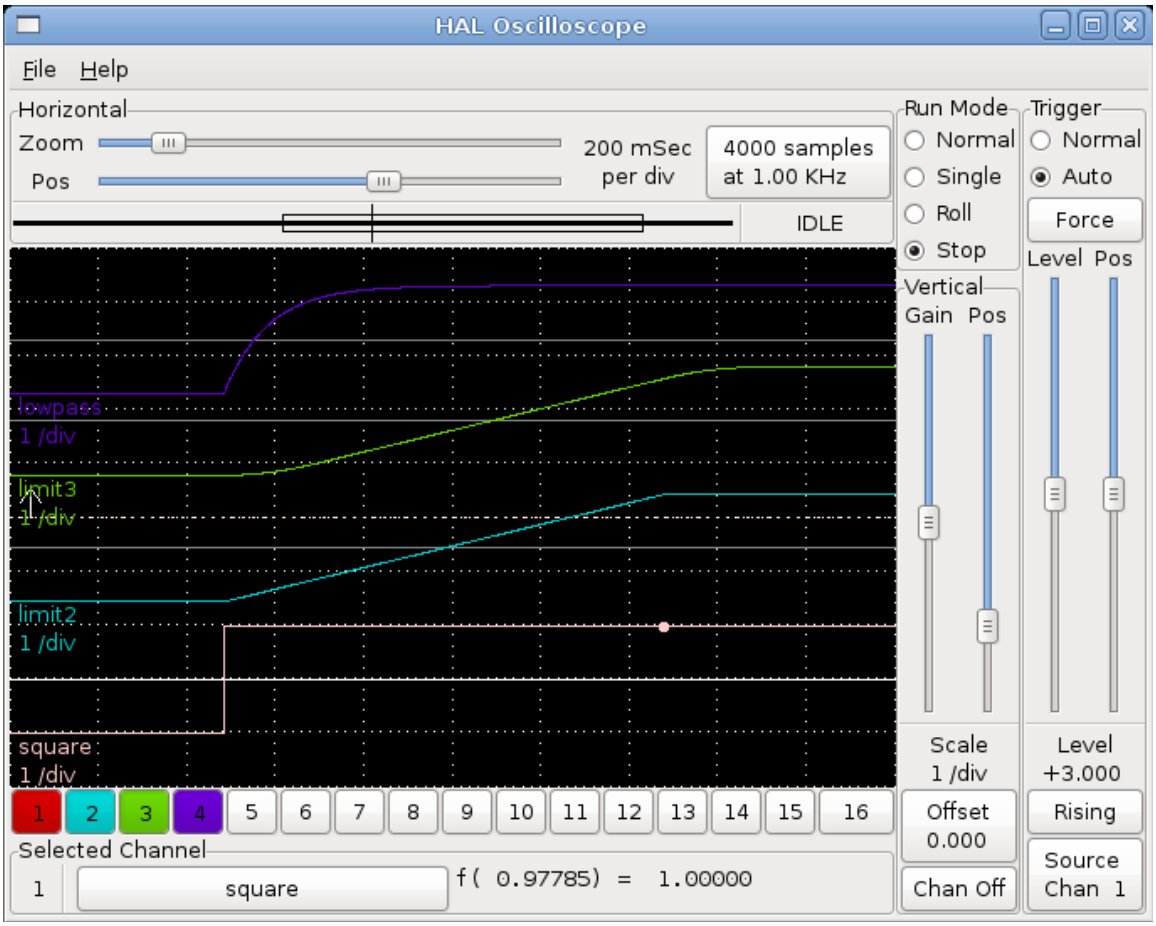


図 14-123

コンポーネントの設定値を変更した場合の影響を確認するには、ターミナルウィンドウで変更できます。 ローパスに対してさまざまなゲイン設定がどのように機能するかを確認するには、ターミナルウィンドウに次のように入力して、さまざまな設定を試してください。

```
setp lowpass.0.gain *.01
```

設定を変更した後、オシロスコープを再度実行して変更を確認します。

終了したら、ターミナルウィンドウで EXIT と入力して、HALRUN をシャットダウンし、HALSCOPE を閉じます。 HALRUN を実行している状態でターミナルウィンドウを閉じないでください。メモリに何かが残って、EMC の読み込みが妨げられる可能性があります。

HALSCOPE の詳細については、HAL のマニュアルを参照してください。

12.12.5 スタンドアロン HAL

場合によっては、HAL だけで GLADEVCP 画面を実行したいことがあります。 たとえば、ステッピングモーターを実行するだけでよいステッピング駆動デバイスがあるとします。 シンプルな開始/停止インターフェースがアプリケーションに必要なすべてであるため、本格的な CNC アプリケーションをロードして構成する必要はありません。

次の例では、1 つの単純な GLADEVCP パネルを作成しました。

BASIC SYNTAX

```
# load the winder.glade GUI and name it winder
loadusr -Wn winder gladevcp -c winder -u handler.py winder.glade

# load realtime components
loadrt threads name1=fast period1=50000 fp1=0 name2=slow period2=1000000
loadrt stepgen step_type=0 ctrl_type=v
loadrt hal_parport cfg="0x378 out"

# add functions to threads
addf stepgen.make-pulses fast
addf stepgen.update-freq slow
addf stepgen.capture-position slow
addf parport.0.read fast
addf parport.0.write fast


# make hal connections
net winder-step parport.0.pin-02-out <= stepgen.0.step
net winder-dir parport.0.pin-03-out <= stepgen.0.dir
net run-stepgen stepgen.0.enable <= winder.start_button


# start the threads
Start

# comment out the following lines while testing and use the interactive
# option halrun -I -f start.hal to be able to show pins etc.

# wait until the gladevcp GUI named winder terminates
waitusr winder
```

```
# stop HAL threads
Stop


# unload HAL all components before exiting
unloadrt all
```

12.13 HAL コンポーネントジェネレータ

12.13.1 前書き

HAL コンポーネントの作成は面倒なプロセスになる可能性があり、そのほとんどは RTAPI_および HAL_関数のセットアップ呼び出しと関連するエラーチェックで行われます。 HALCOMPILE は、このすべてのコードを自動的に記述します。

HAL コンポーネントのコンパイルは、コンポーネントが LINUXCNC ソースツリーの一部であるか外部であるかに関係なく、HALCOMPILE を使用する場合にもはるかに簡単です。

たとえば、C でコーディングする場合、「DDT」などの単純なコンポーネントは約 80 行のコードです。 HALCOMPILE プリプロセッサを使用して記述した場合、同等のコンポーネントは非常に短くなります。

単純なコンポーネントの例

```
component ddt "Compute the derivative of the input function";
pin in float in;
pin out float out;
variable double old;
function _;
license "GPL"; // indicates GPL v2 or later
;;
float tmp = in;
out = (tmp - old) / fperiod;
old = tmp;
```

12.13.2 インストール

コンポーネントをコンパイルするには、DEV をインストールする必要があります。

```
sudo apt install linuxcnc-dev
or
```

```
sudo apt install linuxcnc-uspace-dev
```

もう1つの方法は、SYNAPTIC PACKAGEMANAGER を使用して LINUXCNC-DEV または LINUXCNC-USPACE-DEV をインストールすることです。

12.13.3 コンポーネントの使用

コンポーネントを使用する前に、コンポーネントをロードしてスレッドに追加する必要があります。

例

```
loadrt threads name1=servo-thread period1=1000000
loadrt ddt
addf ddt.0 servo-thread
```

LOADRT と ADDF の詳細については、HALBASICS を参照してください。

コンポーネントをテストするには、HAL チュートリアルの例に従ってください。

12.13.4 定義

- コンポーネント-コンポーネントは、HALCMDLOADRT でロードされる単一のリアルタイムモジュールです。 1つの.COMP ファイルで1つのコンポーネントを指定します。 コンポーネント名とファイル名は一致している必要があります。
- インスタンス-コンポーネントは 0 個以上のインスタンスを持つことができます。 コンポーネントの各インスタンスは同じように作成されますが（すべて同じピン、パラメーター、関数、およびデータがあります）、ピン、パラメーター、およびデータの値が異なる場合は独立して動作します。
- シングルトン-コンポーネントが「シングルトン」である可能性があります。その場合、インスタンスが1つだけ作成されます。 システム内にその種のオブジェクトが文字通り1つしかない場合を除いて、シングルトンコンポーネントを作成することはほとんど意味がありません（たとえば、現在の UNIX 時間をピンに提供することを目的としたコンポーネント、または内蔵 PC スピーカー）

12.13.5 インスタンスの作成

シングルトンの場合、コンポーネントがロードされるときに1つのインスタンスが作成されます。

シングルトン以外の場合、COUNT モジュールパラメータは、作成される番号付きインスタンスの数を決定します。 COUNT が指定されていない場合、NAMES モジュールパラメーターは、作成される名前付きインスタンスの数を決定します。 COUNT も NAMES も指定されていない場合、単一の番号付きインスタンスが作成されます。

12.13.6 暗黙のパラメータ

関数には、コンポーネントを実行するための最後の期間のナノ秒単位の時間である期間パラメーターが暗黙的に渡されます。 浮動小数点を使用する関数は、秒単位の浮動小数点時間であるFPERIOD、または（PERIOD * 1E-9）を参照することもできます。 これは、タイミング情報を必要とするコンポーネントで役立ちます。

12.13.7 構文

.COMP ファイルは、いくつかの宣言とそれに続く;;で構成されます。 独自の行に、モジュールの機能を実装する C コードが続きます。

宣言は次のとおりです。

- COMPONENT HALNAME (DOC);
- PIN PINDIRECTION TYPE HALNAME ([SIZE]|[MAXSIZE: CONDSIZE]) (IF CONDITION) (= STARTVALUE) (DOC) ;
- PARAM PARAMDIRECTION TYPE HALNAME ([SIZE]|[MAXSIZE: CONDSIZE]) (IF CONDITION) (= STARTVALUE) (DOC);
- FUNCTION HALNAME (FP | NOFP) (DOC);
- OPTION OPT (VALUE);
- VARIABLE CTYPE STARREDNAME ([SIZE]);
- DESCRIPTION DOC;
- NOTES DOC;
- SEE_ALSO DOC;
- LICENSE LICENSE;
- AUTHOR AUTHOR;
- INCLUDE HEADERFILE;

括弧はオプション項目を示します。 縦棒は代替案を示します。 大文字の単語は、次のように可変テキストを示します。

- NAME-標準の C 識別子
- STARREDNAME-前に 0 個以上の*が付いた C 識別子。 この構文は、ポインターであるインスタンス変数を宣言するために使用できます。 文法上、*と変数名の間に空白がない場合があることに注意してください。
- HALNAME-拡張識別子。 HAL 識別子の作成に使用すると、アンダースコアはダッシュに置き換えられ、末尾のダッシュまたはピリオドは削除されるため、「THIS_NAME_」は「THIS-NAME」に変換されます。名前が「_」の場合は、 末尾のピリオドも削除されるため、

「FUNCTION_」は「COMPONENT。<NUM>」ではなく「COMPONENT。<NUM>」のような HAL 関数名を付けます。

存在する場合、接頭辞 HAL_は、ピン、パラメーター、および関数を作成するときにコンポーネント名の先頭から削除されます。 ピンまたはパラメーターの HALID で、＃は配列項目を示し、[SIZE]宣言と組み合わせて使用する必要があります。

ハッシュマークは、＃文字数と同じ長さの 0 が埋め込まれた数字に置き換えられます。 C 識別子の作成に使用すると、次の変更が HALNAME に適用されます。

1. 「#」文字、およびそれらの直前の「。」、「_」、または「-」文字はすべて削除されます。
2. 残りの「。」 および「-」文字は「_」に置き換えられます。
3. 繰り返される「_」文字は、単一の「\ _」文字に変更されます。

末尾の「_」は保持されるため、予約済みの名前やキーワード（MIN など）と衝突する HAL 識別子を使用できます。

HALNAME	C Identifier	HAL Identifier
x_y_z	x_y_z	x-y-z
x-y.z	x_y_z	x-y.z
x_y_z_	x_y_z_	x-y-z
x.##.y	x_y(MM)	x.MM.z
x.##	x(MM)	x.MM

- IFCONDITION-ピンまたはパラメーターを作成する必要があるときにゼロ以外の変数パーソナリティを含む式
- SIZE-配列のサイズを示す数値。 配列項目には、0 から SIZE-1 までの番号が付けられています。
- MAXSIZE：CONDSIZE-配列の最大サイズの後に変数のパーソナリティを含む式が続き、常に MAXSIZE 未満と評価される数値。 配列が作成されると、そのサイズは CONDSIZE になります。
- DOC-アイテムを文書化する文字列。 文字列は、次のような C スタイルの「二重引用符で囲まれた」文字列にすることができます。

```
"Selects the desired edge: TRUE means falling, FALSE means rising"
```

または、PYTHON スタイルの「三重引用符」文字列。これには、次のような改行と引用符が埋め込まれている場合があります。

```
"""The effect of this parameter, also known as "the orb of zot",
will require at least two paragraphs to explain.

Hopefully these paragraphs have allowed you to understand "zot"
better."""
```

または、文字列の前にリテラル文字 R を付けることもできます。その場合、文字列は PYTHON の RAW 文字列のように解釈されます。

ドキュメント文字列は「GROFF-MAN」形式です。 このマークアップ形式の詳細については、GROFF_MAN（7）を参照してください。 HALCOMPILE は文字列のバックスラッシュエスケープを解釈することを忘れないでください。たとえば、単語の例にイタリックフォントを設定するには、次のように記述します。

```
"\\flexample\\fB"
```

この場合、R 文字列の円記号を 2 倍にする必要がないため、R 文字列は特に便利です。

```
"\flexample\fB"
```

- TYPE-HAL タイプの 1 つ：ビット、符号付き、符号なし、または浮動小数点。 古い名前 S32 および U32 も使用できますが、符号付きおよび符号なしが推奨されます。
- PINDIRECTION-次のいずれか：IN、OUT、または IO。 コンポーネントは、出力ピンの値を設定し、入力ピンから値を読み取り、IO ピンの値を読み取ったり設定したりできます。
- PARAMDIRECTION-次のいずれか：R または RW。 コンポーネントは R パラメータの値を設定し、RW パラメータの値を読み取ったり設定したりできます。
- STARTVALUE-ピンまたはパラメータの初期値を指定します。 指定されていない場合、アイテムのタイプに応じて、デフォルトは 0 または FALSE になります。
- HEADERFILE-二重引用符（ "MYFILE.H";を含む）または山括弧（<SYSTEMFILE.H>;を含む）のいずれかでのヘッダーファイルの名前。 ヘッダーファイルは、ピンとパラメーターの宣言の前に、ファイルの先頭に（C の#INCLUDE を使用して）インクルードされます。

12.13.8 HAL 関数

- FP-関数が浮動小数点計算を実行することを示します。
- NOFP-整数計算のみを実行することを示します。 どちらも指定されていない場合、FP が想定されます。 HALCOMPILE も GCC も、NOFP とタグ付けされた関数での浮動小数点計算の使用を検出できませんが、そのような操作を使用すると、未定義の動作が発生します。

12.13.9 オプション

現在定義されているオプションは次のとおりです。

- OPTION SINGLETON YES-（デフォルト：NO）カウントモジュールパラメータを作成せず、常に単一のインスタンスを作成します。 シングルトンの場合、アイテムには COMPONENT-NAME.ITEM-NAME という名前が付けられ、シングルトンのない場合、番号付

きインスタンスのアイテムには COMPONENTNAME。<NUM> .ITEM-NAME という名前が付けられます。

- OPTION DEFAULT_COUNT NUMBER-（デフォルト：1）通常、モジュールパラメータのカウントはデフォルトで 1 になります。指定した場合、カウントはデフォルトでこの値になります。
- OPTION COUNT_FUNCTION YES-（デフォルト：NO）通常、作成するインスタンスの数は、モジュールパラメータ COUNT で指定されます。 COUNT_FUNCTION が指定されている場合、関数 INT GET_COUNT（VOID）によって返される値が代わりに使用され、COUNT モジュールパラメーターは定義されません。
- OPTION RTAPI_APPNO-（デフォルト：YES）通常、関数 RTAPI_APP_MAIN（）および RTAPI_APP_EXIT（）は自動的に定義されます。 オプション RTAPI_APPNO を使用すると、そうではなく、C コードで指定する必要があります。 次のプロトタイプを使用します。
  INT RTAPI_APP_MAIN(VOID);
  VOID RTAPI_APP_EXIT(VOID);
  独自の RTAPI_APP_MAIN（）を実装する場合は、関数 INT EXPORT（CHAR * PREFIX、LONG EXTRA_ARG）を呼び出して、プレフィックスのピン、パラメーター、および関数を登録します。
- OPTION DATA TYPE-（デフォルト：なし）非推奨指定された場合、コンポーネントの各インスタンスには、タイプ TYPE のデータブロックが関連付けられます（FLOAT のような単純なタイプ、または TYPEDEF で作成されたタイプの名前にすることができます）。新しいコンポーネントでは、代わりに変数を使用する必要があります。
- OPTION EXTRA_SETUP YES-（デフォルト：NO）指定されている場合、インスタンスごとに EXTRA_SETUP で定義されている関数を呼び出します。 自動的に定義された RTAPI_APP_MAIN を使用する場合、EXTRA_ARG はこのインスタンスの番号です。
- OPTION EXTRA_CLEANUP YES-（デフォルト：NO）指定されている場合、自動定義された RTAPI_APP_EXIT から EXTRA_CLEANUP で定義された関数を呼び出すか、自動定義された RTAPI_APP_MAIN でエラーが検出された場合。
- OPTION USERSPACE YES-（デフォルト：NO）指定されている場合、このファイルは、通常の（つまり、リアルタイムの）コンポーネントではなく、ユーザースペース（つまり、非リアルタイム）のコンポーネントを記述します。 ユーザースペースコンポーネントには、FUNCTION ディレクティブで定義された関数が含まれていない場合があります。代わりに、すべてのインスタンスが構築された後、C 関数 VOID USER_MAINLOOP（VOID）; と呼ばれます。 この関数が戻ると、コンポーネントは終了します。 通常、USER_MAINLOOP（）は FOR_ALL_INSTS（）を使用して各インスタンスの更新アクションを実行してから、短時間スリープします。 USER_MAINLOOP（）のも

う 1 つの一般的なアクションは、GUI ツールキットのイベントハンドラーループを呼び出すことです。

- OPTION USERINIT YES-（デフォルト：NO）オプション USERSPACE（上記を参照）が NO に設定されている場合、このオプションは無視されます。 USERINIT が指定されている場合、関数 USERINIT（ARGC、ARGV）は、RTAPI_APP_MAIN（）の前に（したがって、HAL_INIT（）の呼び出しの前に）呼び出されます。 この関数は、コマンドライン引数を処理したり、他のアクションを実行したりする場合があります。 その戻りタイプは無効です。 HAL コンポーネントを作成するのではなく終了したい場合（たとえば、コマンドライン引数が無効だったため）、EXIT（）を呼び出すことができます。
- OPTION EXTRA_LINK_ARGS "。。。"-（デフォルト： ""）オプション USERSPACE（上記を参照）が NO に設定されている場合、このオプションは無視されます。 ユーザースペースコンポーネントをリンクする場合、指定された引数がリンク行に挿入されます。 コンパイルは一時ディレクトリ「-L」で行われるため、注意してください。 .COMP ソースファイルが存在するディレクトリではなく、一時ディレクトリを参照します。
- OPTION EXTRA_COMPILE_ARGS "。。。"-（デフォルト： ""）オプション USERSPACE（上記を参照）が NO に設定されている場合、このオプションは無視されます。 ユーザースペースコンポーネントをコンパイルするとき、指定された引数はコンパイラのコマンドラインに挿入されます。

オプションの VALUE が指定されていない場合は、オプションを指定するのと同じです。 。。。はい。 オプションに不適切な値を割り当てた結果は未定義です。 他のオプションを使用した結果は未定義です。

12.13.10 ライセンスとオーサーシップ

- LICENSE-ドキュメントおよび MODULE_LICENSE（）モジュール宣言のモジュールのライセンスを指定します。 たとえば、モジュールのライセンスが GPL v2 以降であることを指定するには、

```
license "GPL"; // indicates GPL v2 or later
```

MODULE_LICENSE（）の意味および追加のライセンス識別子の詳細については、<LINUX / MODULE.H>を参照してください。 またはマニュアルページ RTAPI_MODULE_PARAM（3）

この宣言は必須です。

AUTHOR - SPECIFY THE AUTHOR OF THE MODULE FOR THE DOCUMENTATION.

12.13.11 インスタンスごとのデータストレージ

- VARIABLE CTYPE STARREDNAME;
- VARIABLE CTYPE STARREDNAME[SIZE];
- VARIABLE CTYPE STARREDNAME = DEFAULT;

- VARIABLE CTYPE STARREDNAME[SIZE] = DEFAULT;

タイプ CTYPE のインスタンスごとの変数 STARREDNAME を、オプションで SIZE アイテムの配列として、オプションでデフォルト値 DEFAULT を使用して宣言します。 DEFAULT のないアイテムは、ALL-BITS-ZERO に初期化されます。 CTYPE は、FLOAT、U32、S32、INT などの単純な 1 ワードの C タイプです。配列変数へのアクセスには角括弧が使用されます。

変数をポインタ型にする場合は、「*」と変数名の間にスペースを入れないでください。したがって、次のことが許容されます。

```
variable int *example;
```

しかし、以下はそうではありません：

```
variable int* badexample;
variable int * badexample;
```

12.13.12 コメント

C ++スタイルの 1 行コメント（//。。。）および

C スタイルの複数行コメント（/*。。。*/）は、どちらも宣言セクションでサポートされています。

12.13.13 制限

HAL はピン、パラメーター、および関数に同じ名前を付けることを許可しますが、HALCOMPILE は許可しません。

使用できない、または問題を引き起こす可能性のある変数名と関数名には、次のものがあります。

- _COMP で始まるもの。
- COMP_ID
- FPERIOD
- RTAPI_APP_MAIN
- RTAPI_APP_EXIT
- EXTRA_SETUP
- EXTRA_CLEANUP

12.13.14 コンビニエンスマクロ

宣言セクションの項目に基づいて、HALCOMPILE は STRUCT__COMP_STATE と呼ばれる C 構造体を作成します。 ただし、この構造体のメンバー（*（INST-> NAME）など）を参照する代わりに、通常、以下のマクロを使用して参照されます。 STRUCT __COMP_STATE およびこれらのマクロの詳細は、HALCOMPILE のバージョンごとに変わる可能性があります。

- FUNCTION（NAME）-このマクロを使用して、以前に関数 NAME で宣言されたリアルタイム関数の定義を開始します。 関数には、関数の呼び出し間のナノ秒の整数であるパラメーター期間が含まれています。
- EXTRA_SETUP（）-このマクロを使用して、このインスタンスの追加セットアップを実行するために呼び出される関数の定義を開始します。 失敗を示すには負の UNIXERRNO 値を返します（たとえば、I / O ポートの予約に失敗した場合は-EBUSY を返します）、または成功を示すには 0 を返します。
- EXTRA_CLEANUP（）-このマクロを使用して、コンポーネントの追加のクリーンアップを実行するために呼び出される関数の定義を開始します。 この関数は、1 つだけでなく、コンポーネントのすべてのインスタンスをクリーンアップする必要があることに注意してください。 「PIN_NAME」、「PARAMETER_NAME」、および「DATA」マクロはここでは使用できません。
- PIN_NAME または PARAMETER_NAME-各ピンの PIN_NAME または PARAMPARAMETER_NAME には、名前を単独で使用してピンまたはパラメーターを参照できるようにするマクロがあります。 PIN_NAME または PARAMETER_NAME が配列の場合、マクロは PIN_NAME（IDX）または PARAM_NAME（IDX）の形式になります。ここで、IDX はピン配列へのインデックスです。 配列が可変サイズの配列である場合、その CONDSIZE までのアイテムを参照することは合法です。 アイテムが条件付きアイテムである場合、その条件がゼロ以外の値に評価されたときにのみ参照することが合法です。
- VARIABLE_NAME-変数ごとに VARIABLE_NAME には、変数を参照するために名前を単独で使用できるようにするマクロがあります。 VARIABLE_NAME が配列の場合、通常の C スタイルの添え字が使用されます：VARIABLE_NAME [IDX]
- DATA-「オプションデータ」が指定されている場合、このマクロはインスタンスデータへのアクセスを許可します。
- FPERIOD-このリアルタイム関数の呼び出し間の浮動小数点秒数。
- FOR_ALL_INSTS（）{。 。 。 }-ユーザースペースコンポーネント用。 このマクロは、定義されたすべてのインスタンスを繰り返し処理します。 ループの本体内では、PIN_NAME、PARAMETER_NAME、および DATA マクロは、リアルタイム関数の場合と同じように機能します。

12.13.15 1 つの機能を持つコンポーネント

コンポーネントに関数が 1 つしかなく、文字列「FUNCTION」が;;の後のどこにも表示されない場合、;;の後の部分。 すべてがコンポーネントの単一機能の本体であると見なされます。 この例については、SIMPLECOMP を参照してください。

12.13.16 コンポーネントのパーソナリティ

コンポーネントに「IF 条件」または「[MAXSIZE：CONDSIZE]」のピンまたはパラメーターがある場合、それはパーソナリティのあるコンポーネントと呼ばれます。 各インスタンスのパーソナリティは、モジュールのロード時に指定されます。 パーソナリティは、必要な場合にのみピンを作成するために使用できます。 たとえば、パーソナリティは論理コンポーネントで使用され、各論理ゲートへの可変数の入力ピンを許可し、基本的なブール論理関数 AND、OR、および XOR のいずれかを選択できるようにします。

許可されるパーソナリティアイテムのデフォルト数は、コンパイル時の設定（64）です。 デフォルトは、HALCOMPILE を使用して構築されたディストリビューションに含まれる多数のコンポーネントに適用されます。

ユーザー作成コンポーネントに許可されるパーソナリティアイテムの数を変更するには、HALCOMPILE で--PERSONALITY オプションを使用します。 たとえば、最大 128 のパーソナリティ時間を許可するには：

```
[sudo] halcompile --personality=128 --install ...
```

パーソナリティを持つコンポーネントを使用する場合、通常の使用法は、指定されたコンポーネントインスタンスごとにパーソナリティアイテムを指定することです。 ロジックコンポーネントの 3 つのインスタンスの例：

```
loadrt logic names=and4,or3,nand5, personality=0x104,0x203,0x805
```

注意

LOADRT 行でパーソナリティよりも多くのインスタンスが指定されている場合、パーソナリティが指定されていないインスタンスにはパーソナリティ 0 が割り当てられます。要求されたインスタンス数が許可されたパーソナリティの数を超える場合、パーソナリティは許可されたパーソナリティの数を法としてインデックス付けすることによって割り当てられます。 そのような割り当てを示すメッセージが出力されます。

12.13.17 コンパイル

.COMP ファイルをソースディレクトリ LINUXCNC / SRC / HAL / COMPONENTS に配置し、MAKE を再実行します。 COMP ファイルはビルドシステムによって自動的に検出されます。

.COMP ファイルがハードウェアのドライバーである場合、それは LINUXCNC / SRC / HAL / DRIVERS に配置される可能性があり、LINUXCNC がユーザースペースシミュレーターとして構成されていない限りビルドされます。

12.13.18 ソースツリーの外部でリアルタイムコンポーネントをコンパイルする

HALCOMPILE は、リアルタイムコンポーネントを単一のステップで処理、コンパイル、およびインストールでき、RTEXAMPLE.KO を LINUXCNC リアルタイムモジュールディレクトリに配置します。

```
[sudo] halcompile --install rtexample.comp
```

NOTE

DEB パッケージのインストールから LINUXCNC を使用する場合は、SUDO（ROOT 権限用）が必要です。RUN-IN-PLACE（RIP）ビルドを使用する場合、ROOT 権限は必要ありません。

または、EXAMPLE.KO（またはシミュレーターの場合は EXAMPLE.SO）を現在のディレクトリに残して、1 つのステップで処理およびコンパイルすることもできます。

```
halcompile --compile rtexample.comp
```

または、EXAMPLE.C を現在のディレクトリに残して、単純に処理することもできます。

```
halcompile rtexample.comp
```

HALCOMPILE は、上記の--INSTALL および--COMPILE オプションを使用して、C で記述されたコンポーネントをコンパイルおよびインストールすることもできます。

```
[sudo] halcompile --install rtexample2.c
```

MAN 形式のドキュメントは、宣言セクションの情報から作成することもできます。

```
halcompile --document rtexample.comp
```

結果のマンページ、EXAMPLE.9 は次のように表示できます。

```
man ./example.9
```

または、マニュアルページの標準の場所にコピーします。

12.13.19 ソースツリーの外部でユーザースペースコンポーネントをコンパイルする

HALCOMPILE は、ユーザースペースコンポーネントを処理、コンパイル、インストール、および文書化できます。

```
halcompile usrexample.comp
halcompile --compile usrexample.comp
[sudo] halcompile --install usrexample.comp
halcompile --document usrexample.comp
```

これは.COMP ファイルに対してのみ機能し、.C ファイルに対しては機能しません。

12.13.20 例

12.13.21 絶え間ない

FUNCTION_」宣言は「CONSTANT.0」などの名前の関数を作成することに注意してください。 ファイル名はコンポーネント名と一致する必要があります。

component constant;

```
pin out float out;
param r float value = 1.0;
function _;
license "GPL"; // indicates GPL v2 or later
;;
FUNCTION(_) { out = value; }
```

12.13.22 SINCOS

このコンポーネントは、入力角度のサインとコサインをラジアンで計算します。 入力は「周波数」パラメータに基づいて自由に実行されるのではなく、角度であるため、SIGGEN の「正弦」および「余弦」出力とは異なる機能があります。

ピンは、関数 SIN（）および COS（）に干渉しないように、ソースコードで SIN_および COS_という名前で宣言されています。 HAL ピンはまだ SINCOS。<NUM> .SIN と呼ばれています。

```
component sincos;
pin out float sin_;
pin out float cos_;
pin in float theta;
function _;
license "GPL"; // indicates GPL v2 or later
;;
#include <rtapi_math.h>
FUNCTION(_) { sin_ = sin(theta); cos_ = cos(theta); }
```

12.13.23 out8

このコンポーネントは、「OUT8」と呼ばれる架空のカードのドライバーです。このカードには、単一の 8 ビット値として扱われる 8 ピンのデジタル出力があります。 システムにはさまざまな数のそのようなカードが存在する可能性があり、それらはさまざまなアドレスに存在する可能性があります。 OUT は<ASM / IO.H>で使用される識別子であるため、ピンは OUT_と呼ばれます。 これは、EXTRA_SETUP および EXTRA_CLEANUP を使用して I / O

領域を要求し、エラーが発生した場合またはモジュールがアンロードされた場合にそれを解放する方法を示しています。

```
component out8;
pin out unsigned out_ "Output value; only low 8 bits are used";
param r unsigned ioaddr;

function _;

option count_function;
option extra_setup;
option extra_cleanup;
option constructable no;

license "GPL"; // indicates GPL v2 or later
;;
#include <asm/io.h>
#define MAX 8
int io[MAX] = {0,};
RTAPI_MP_ARRAY_INT(io, MAX, "I/O addresses of out8 boards");
int get_count(void) {
int i = 0;
for(i=0; i<MAX && io[i]; i++) { /* Nothing */ }
return i;
}
EXTRA_SETUP() {
if(!rtapi_request_region(io[extra_arg], 1, "out8")) {
// set this I/O port to 0 so that EXTRA_CLEANUP does not release the IO
// ports that were never requested.
io[extra_arg] = 0;
return -EBUSY;
}
ioaddr = io[extra_arg];
return 0; }
EXTRA_CLEANUP() {
int i;
for(i=0; i < MAX && io[i]; i++) {
rtapi_release_region(io[i], 1);
}
}
FUNCTION(_) { outb(out_, ioaddr); }
```

12.13.24 hal_loop

```
component hal_loop;
pin out float example;
```

コンポーネントのこのフラグメントは、コンポーネント名での HAL_プレフィックスの使用を示しています。 LOOP は標準の LINUX カーネルモジュールの名前であるため、LINUX ループモジュールもシステムに存在する場合、ループコンポーネントが正常にロードされない可能性があります。

ロードされると、HALCMD SHOWCOMP は HAL_LOOP と呼ばれるコンポーネントを表示します。 ただし、HALCMD SHOW PIN で示されるピンは、HAL-LOOP.0.EXAMPLE ではなく LOOP.0.EXAMPLE になります。

12.13.25 配列デモ

このリアルタイムコンポーネントは、固定サイズの配列の使用法を示しています。

```
component arraydemo "4-bit Shift register";
pin in bit in;
pin out bit out-# [4];
function _ nofp;
license "GPL"; // indicates GPL v2 or later
;;
int i;
for(i=3; i>0; i--) out(i) = out(i-1);
out(0) = in;
```

12.13.26 rand

このユーザースペースコンポーネントは、出力ピンの値を（0,1）の範囲の新しいランダム値に約 1MS ごとに 1 回変更します。

```
component rand;
option userspace;
pin out float out;
license "GPL"; // indicates GPL v2 or later
;;
#include <unistd.h>

void user_mainloop(void) {
while(1) {
usleep(1000);
```

```
FOR_ALL_INSTS() out = drand48();
}
}
```

12.13.27 logic

このリアルタイムコンポーネントは、「パーソナリティ」を使用して可変サイズの配列とオプションのピンを作成する方法を示しています。

```
component logic "LinuxCNC HAL component providing experimental logic functions";
pin in bit in-##[16 : personality & 0xff];
pin out bit and if personality & 0x100;
pin out bit or if personality & 0x200;
pin out bit xor if personality & 0x400;
function _ nofp;
description """
Experimental general 'logic function' component. Can perform 'and', 'or'
and 'xor' of up to 16 inputs. Determine the proper value for 'personality'
by adding:
.IP \\(bu 4
The number of input pins, usually from 2 to 16
.IP \\(bu
256 (0x100) if the 'and' output is desired
.IP \\(bu
512 (0x200) if the 'or' output is desired
.IP \\(bu
1024 (0x400) if the 'xor' (exclusive or) output is desired""";
license "GPL"; // indicates GPL v2 or later
;;
FUNCTION(_) {
int i, a=1, o=0, x=0;
for(i=0; i < (personality & 0xff); i++) {
if(in(i)) { o = 1; x = !x; }
else { a = 0; }
}
if(personality & 0x100) and = a;
if(personality & 0x200) or = o;
if(personality & 0x400) xor = x;
}
```

このコンポーネントの一般的な負荷線は次のようになります。

```
loadrt logic count=3 personality=0x102,0x305,0x503
```

これにより、次のピンが作成されます。

- 2 入力 AND ゲート：LOGIC.0.AND、LOGIC.0.IN-00、LOGIC.0.IN-01
- 5 入力 AND および OR ゲート：LOGIC.1.AND、LOGIC.1.OR、LOGIC.1.IN-00、LOGIC.1.IN-01、LOGIC.1.IN-02、LOGIC.1.IN- 03、LOGIC.1.IN-04、
- 3 入力 AND および XOR ゲート：LOGIC.2.AND、LOGIC.2.XOR、LOGIC.2.IN-00、LOGIC.2.IN-01、LOGIC.2.IN-02

12.13.28 一般的な機能

この例は、MAIN 関数から関数を呼び出す方法を示しています。

また、HAL ピンの参照をこれらの関数に渡す方法も示しています。

```
component example;
pin in s32 in;
pin out bit out1;
pin out bit out2;
function _;
license "GPL";
;;
// general pin set true function
void set(hal_bit_t *p){
*p = 1;
}
// general pin set false function
void unset(hal_bit_t *p){
*p = 0;
}
//main function
FUNCTION(_) {
if (in < 0){
set(&out1);
unset(&out2);
}else if (in >0){
unset(&out2);
set(&out2);
}else{
unset(&out1);
unset(&out2);
```

```
}
}
```

このコンポーネントは、2 つの一般的な関数を使用して、それを参照する HAL ビットピンを操作します。

12.13.29 コマンドラインの使用法

HALCOMPILE のマニュアルページには、HALCOMPILE を呼び出すための詳細が記載されています。

```
$ man halcompile
```

HALCOMPILE の使用法の簡単な要約は次のとおりです。

```
$ halcompile --help
```

12.14 ユーザースペース PYTHON コンポーネントの作成

12.14.1 基本的な使い方

ユーザースペースコンポーネントは、ピンとパラメーターを作成することから始め、ループに入り、入力からのすべての出力を定期的に駆動します。 次のコンポーネントは、入力ピン（PASSTHROUGH.IN）に表示される値を出力ピン（PASSTHROUGH.OUT）に約 1 秒に 1 回コピーします。

```
#!/usr/bin/env python
import hal, time
h = hal.component("passthrough")
h.newpin("in", hal.HAL_FLOAT, hal.HAL_IN)
h.newpin("out", hal.HAL_FLOAT, hal.HAL_OUT)
h.ready()
try:
while 1:
time.sleep(1)
h['out'] = h['in']
except KeyboardInterrupt:
raise SystemExit
```

上記のリストを「PASSTHROUGH」という名前のファイルにコピーし、実行可能（CHMOD + X）にして、$ PATH に配置します。 次に、試してみてください。

```
Halrun

halcmd: loadusr passthrough

```

```
halcmd: show pin

Component Pins:
Owner   Type  Dir         Value  Name
03   float  IN          0             passthrough.in
03   float  OUT         0             passthrough.out
halcmd: setp passthrough.in 3.14

halcmd: show pin

Component Pins:
Owner   Type  Dir         Value  Name
03   float  IN          3.14          passthrough.in
03   float  OUT         3.14          passthrough.out
```

12.14.2 ユーザースペースのコンポーネントと遅延

「SHOWPIN」とすばやく入力すると、PASSTHROUGH.OUT の古い値が 0 のままであることがわかります。これは、TIME.SLEEP（1）が呼び出され、出力ピンへの割り当てが最大で 1 回行われるためです。 毎秒。 これはユーザースペースコンポーネントであるため、パススルーコンポーネントによって使用されるメモリがディスクにスワップされると、割り当て間の実際の遅延がはるかに長くなる可能性があり、そのメモリがスワップインされるまで割り当てが遅延する可能性があります。

したがって、ユーザースペースコンポーネントは、コントロールパネルなどのユーザーインタラクティブ要素には適していますが（ミリ秒の範囲の遅延は認識されず、より長い遅延は許容されます）、ステッパードライバーボードにステップパルスを送信することには適していません（遅延は常に 何があっても、マイクロ秒の範囲）。

12.14.3 ピンとパラメータを作成する

```
h = hal.component("passthrough")
```

コンポーネント自体は、コンストラクター HAL.COMPONENT の呼び出しによって作成されます。引数は、HAL コンポーネント名と、（オプションで）ピン名とパラメーター名に使用されるプレフィックスです。 プレフィックスが指定されていない場合は、コンポーネント名が使用されます。

```
h.newpin("in", hal.HAL_FLOAT, hal.HAL_IN)
```

次に、コンポーネントオブジェクトのメソッドを呼び出すことによってピンが作成されます。引数は、ピン名のサフィックス、ピンタイプ、およびピンの方向です。 パラメーターの場合、引数は、パラメーター名のサフィックス、パラメーターのタイプ、およびパラメーターの方向です。

Table 14.3: HAL Option Names

Pin and Parameter Types:	HAL_BIT	HAL_FLOAT	HAL_S32	HAL_U32
Pin Directions:	HAL_IN	HAL_OUT	HAL_IO	
Parameter Directions:	HAL_RO	HAL_RW		

完全なピンまたはパラメーター名は、プレフィックスとサフィックスを「。」で結合することによって形成されるため、この例では、作成されるピンは PASSTHROUGH.IN と呼ばれます。

```
h.ready()
```

すべてのピンとパラメーターが作成されたら、.READY（）メソッドを呼び出します。

1.1.1.1 プレフィックスの変更

プレフィックスは、.SETPREFIX（）メソッドを呼び出すことで変更できます。 現在のプレフィックスは、.GETPREFIX（）メソッドを呼び出すことで取得できます。

12.14.4 ピンとパラメータの読み取りと書き込み

適切な PYTHON 識別子でもあるピンとパラメーターの場合、属性構文を使用して値にアクセスまたは設定できます。

```
h.out = h.in
```

すべてのピンについて、それらが適切な PYTHON 識別子であるかどうかに関係なく、値は添え字構文を使用してアクセスまたは設定できます。

```
h['out'] = h['in']
```

1.1.1.1 駆動出力（HAL_OUT）ピン

定期的に、通常はタイマーに応答して、すべての HAL_OUT ピンに新しい値を割り当てることによって「駆動」する必要があります。 これは、値が最後に割り当てられた値と異なるかどうかに関係なく実行する必要があります。 ピンが信号に接続されている場合、その古い出力値は信号にコピーされないため、コンポーネントが新しい値を割り当てた場合にのみ、適切な値が信号に表示されます。

12.14.4.1 双方向（HAL_IO）ピンの駆動

上記のルールは双方向ピンには適用されません。 代わりに、コンポーネントが値を変更したい場合にのみ、双方向ピンをコンポーネントによって駆動する必要があります。 たとえば、正規のエンコーダインターフェイスでは、エンコーダコンポーネントはインデックスイネーブルピンを FALSE に設定するだけで（インデックスパルスが見られ、古い値が TRUE の場合）、TRUE に設定することはありません。 ピンを繰り返し FALSE で駆動すると、接続されている他のコンポーネントが、別のインデックスパルスが見られたかのように動作する可能性があります。

12.14.5 終了

コンポーネントの HALCMD アンロード要求は、KEYBOARDINTERRUPT 例外として配信されます。アンロード要求が到着すると、プロセスは短時間で終了するか、シャットダウンプロセスを完了するためにかなりの作業（ファイルの読み取りや書き込みなど）を実行する必要がある場合は、コンポーネントで.EXIT（）メソッドを呼び出す必要があります。

12.14.6 役立つ機能

1.1.1.1 COMPONENT_EXISTS

この時点で、指定されたコンポーネントは存在しますか？

例：

HAL.COMPONENT_EXISTS（ "TESTPANEL"）

12.14.6.1 component_is_ready

指定されたコンポーネントはこの時点で準備ができていますか。

例：

HAL.COMPONENT_IS_READY（ "TESTPANEL"）

12.14.6.2 GET_MSG_LEVEL

現在のリアルタイムメッセージレベルを取得します。

12.14.6.3 SET_MSG_LEVEL

現在のリアルタイムメッセージレベルを設定します。

情報のデバッグに使用されます。

12.14.6.4 connect

ピンを信号に接続します。

例：

HAL.CONNECT（ "PINNAME"、 "SIGNAL_NAME"）

12.14.6.5 get_value

ピン、パラメータ、または信号を直接読み取ります。

例：

VALUE = HAL.GET_VALUE（ "IOCONTROL.0.EMC-ENABLE-IN"）

12.14.6.6 new_signal

指定されたタイプの新しい信号を作成します。

例

HAL.NEW_SIG（ "SIGNALNAME"、HAL.HAL_BIT）

12.14.6.7 pin_has_writer

指定されたピンには駆動ピンが接続されていますか？

TRUE または FALSE を返します。

H.IN.PIN_HAS_WRITER（）

12.14.6.8 get_name

HAL オブジェクト名を取得します

H.IN.GET_NAME（）

文字列を返す

12.14.6.9 get_type

HAL オブジェクトのタイプを取得します

H.IN.GET_TYPE（）

整数を返します

12.14.6.10 get_dir

HAL オブジェクトの方向タイプを取得します

H.IN.GET_DIR（）

整数を返します

12.14.6.11 get

HAL オブジェクト値を取得します

H.IN.GET（）

12.14.6.12 set

HAL オブジェクト値を設定します

H.OUT.SET（10）

12.14.6.13 is_pin

オブジェクトはピンですか、それともパラメータですか？

H.IN.IS_PIN（）

BOOL を返します

12.14.6.14 sampler_base

TODO

12.14.6.15 stream_base

TODO

12.14.6.16 stream

TODO

12.14.6.17 set_p

HAL システムの任意のピンのピン値を設定します。

例：

HAL.SET_P（ "PINNAME"、 "10"）

12.14.7 定数

これらを使用して、保持する値ではなく詳細を指定します。

- HAL_BIT
- HAL_FLOAT
- HAL_S32
- HAL_U32
- HAL_IN
- HAL_OUT
- HAL_RO
- HAL_RW

- MSG_NONE
- MSG_ALL
- MSG_DBG
- MSG_ERR
- MSG_INFO
- MSG_WARN

12.14.8 システムインフォメーション

これらを読んで、リアルタイムシステムに関する情報を入手してください。

- IS_KERNELSPACE
- IS_RT
- IS_SIM
- IS_USERSPACE

12.14.9 GLADEVCP ハンドラーで HAL_GLIB とともに使用する

GLADEVCP は HAL_GLIB ライブラリを使用します。このライブラリは、HAL 入力ピンに「ウォッチャー」信号を接続するために使用できます。 この信号は、HAL ピンの状態が変化したときに呼び出す関数を登録するために使用できます。

モジュールと HAL モジュールをインポートする必要があります。

```
import hal_glib
import hal
```

次に、ピンを作成し、値が変更された（ウォッチャー）信号を関数呼び出しに接続します。

```
class HandlerClass:
def __init__(self, halcomp,builder,useropts):
self.example_trigger = hal_glib.GPin(halcomp.newpin('example-trigger', hal.HAL_BIT, -
hal.HAL_IN))
self.example_trigger.connect('value-changed', self._on_example_trigger_change)
```

そして、呼び出される関数があります：

```
def _on_example_trigger_change(self,pin,userdata=None):
print "pin value changed to:" % (pin.get())
print "pin name= %s" % (pin.get_name())
print "pin type= %d" % (pin.get_type())

# this can be called outside the function
```

```
self.example_trigger.get()
```

12.14.10 QTVCP ハンドラーで HAL_GLIB とともに使用する

QTVCP は HAL_GLIB ライブラリを使用します。このライブラリは、HAL 入力ピンに「ウォッチャー」信号を接続するために使用できます。 この信号は、HAL ピンの状態が変化したときに呼び出す関数を登録するために使用できます。

HAL モジュールをインポートする必要があります。

```
import hal
```

次に、ピンを作成し、VALUE_CHANGED（ウォッチャー）シグナルを関数呼び出しに接続します。

```
########################
# **** INITIALIZE **** #
########################
# widgets allows access to widgets from the qtvcp files
# at this point the widgets and hal pins are not instantiated
def __init__(self, halcomp,widgets,paths):
self.hal = halcomp
self.testPin = self.hal.newpin('test-pin', hal.HAL_BIT, hal.HAL_IN)
self.testPin.value_changed.connect(lambda s: self.setTestPin(s))
```

そして、呼び出される関数があります。

これは、ピンの値と情報を取得する方法を示しています。

```
####################
# general functions #
####################
def setTestPin(self, data):
print "Test pin value changed to:" % (data)
print 'halpin object =', self.w.sender()
print 'Halpin name: ',self.sender().text()
print 'Halpin type: ',self.sender().get_type()

# this can be called outside the function
print self.testPin.get()
```

12.14.11 プロジェクトのアイデア

- ボタン、スイッチ、およびインジケーターを備えた外部コントロールパネルを作成します。すべてをマイクロコントローラーに接続し、シリアルインターフェースを使用してマイクロコントローラーを PC に接続します。 PYTHON には、PYSERIAL と呼ばれる非常に有能なシリア

ルインターフェイスモジュールがあります（ユニバースリポジトリにあるUBUNTUパッケージ名「PYTHON-SERIAL」）

- LCDPROC互換のLCDモジュールを接続し、それを使用して、選択した情報を含むデジタル表示を表示します（UBUNTUパッケージ名「LCDPROC」、ユニバースリポジトリ内）
- PYTHONでサポートされているGUIライブラリ（GTK、QT、WXWINDOWSなど）を使用して仮想コントロールパネルを作成します

13章 Kinematics

13.1 前書き

CNC マシンについて話すとき、私たちは通常、特定の場所に移動してさまざまなタスクを実行するように命令されたマシンについて考えます。 機械空間の統一されたビューを持ち、それを 3D 空間上の人間の視点に合わせるために、ほとんどの機械（すべてではないにしても）はデカルト座標系と呼ばれる共通の座標系を使用します。

デカルト座標系は、それぞれが他の 2 つに垂直な 3 つの軸（X、Y、Z）で構成されています。 1

G コードプログラム（RS274 / NGC）について話すとき、パラメータ（X- Y- Z-）として位置を持ついくつかのコマンド（G0、G1 など）について話します。 これらの位置は、デカルト位置を正確に参照しています。 LINUXCNC モーションコントローラーの一部は、これらの位置を機械の運動学に対応する位置に変換する役割を果たします。 2

13.1.1 ジョイントと軸

CNC マシンのジョイントは、マシンの物理的な自由度の 1 つです。 これは、線形（親ねじ）または回転（回転テーブル、ロボットアームジョイント）の場合があります。 特定のマシンには、任意の数のジョイントが存在する可能性があります。 たとえば、1 つの人気のあるロボットには 6 つのジョイントがあり、一般的な単純なフライス盤には 3 つしかありません。

ジョイントがキネマティクス軸に一致するように配置されている特定のマシン（軸 X に沿ったジョイント 0、軸 Y に沿ったジョイント 1、軸 Z に沿ったジョイント 2）があり、これらのマシンはデカルトマシン（またはトリビアルキネマティクスを備えたマシン）と呼ばれます。 これらはフライス加工で使用される最も一般的な機械ですが、機械制御の他の領域（溶接：プーマタイプのロボットなど）ではあまり一般的ではありません。

LINUXCNC は、X Y Z A B C U VW という名前の軸をサポートします。XYZ 軸は通常、通常のデカルト座標を参照します。 A B C 軸は、それぞれ X YZ 軸を中心とした回転座標を指します。 U VWAX は、それぞれ X YZ 軸に対して一般的に同一直線上にある追加の座標を参照します。

NOTE

1「軸」という言葉は、CNC 機械について話したり、機械の移動方向を指すときにも一般的に（そして間違って）使用されます。

2 キネマティクス：デカルト空間から関節空間に変換するための双方向関数

13.2 トリビアルキネマティクス

最も単純なマシンは、各ジョイントがデカルト軸の 1 つに沿って配置されるマシンです。 これらのマシンでは、デカルト空間（G コードプログラム）から関節空間（マシンの実際のアクチュエーター）へのマッピングは簡単です。 これは単純な 1：1 マッピングです。

```
pos->tran.x = joints[0];
pos->tran.y = joints[1];
pos->tran.z = joints[2];
```

上記のコードスニペットでは、マッピングがどのように行われるかを確認できます。X 位置はジョイント 0 と同じであり、Y 位置はジョイント 1 と同じです。上記は直接運動学（変換の一方向）を示しています。 次のコードスニペットは、逆運動学（または変換の逆方向）を参照しています。

```
joints[0] = pos->tran.x;
joints[1] = pos->tran.y;
joints[2] = pos->tran.z;
```

LINUXCNC では、アイデンティティキネマティクスは TRIVKINS キネマティクスモジュールで実装され、9 軸に拡張されます。 軸座標とジョイント番号の間のデフォルトの関係は次のとおりです。

```
pos->tran.x = joints[0];
pos->tran.y = joints[1];
pos->tran.z = joints[2];
pos->a = joints[3];
pos->b = joints[4];
pos->c = joints[5];
pos->u = joints[6];
pos->v = joints[7];
pos->w = joints[8];
```

同様に、トリブキンの逆運動学のデフォルトの関係は次のとおりです。

```
joints[0] = pos->tran.x;
joints[1] = pos->tran.y;
joints[2] = pos->tran.z;
joints[3] = pos->a;
joints[4] = pos->b;
joints[5] = pos->c;
joints[6] = pos->u;
joints[7] = pos->v;
joints[8] = pos->w;
```

使用される軸文字に省略がない場合は、些細な「キン」（トリブキンキネマティクス）またはデカルトマシンの変換を行うのは簡単です。

マシンに1つ以上の軸文字がない場合は、少し複雑になります。 軸文字が省略される問題は、TRIVKINS モジュールで COORDINATES = MODULE パラメーターを使用することで解決されます。 指定された各座標には、ジョイント番号が連続して割り当てられます。 旋盤は COORDINATES = XZ で記述できます。ジョイントの割り当ては次のようになります。

```
joints[0] = pos->tran.x
joints[1] = pos->tran.z
```

軸文字を省略した構成では、COORDINATES =パラメーターの使用をお勧めします。 3

TRIVKINS キネマティクスモジュールでは、複数の関節に同じ座標を指定することもできます。 この機能は、Y 座標に 2 つの独立したモーターを備えたガントリーのようなマシンで役立ちます。 このようなマシンは COORDINATES = XYYZ を使用して、ジョイントの割り当てを行うことができます。

NOTE

3 歴史的に、TRIVKINS モジュールは COORDINATES =パラメータをサポートしていなかったため、旋盤の構成は XYZ マシンとして構成されることがよくありました。 未使用の Y 軸は、1）すぐにホームに戻る、2）単純なループバックを使用して、位置コマンドの HAL ピンを位置フィードバックの HAL ピンに接続する、3）GUI ディスプレイに非表示にするように構成されました。 多くの SIM 設定は、共通の HAL ファイルを共有するためにこれらのメソッドを使用します。

```
joints[0] = pos->tran.x
joints[1] = pos->tran.y
joints[2] = pos->tran.y
joints[3] = pos->tran.z
```

詳細については、TRIVKINS の MAN ページを参照してください。

13.3 重要な運動学

マシンのセットアップにはかなりの種類があります（ロボット：プーマ、スカラ、ヘキサポッドなど）。 それらのそれぞれは、線形および回転ジョイントを使用してセットアップされます。 これらの関節は通常デカルト座標と一致しないため、変換を行うキネマティクス関数が必要です（実際には 2 つの関数：フォワードキネマティクス関数とインバースキネマティクス関数）。

上記を説明するために、バイポッド（三脚の簡略版であるヘキサポッドの簡略版）と呼ばれる単純な運動学を分析します。

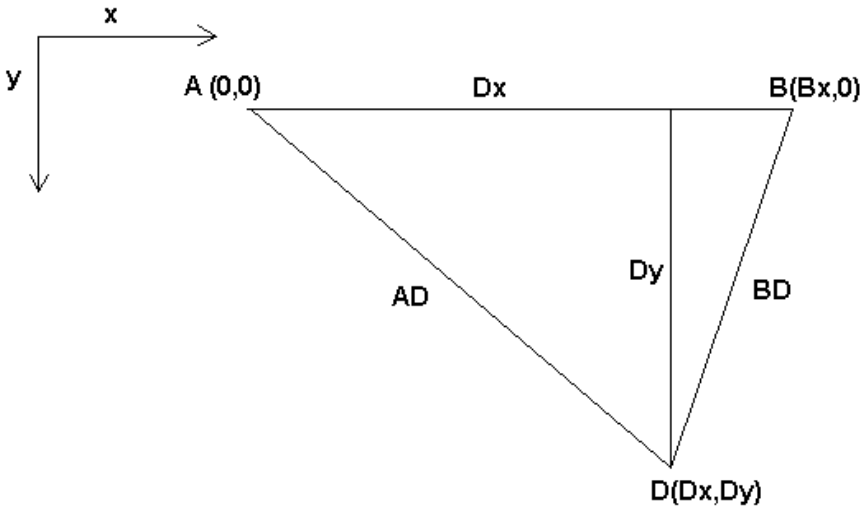


図 15-124

私たちが話しているバイポッドは、壁に配置された 2 つのモーターで構成されるデバイスであり、そこからデバイスがワイヤーを使用して吊り下げられます。 この場合のジョイントは、モーターからデバイスまでの距離です（図では AD および BD と呼ばれています）。

モーターの位置は慣例により固定されています。 モーター A は（0,0）にあり、X 座標は 0 であり、Y 座標も 0 です。モーター B は（Bx、0）に配置され、X 座標は Bx です。

ツールチップは、距離 AD と BD、およびデカルト座標 Dx、Dy によって定義される点 D にあります。キネマティクスの仕事は、関節の長さ（AD、BD）からデカルト座標（Dx、Dy）に、またはその逆に変換することです。

13.3.1 フォワードトランスフォーメーション

ジョイント空間からデカルト空間に変換するには、いくつかの三角法の規則（点（0,0）、（Dx、0）、（Dx、Dy）と三角形（Dx、0）、（Bx によって決定される直角三角形）を使用します。 、0）および（Dx、Dy）。

私たちはそれを簡単に見ることができます：

$$AD^2 = x^2 + y^2 \; BD^2 = (Bx - x)^2 + y^2, \text{ likewise: } BD^2 = (Bx - x)^2 + y^2$$

簡単にわかります。一方を他方から引くと、次のようになります。

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

したがって：

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

そこから計算します：

$$y = \sqrt{AD2 - x2}$$

Y の計算には差の平方根が含まれるため、実数にならない場合があることに注意してください。この関節位置に単一のデカルト座標がない場合、その位置は特異点であると言われます。 この場合、順運動学は-1 を返します。

実際のコードに翻訳：

```
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;
```

13.3.2 逆変換

この例では、逆運動学を直接記述できるため、はるかに簡単です。

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

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

または実際のコードに翻訳：

```
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;
```

13.4 実装の詳細

キネマティクスモジュールは HAL コンポーネントとして実装されており、ピンとパラメータのエクスポートが許可されています。 これは、（HAL 関数とは対照的に）いくつかの「C」関数で構成されています。

```
int kinematicsForward(const double *joint, EmcPose *world,
const KINEMATICS_FORWARD_FLAGS *fflags,
KINEMATICS_INVERSE_FLAGS *iflags)
```

順運動学機能を実装します。

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

逆運動学機能を実装します。

```
KINEMATICS_TYPE kinematicsType(void)
```

キネマティクスタイプ識別子（通常は KINEMATICS_BOTH）を返します。

```
int kinematicsHome(EmcPose *world, double *joint,
KINEMATICS_FORWARD_FLAGS *fflags,
KINEMATICS_INVERSE_FLAGS *iflags)
```

ホームキネマティクス関数は、すべての引数を既知のホーム位置の適切な値に設定します。 呼び出されるとき、これらは、既知の場合、たとえば INI ファイルからの初期値に設定する必要があります。ホームキネマティクスが任意の開始点を受け入れることができる場合は、これらの初期値を使用する必要があります。

```
int rtapi_app_main(void)
void rtapi_app_exit(void)
```

これらは、RTAPI モジュールの標準のセットアップおよび分解機能です。

それらが単一のソースファイルに含まれている場合、キネマティクスモジュールは HALCOMPILE によってコンパイルおよびインストールできます。 詳細については、HALCOMPILE（1）のマンページまたは HAL のマニュアルを参照してください。

14章 「変更された」DENAVIT-HARTENBERG（DH）の設定 のパラメータ

14.1 Prelude

LINUXCNC は、DENAVIT-HARTENBERG パラメータを介して一般的に指定されるシリアルキネマティクスの一般化されたセットをサポートするものを含む、多くのキネマティクスモジュールをサポートします。

このドキュメントでは、GENSERKINS キネマティクスを使用して LINUXCNC で MITSUBISHIRV-6SDL の DH パラメータを設定する方法について説明します。

NOTE

このドキュメントでは、VISMACH モデルの作成については説明していません。これは確かに非常に便利ですが、このドキュメントで導出された GENSERKINS モデルと一致させるには、同様に注意深いモデリングが必要です。

Note

エラーや欠点がある可能性があります。自己責任で使用してください。

14.2 一般

産業用ロボットの急増に伴い、LINUXCNC で中古ロボットを制御することへの関心が高まっています。産業や製造で使用される一般的なタイプのロボットは、剛体リンクで接続された一連の電動ジョイントとして設計された「シリアルマニピュレータ」です。 シリアルロボットには、多くの場合、空間内のオブジェクトの位置（XYZ）と方向（ABC またはピッチ、ロール、ヨー）の両方に必要な 6 つの自由度に必要な 6 つのジョイントがあります。 多くの場合、これらのロボットは、ベースからエンドエフェクタまで伸びるアーム構造を持っています。

このようなシリアルロボットの制御には、関節角度がわかっている場合の参照座標系に対するエンドエフェクタの位置と方向の計算（順運動学）と、特定の端に必要な関節角度のより複雑な逆計算が必要です。 -参照座標系に対するエフェクターの位置と方向（逆運動学）。 これらの計算に使用される標準の数学ツールは、基本的にパラメータと数式のテーブルである行列であり、順運動学および逆運動学の計算に含まれる回転と平行移動の処理を容易にします。

LINUXCNC は、一般的なシリアルロボットの順方向および逆方向の運動学を計算するために、GENSERKINS と呼ばれるアルゴリズムを実装する運動学モジュールを提供するため、シリアルロボットには数学の詳細な知識は必要ありません。 特定のシリアルロボットを制御するには、ロボットの機械的構造の数学的モデルを構築して計算を実行できるように、ゲンサーキンにデータを提供する必要があります。

必要なデータは、50 年代に JACQUESDENAVIT と RICHARDHARTENBERG によって導入され、DH パラメータと呼ばれる標準化された形式である必要があります。 DENAVIT と HARTENBERG は、4 つのパラメーターを使用して、1 つの関節が次の関節にどのようにリンクされているかを説明しました。 これらのパラメーターは、基本的に 2 つの回転（アルファとシータ）と 2 つの平行移動（A と D）を表します。

14.3 変更された DH パラメータ

よくあることですが、この「標準」は「変更された DH パラメータ」を導入した他の著者によって変更されており、GENSERKINS は出版物「INTRODUCTIONTO ROBOTICS、 ジョン J.クレイグによる「力学と制御」。 DH パラメータには多くの情報がありますが、実際に使用される規則を作成者が定義することはめったにないことに注意してください。 さらに、A という名前のパラメーターを R に変更する必要があることに気づき、混乱を招きました。 このドキュメントは、CRAIG による上記の出版物の規則に準拠していますが、ゲンサーキンとその HAL ピンとの一貫性を保つために、ジョイントとパラメーターの列挙は数字の 0 から始まるという違いがあります。

標準および変更された DH パラメータは、各関節の 4 つの数値（A、D、ALPHA、および THETA）で構成され、1 つの関節にある座標系（CS）を移動および回転して、次の関節に合わせる方法を記述します。 整列とは、CS の Z 軸が関節の回転軸と一致し、正の方向を指すことを意味します。右手の法則を使用して、親指を Z 軸の正の方向に向けると、指が指します。 関節の正の回転方向。 これを行うには、パラメータの導出を開始する前に、すべてのジョイントの正の方向を決定する必要があることが明らかになります。

「標準」表記と「変更済み」表記の違いは、パラメータがリンクに割り当てられる方法にあります。 GENSERKINS で「標準」の DH パラメータを使用すると、正しい数学的モデルが得られません。

14.4 GENSERKINS で使用されるように変更された DH パラメータ

GENSERKINS はシータ値へのオフセットを処理しないことに注意してください。シータは LINUXCNC によって制御される結合変数です。 CS がジョイントと位置合わせされている場合、その Z 軸を中心とした回転は、LINUXCNC によってそのジョイントに命令された回転と同じです。 これにより、ロボットの関節の 0°位置を任意に定義することができなくなります。

構成可能な3つのパラメーターは次のとおりです。

1. アルファ：「現在の座標系」のX軸を中心とした正または負の回転（ラジアン）
2. A：システムのINIファイルで定義されたマシン単位（MMまたはインチ）で指定された2つのジョイント軸間のXに沿った正の距離。
3. D：Zに沿った正または負の長さ（マシン単位でも）

パラメータセットは常に同じ順序で導出され、Dパラメータを設定することでセットが完成します。これにより、CSのZ軸が次のジョイントと整列したままになりません。 これは紛らわしいように思われるかもしれませんが、このルールに固執すると、パラメーターのワーキングセットが生成されます。Dパラメータを設定したら、CSのX軸が次のジョイントの軸を指す必要があります。

14.5 ジョイントとパラメータの番号付け

LINUXCNCの最初のジョイントはジョイント0です（ソフトウェアのカウントは0で始まるため）が、ほとんどの出版物は番号1で始まります。これはすべてのパラメーターにも当てはまります。 つまり、番号付けはA-0、ALPHA-0、D-0で始まり、A-5、ALPHA-5、およびD-5で終わります。 パブリケーションに従ってGENSERKINSパラメータを設定する場合は、このことに注意してください。

14.6 開始方法

慣例では、参照CSをロボットのベースに配置し、Z軸を最初の関節の軸と一致させ、X軸を次の関節の軸に向けることから始めます。

これにより、LINUXCNCのDRO値がそのポイントを参照するようになります。 そうすると、A-0とALPHA-0が0に設定されます。

上記の出版物（CRAIG）もD-0を0に設定しています。これは、ベースの下部に参照CSを配置するために変位オフセットが必要な場合に混乱を招きます。 D-0 =を変位に設定すると、正しい結果が得られます。 このように、パラメータの最初のセットはALPHA-0 = 0、A-0 = 0、D0 =変位であり、CSのX軸は次のジョイント（ジョイント-1）の軸を指します。

ネットセット（ALPHA-1、A-1、D-1）の導出は次のとおりです。常に、6番目のセット（ALPHA-5、A-5、D-5）まで同じシーケンスを使用します。

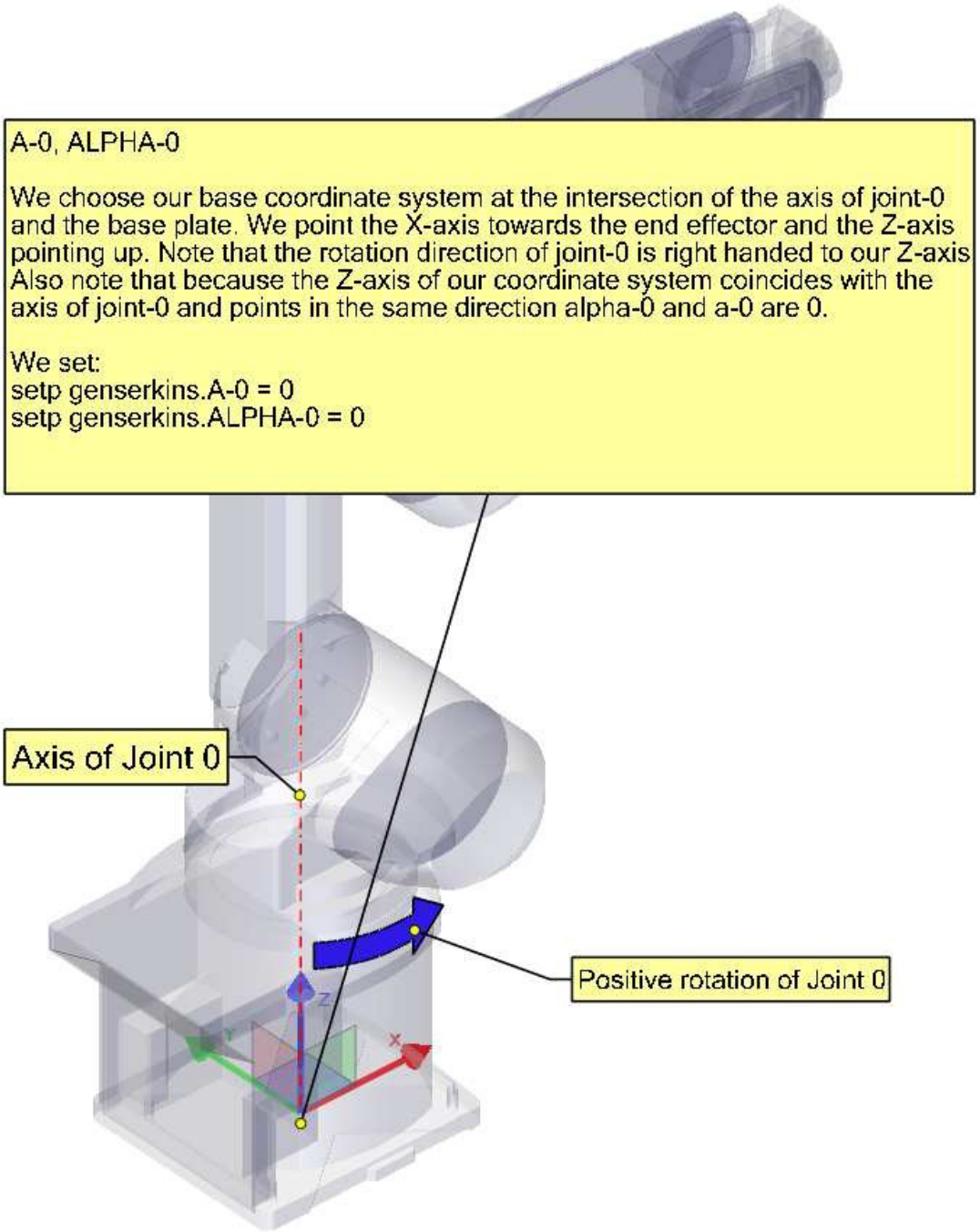
したがって、エンドエフェクタのTCP-CSはハンドフランジの中央にあります。

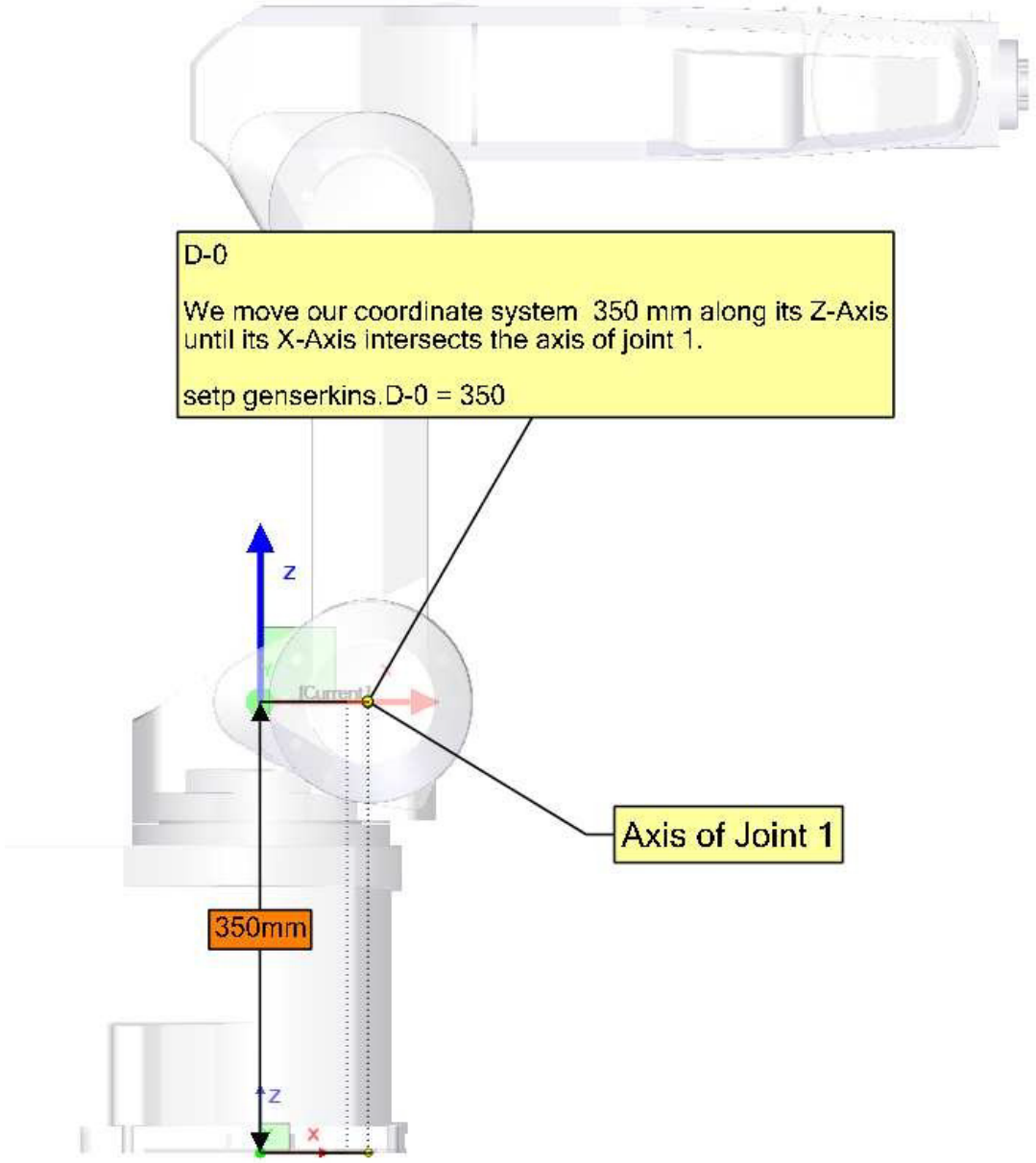
14.7 特殊なケース

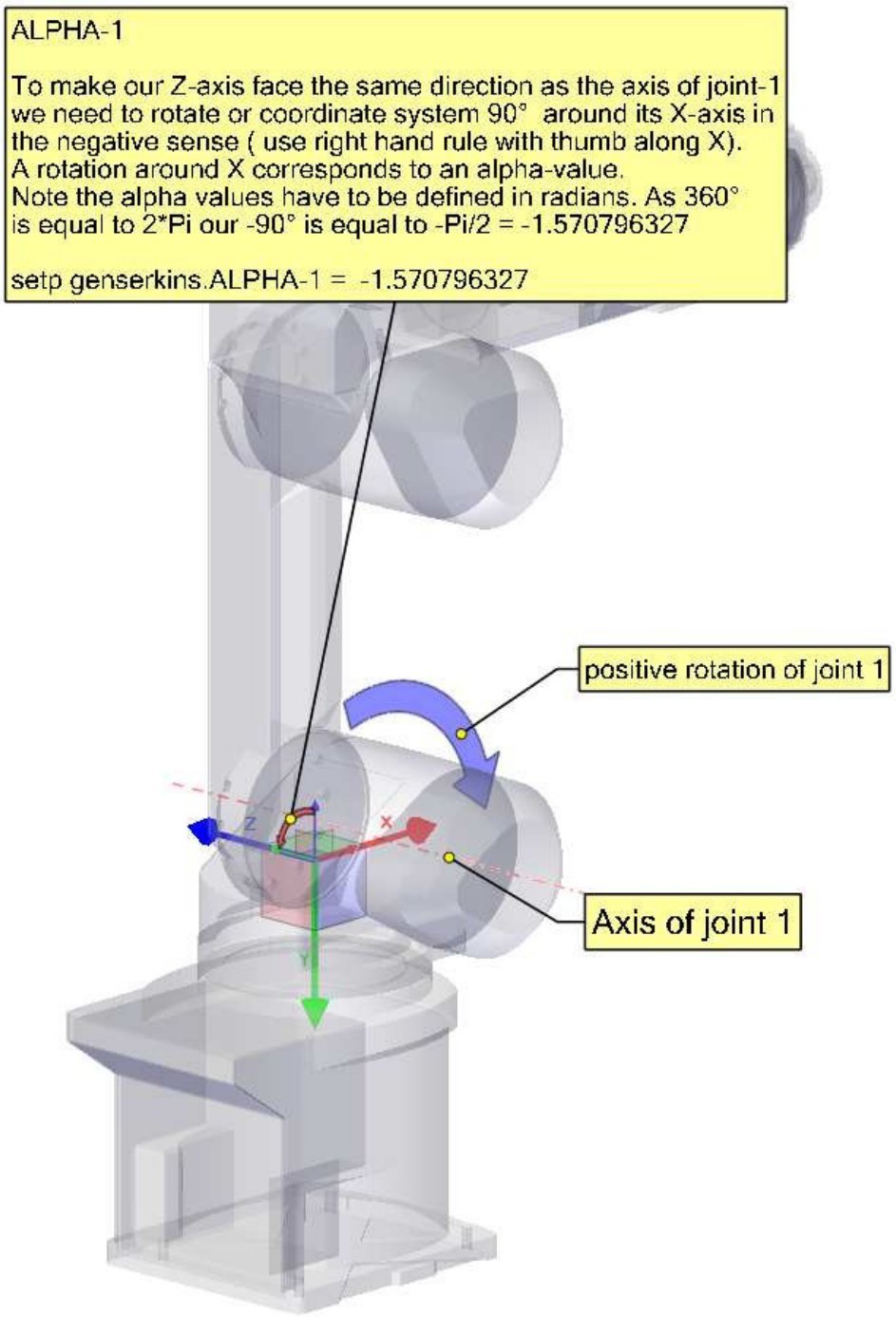
次の関節軸が最後の関節軸と平行である場合、Dパラメータの値を任意に選択できますが、0以外に設定しても意味がありません。

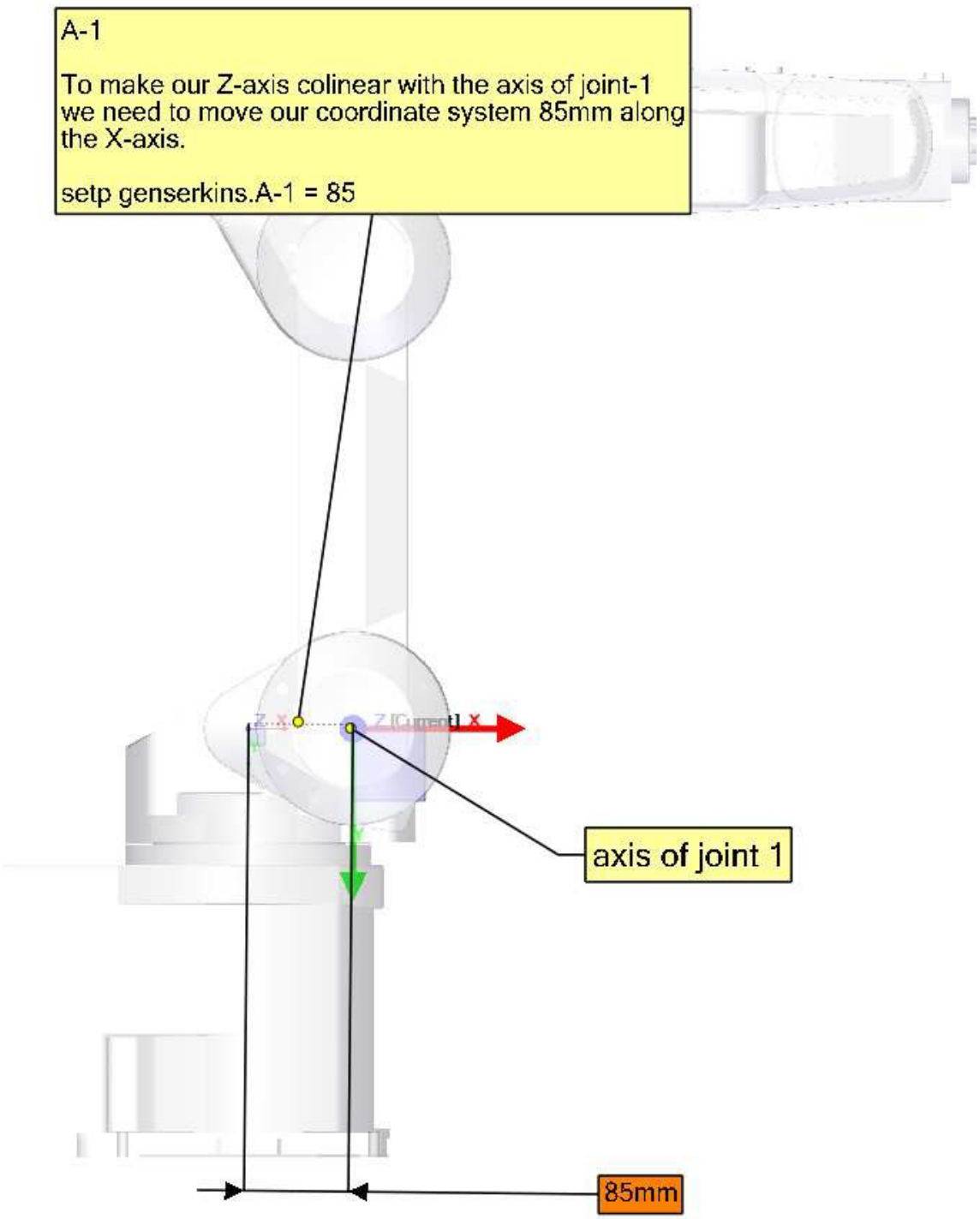
14.8 詳細例（RV-6SL）

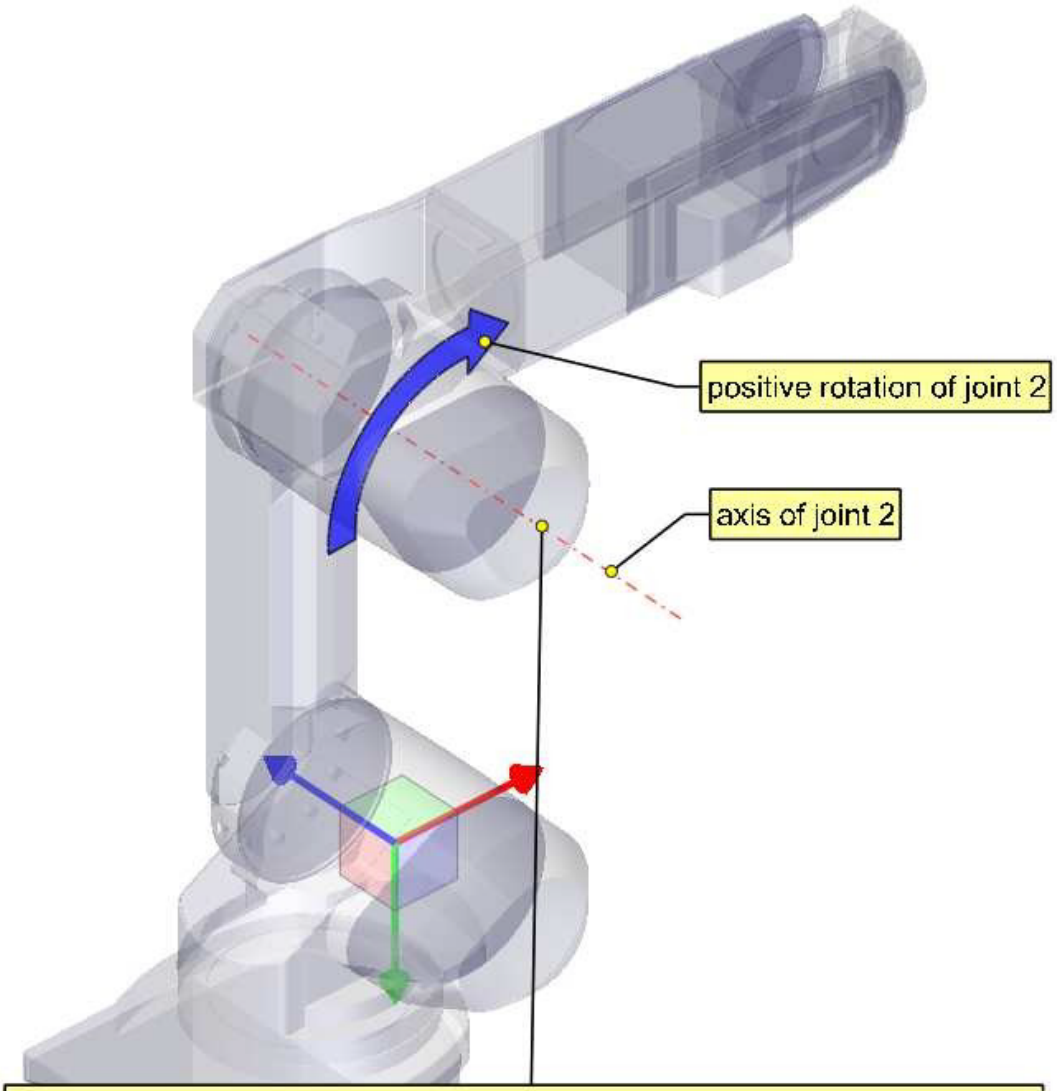
以下に説明するのは、三菱 RV-6SDL に必要な「変更された DH パラメータ」を導出する方法と、LINUXCNC のゲンサーキン運動学で使用される HAL ファイルのパラメータを設定する方法です。 必要な寸法は、ロボットの製造元から提供された寸法図から取得するのが最適です。



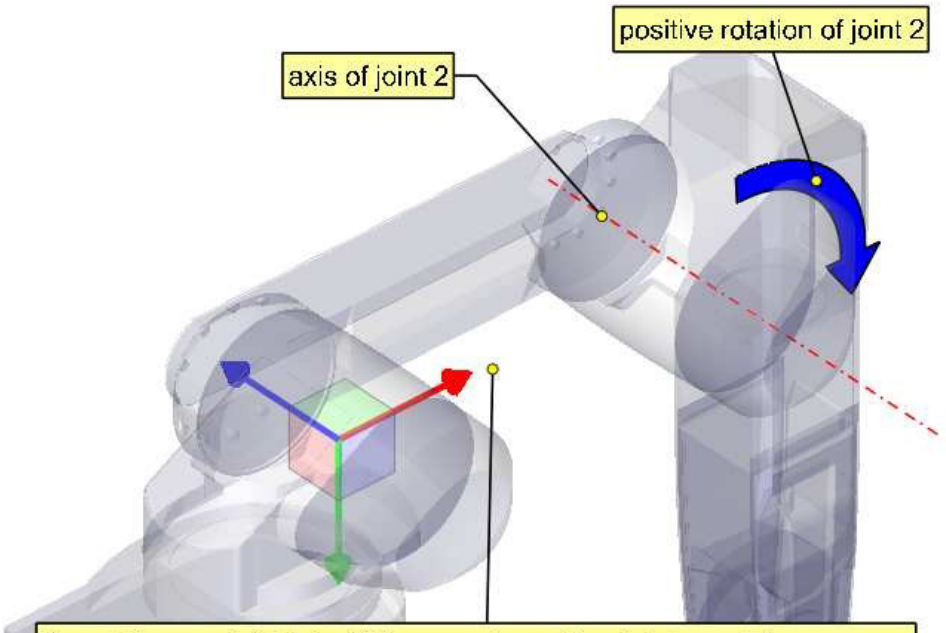

D-0
We move our coordinate system 350 mm along its Z-Axis until its X-Axis intersects the axis of joint 1.
setp genserkins.D-0 = 350
Z
Axis of Joint 1
350mm
Z
X


ALPHA-1
To make our Z-axis face the same direction as the axis of joint-1 we need to rotate or coordinate system 90° around its X-axis in the negative sense (use right hand rule with thumb along X).
A rotation around X corresponds to an alpha-value.
Note the alpha values have to be defined in radians. As 360° is equal to 2*Pi our -90° is equal to -Pi/2 = -1.570796327
setp genserkins.ALPHA-1 = -1.570796327
positive rotation of joint 1
Axis of joint 1
Z
X
Y


A-1
To make our Z-axis colinear with the axis of joint-1 we need to move our coordinate system 85mm along the X-axis.
setp genserkins.A-1 = 85
axis of joint 1
85mm



Note:
In order to make our X-axis intersect the axis of joint-2 we would need to rotate our coordinate system around its Z-axis. To do this we could, in theory, define theta-1 equal to -90°.
However genserkins does not allow the definition of theta values. In genserkins.c we see that the theta values for all joints are set to 0.
Now theta of course is the rotation of the joint itself and so is variable in an angular joint. Theta values are only used to define the home pose of a robot in the way of an offset.
So if we could define theta-1 equal to -90° we could define joint-1 to be oriented this way for 0°. Since we cannot define it we need to rotate joint-1 in a way so that our X-axis intersects with the axis of the next joint.



By rotating our joint-1 by 90° we made our X-axis intersect the axis of the next joint and we can continue defining our DH-Parameters.

D-1

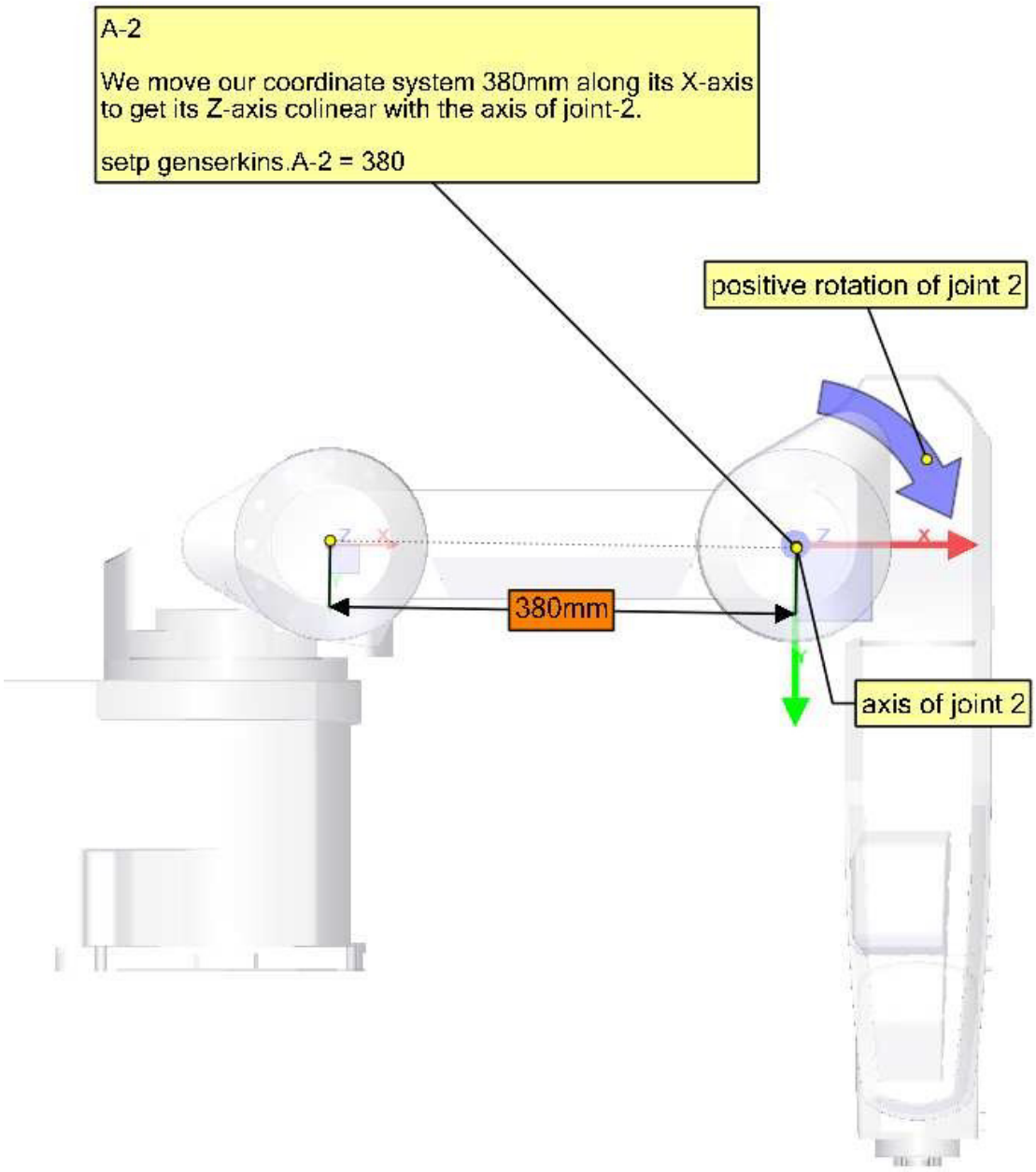
Since, after we rotated joint 1, our X-axis already intersects the axis of joint-2 we do not need to move our coordinate system along the Z-axis.

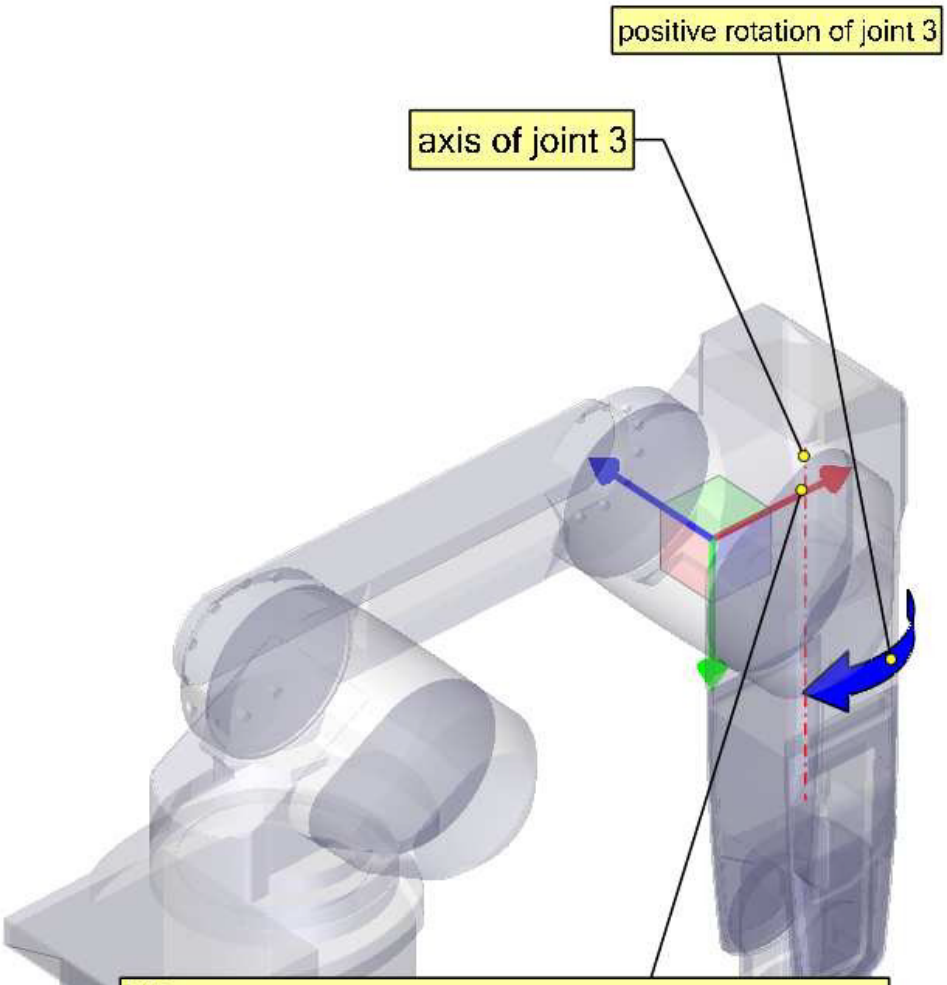
setp genserkins.D-1 = 0

ALPHA-2

The axis of joint -2 is parallel to the axis of joint -1 and points in the same direction. Thus we do not need to rotate our Z-axis.

setp genserkins.ALPHA-2 = 0


A-2
We move our coordinate system 380mm along its X-axis to get its Z-axis colinear with the axis of joint-2.
setp genserkins.A-2 = 380
positive rotation of joint 2
380mm
axis of joint 2



D-2

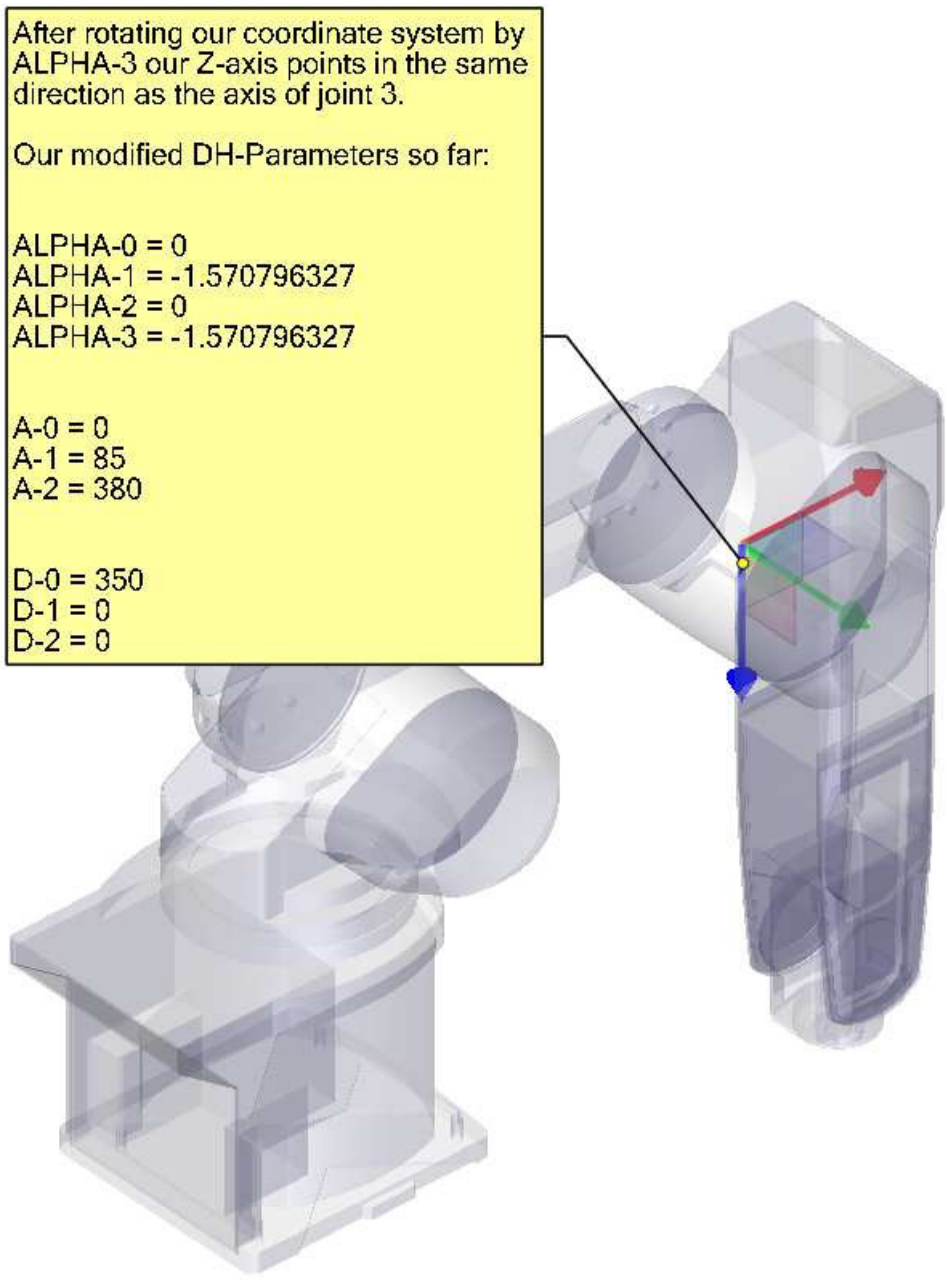
Our X-axis again already intersects the axis of the next joint 3.
So our d2 parameter is again 0.

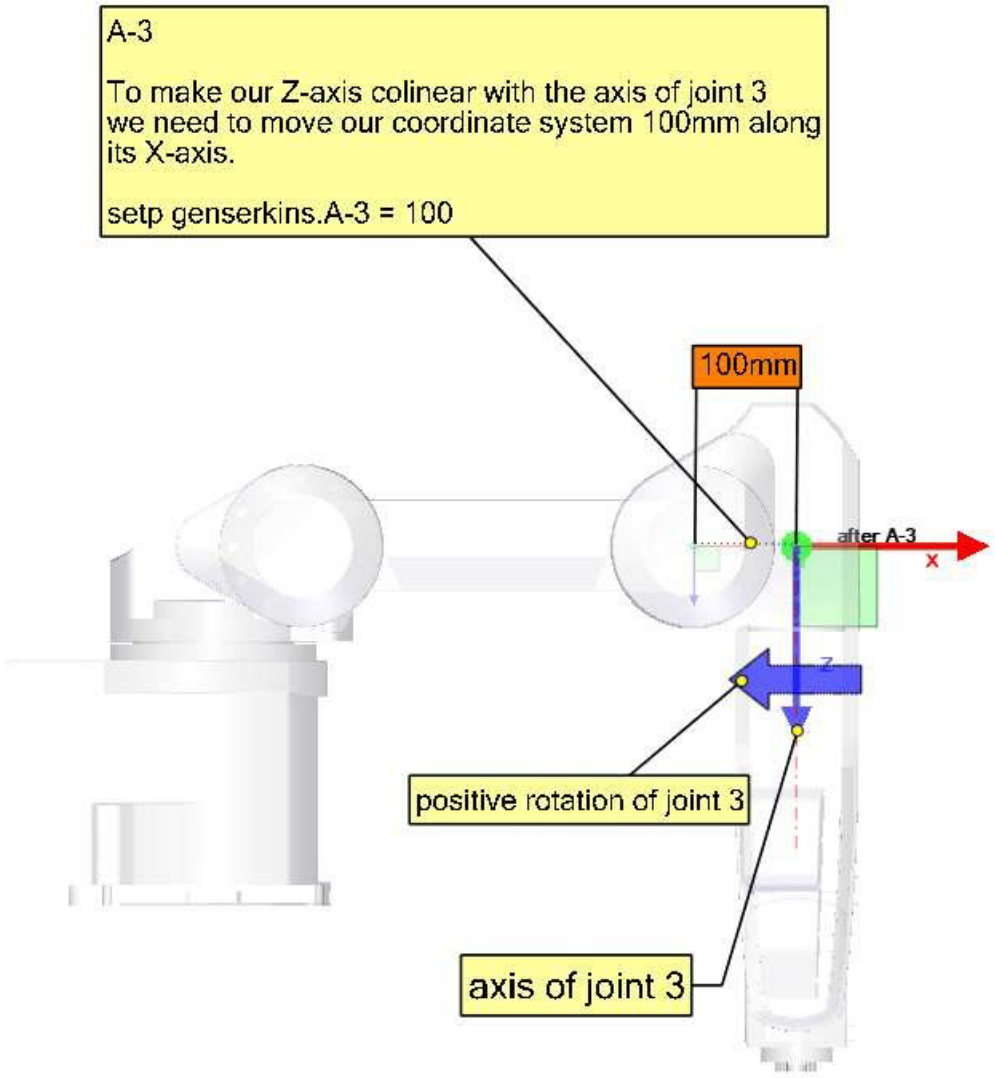
setp genserkins.D-2 = 0

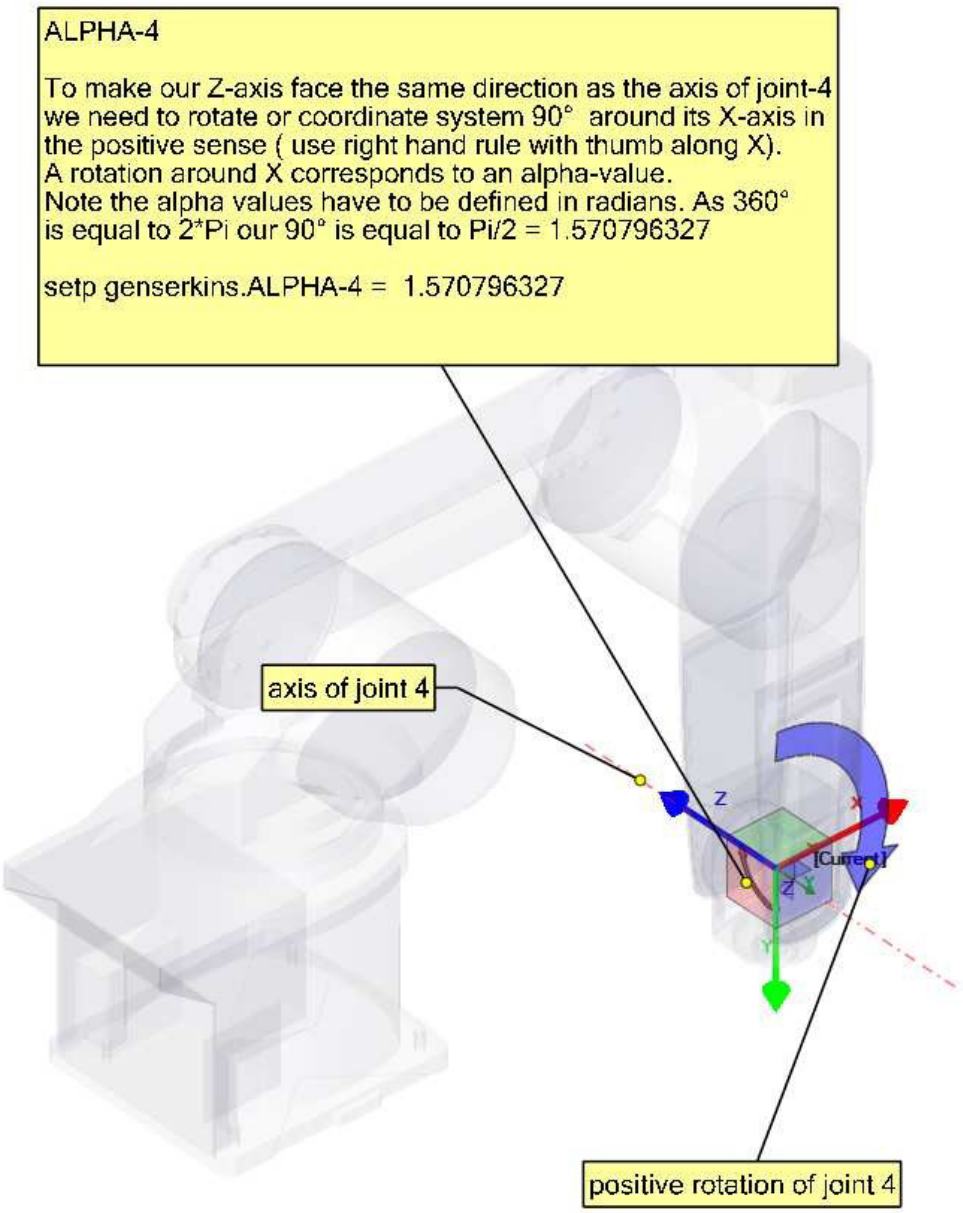
ALPHA-3

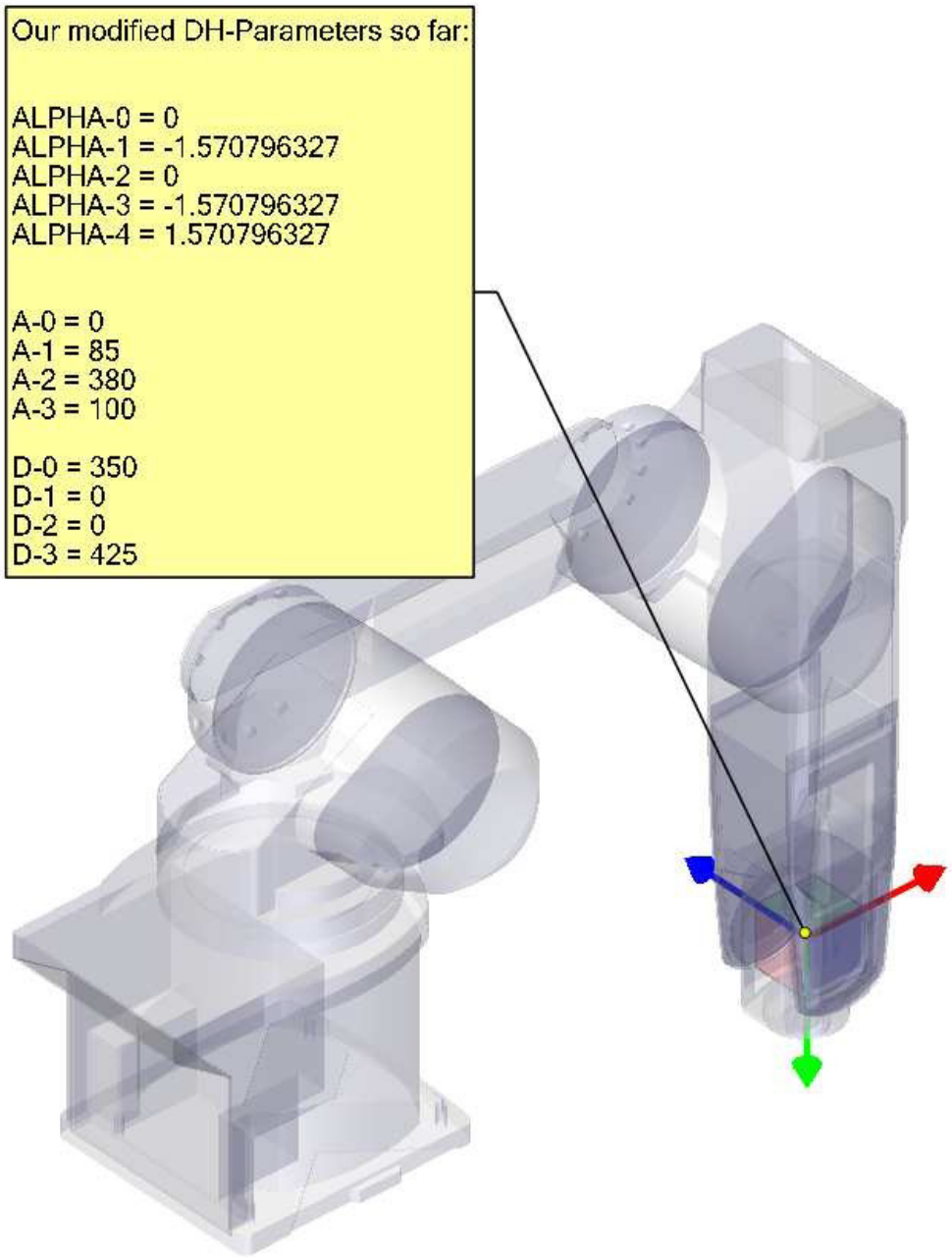
To make our Z-axis face the same direction as the axis of joint-3 we need to rotate or coordinate system 90° around its X-axis in the negative sense (use right hand rule with thumb along X).
A rotation around X corresponds to an alpha-value.
Note the alpha values have to be defined in radians. As 360° is equal to 2*Pi our -90° is equal to -Pi/2 = -1.570796327

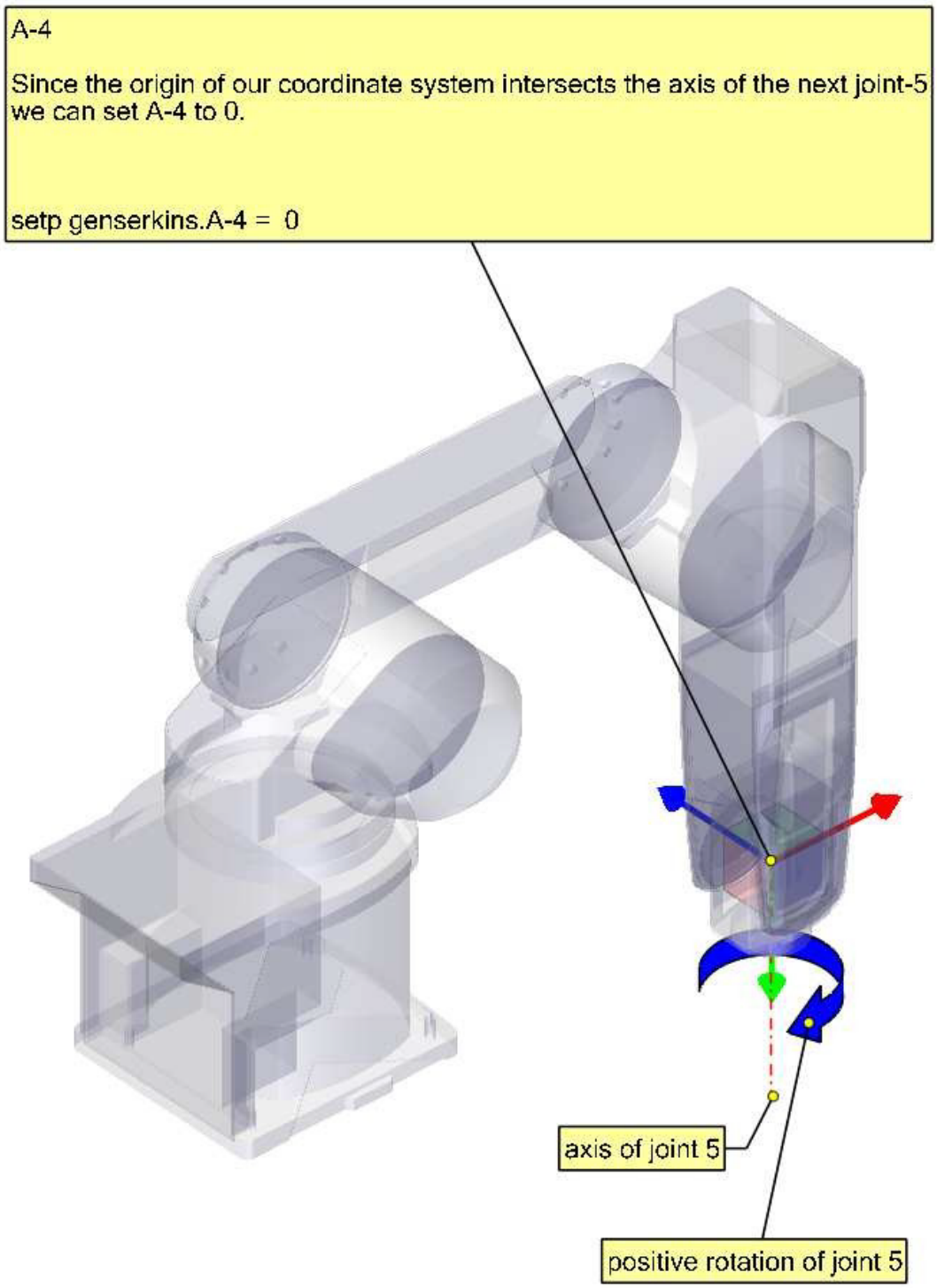
setp genserkins.ALPHA-3 = -1.570796327


After rotating our coordinate system by ALPHA-3 our Z-axis points in the same direction as the axis of joint 3.
Our modified DH-Parameters so far:
ALPHA-0 = 0
ALPHA-1 = -1.570796327
ALPHA-2 = 0
ALPHA-3 = -1.570796327
A-0 = 0
A-1 = 85
A-2 = 380
D-0 = 350
D-1 = 0
D-2 = 0


A-3
To make our Z-axis colinear with the axis of joint 3 we need to move our coordinate system 100mm along its X-axis.
setp genserkins.A-3 = 100
100mm
after A-3
X
Z
positive rotation of joint 3
axis of joint 3


ALPHA-4
To make our Z-axis face the same direction as the axis of joint-4 we need to rotate or coordinate system 90° around its X-axis in the positive sense (use right hand rule with thumb along X).
A rotation around X corresponds to an alpha-value.
Note the alpha values have to be defined in radians. As 360° is equal to 2*Pi our 90° is equal to Pi/2 = 1.570796327
setp genserkins.ALPHA-4 = 1.570796327
axis of joint 4
positive rotation of joint 4
Z
X
[Current]


Our modified DH-Parameters so far:
ALPHA-0 = 0
ALPHA-1 = -1.570796327
ALPHA-2 = 0
ALPHA-3 = -1.570796327
ALPHA-4 = 1.570796327
A-0 = 0
A-1 = 85
A-2 = 380
A-3 = 100
D-0 = 350
D-1 = 0
D-2 = 0
D-3 = 425


A-4
Since the origin of our coordinate system intersects the axis of the next joint-5 we can set A-4 to 0.
setp genserkins.A-4 = 0
axis of joint 5
positive rotation of joint 5

D-4

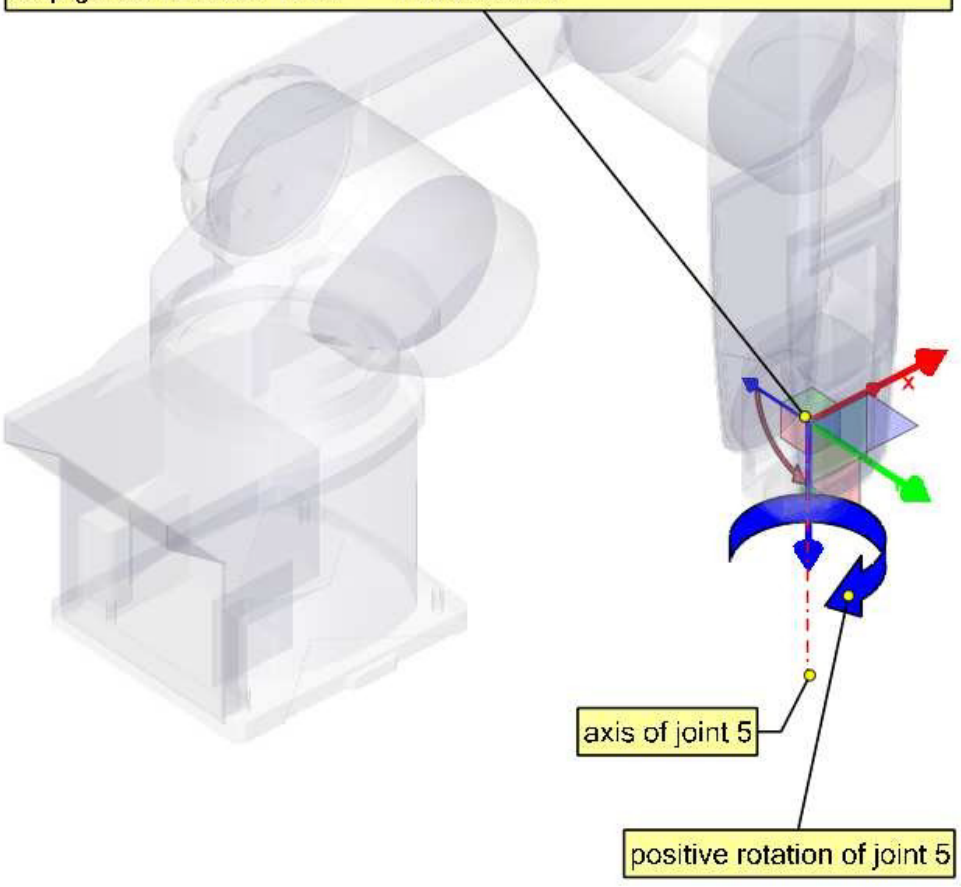
Since the origin of our coordinate system lies on the axis of the next joint our d4 parameter is also 0.

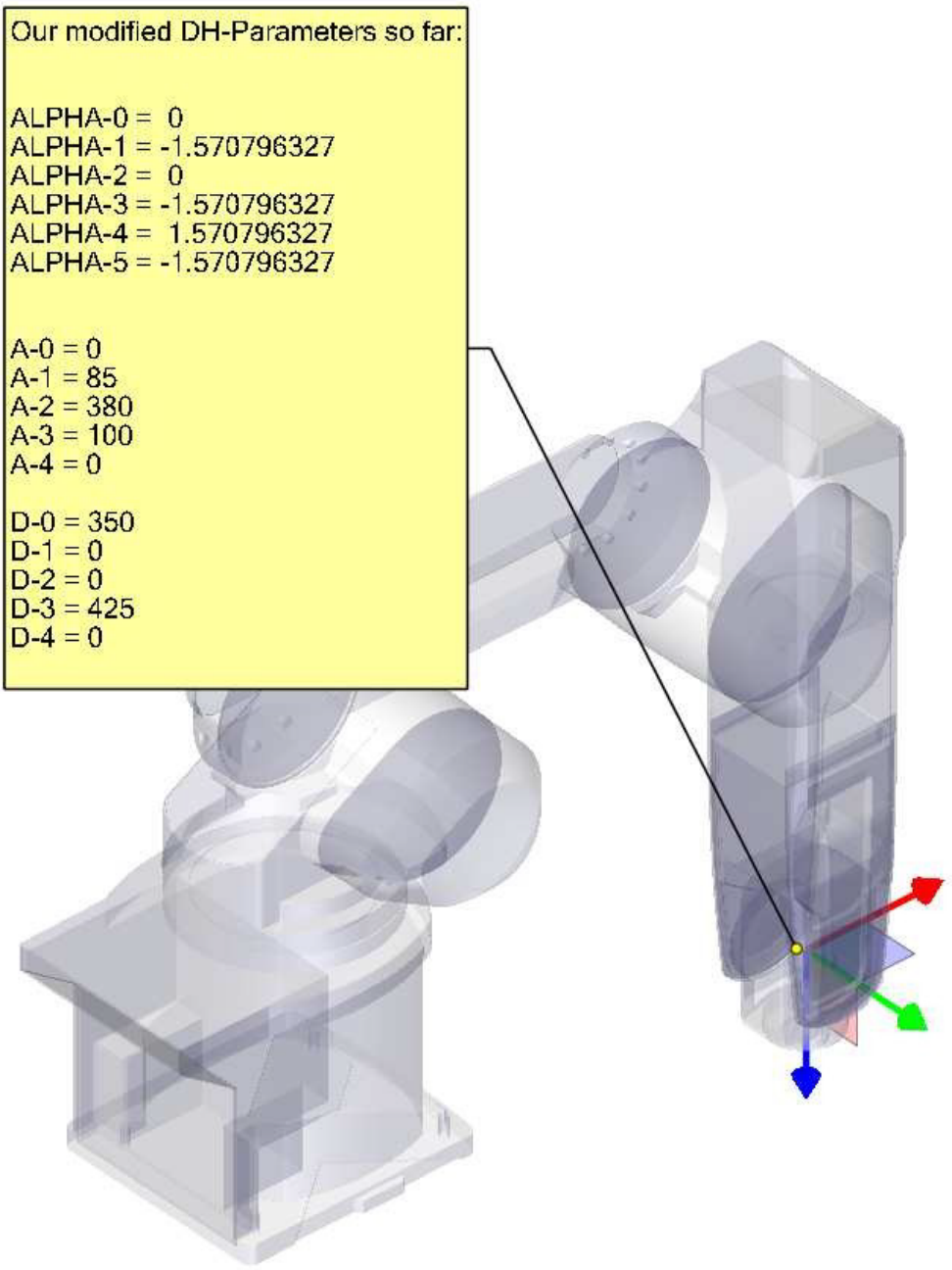
setp genserkins.D-4 = 0

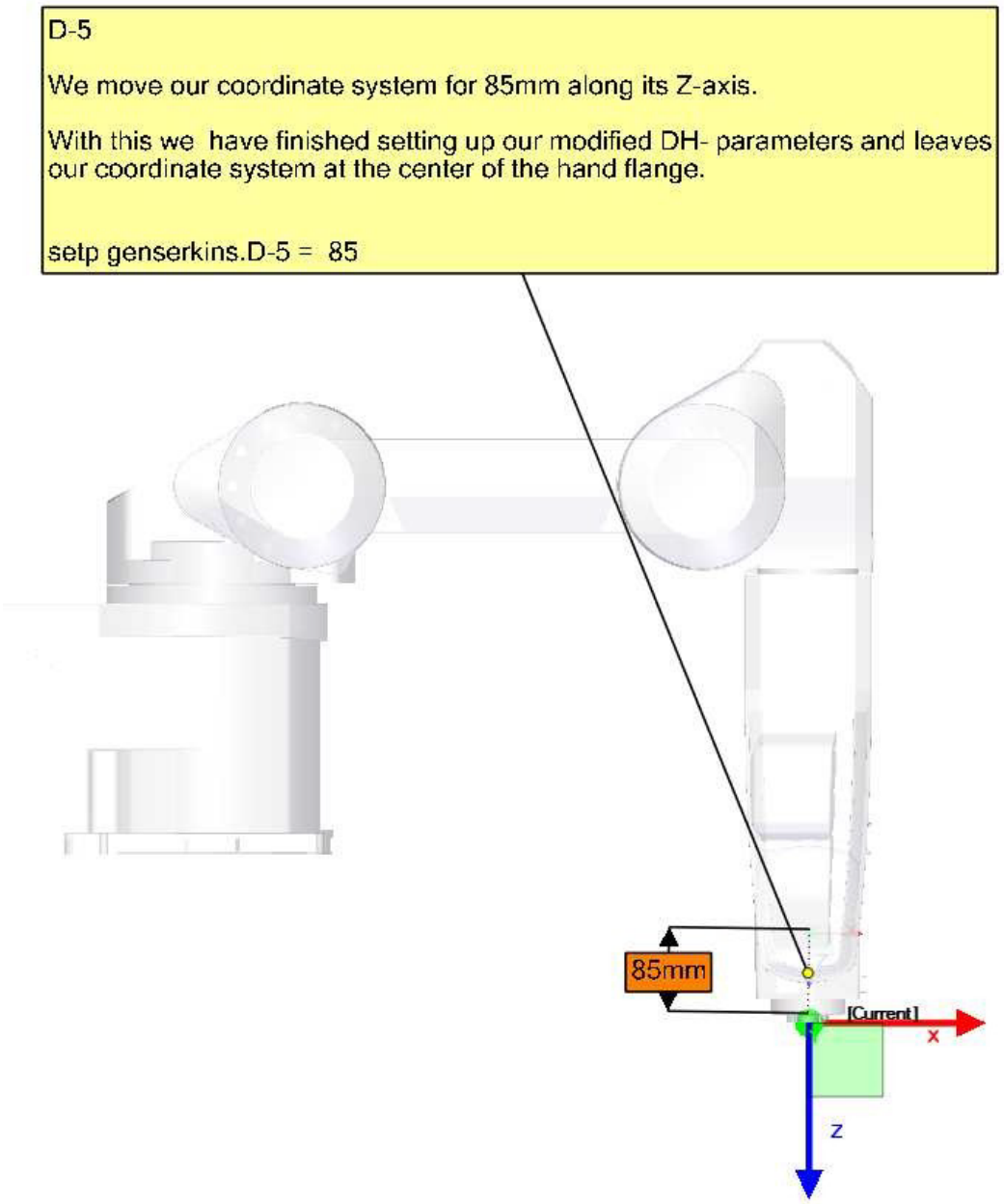
ALPHA-5

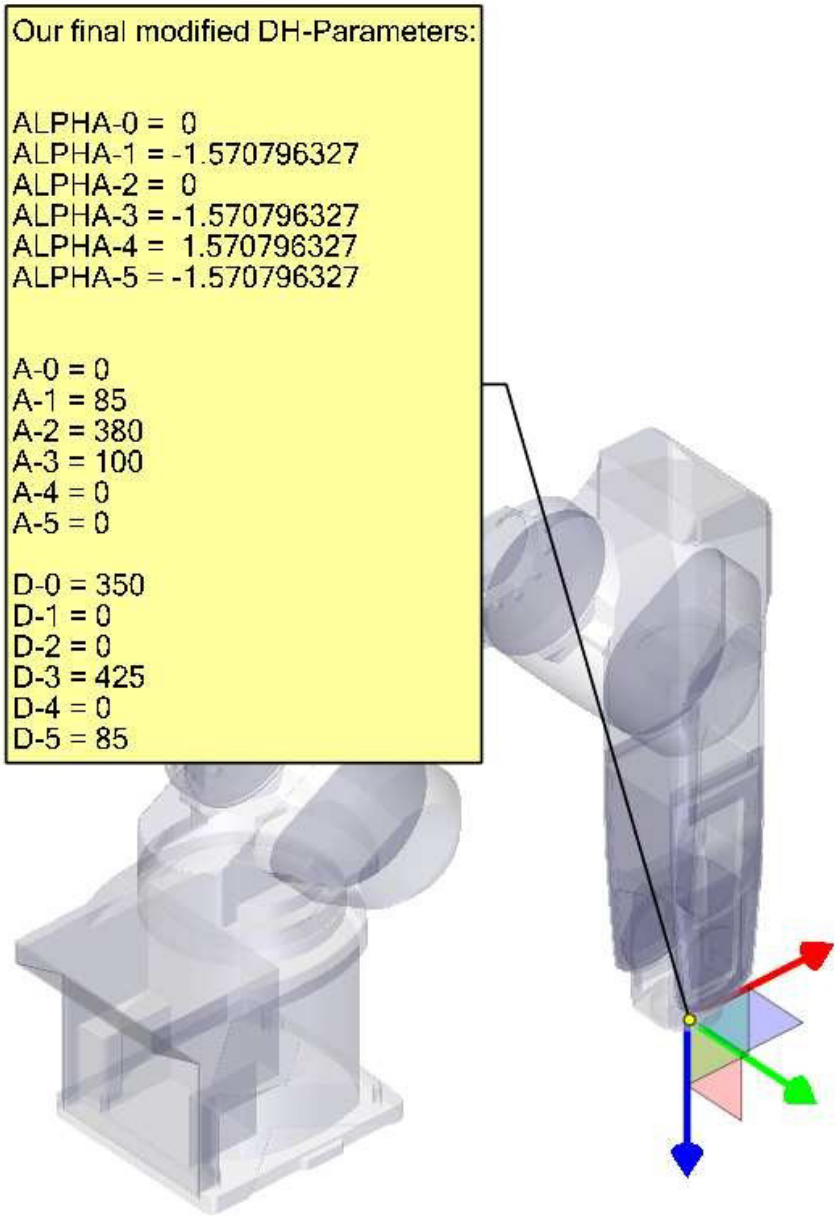
To make our Z-axis face the same direction as the axis of joint-5 we need to rotate or coordinate system 90° around its X-axis in the negative sense (use right hand rule with thumb along X).
A rotation around X corresponds to an alpha-value.
Note the alpha values have to be defined in radians. As 360° is equal to 2*Pi our -90° is equal to -Pi/2 = -1.570796327

setp genserkins.ALPHA-5 = -1.570796327




Our modified DH-Parameters so far:
ALPHA-0 = 0
ALPHA-1 = -1.570796327
ALPHA-2 = 0
ALPHA-3 = -1.570796327
ALPHA-4 = 1.570796327
ALPHA-5 = -1.570796327
A-0 = 0
A-1 = 85
A-2 = 380
A-3 = 100
A-4 = 0
D-0 = 350
D-1 = 0
D-2 = 0
D-3 = 425
D-4 = 0


D-5
We move our coordinate system for 85mm along its Z-axis.
With this we have finished setting up our modified DH- parameters and leaves our coordinate system at the center of the hand flange.
setp genserkins.D-5 = 85
85mm
[Current]
X
Z



14.9 クレジット

RV-6SL ロボットのすべてのテキストとグラフィックスを提供してくれたユーザー ACIERA に感謝します。

15章　5 軸 Kinematics

15.1　序章

LinuxCNC で制御される調整された多軸 CNC 工作機械は、機械のタイプごとに特別な運動学コンポーネントを必要とします。 この章では、最も一般的な 5 軸マシン構成のいくつかについて説明し、次に 2 種類のマシンの一般的な数学的プロセスで順方向（作業からジョイント座標へ）および逆方向（ジョイントから作業へ）変換を開発します。

キネマティクスコンポーネントと、コンピューター画面での動作を示すための vismach シミュレーションモデルが提供されています。 HAL ファイルデータの例も示されています。

15.2　5 軸工作機械の構成

このセクションでは、協調的な動きで制御される 5 つのジョイントまたは自由度を備えた一般的な 5 軸フライス盤またはルーターマシンを扱います。

3 軸工作機械は工具の向きを変えることができないため、5 軸工作機械は 2 つの余分な軸を使用して、自由形状の表面を効率的に加工するために切削工具を適切な向きに設定します。

典型的な 5 軸工作機械の構成を図 1 および 2 に示します。 図のセクションの 3、5、7、および 9-11 [1,2]。

5 軸工作機械の運動学は、6 軸シリアルアームロボットの運動学よりもはるかに単純です。これは、3 つの軸が通常は直線軸であり、2 つだけが回転軸であるためです。

15.3　ツールの向きと場所

CAD / CAM システムは通常、ワークピースの 3D CAD モデルと、CNC5 軸マシンに入力するための CAM データを生成するために使用されます。 工具またはカッターの位置（CL）データは、カッターの先端の位置とワークピースの座標系に対するカッターの向きで構成されます。 ほとんどの CAM システムで生成され、図 1 に示されている 2 つのベクトルには、次の情報が含まれています。

$$K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} \text{ orientation vector;} \qquad Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \text{ position vector} \qquad (1)$$

K ベクトルは 6 軸ロボット運動学[3]で使用されたポーズ行列 E6 の 3 番目のベクトルに相当し、Q ベクトルは E6 の 4 番目のベクトルに相当します。 たとえば MASTERCAM では、この情報は中間出力の「.nci」ファイルに含まれています。

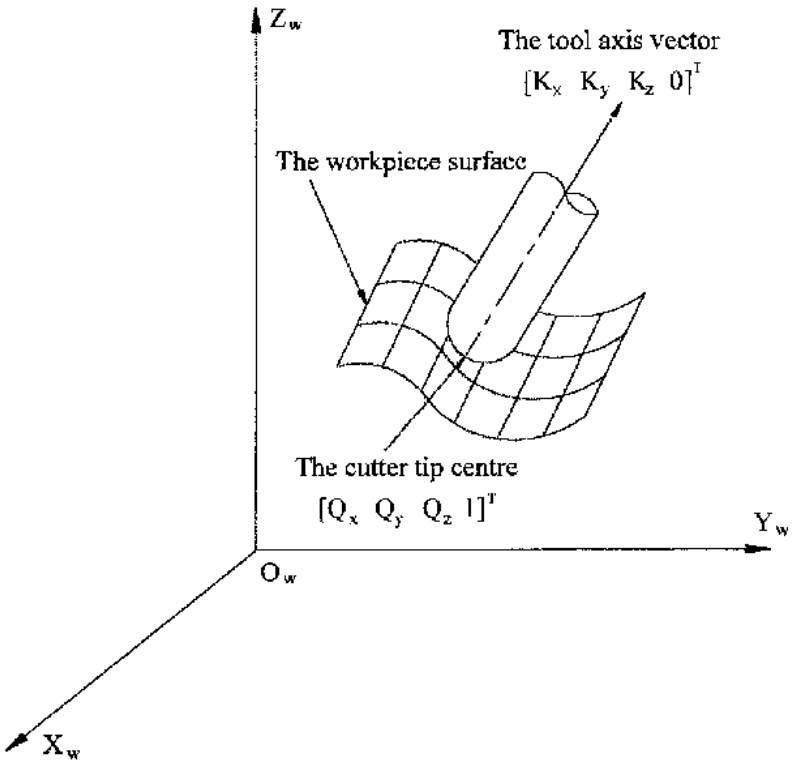


図 17-125

15.4 平行移動と回転行列

同種変換は、多軸機械運動学の数学を記述する簡単な方法を提供します。 空間 H の変換は、4x4 の行列であり、平行移動と回転の変換を表すことができます。 ベクトル u = {x、y、z、1} T で記述された点 x、y、x が与えられると、その変換 v は行列積で表されます。

$$v = H \cdot u$$

5 軸運動学の基礎となる 4 つの基本的な変換行列があります。

$$T(a,b,c) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R(X,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\theta & -S\theta & 0 \\ 0 & S\theta & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2)$$

$$R(Y,\theta) = \begin{bmatrix} C\theta & 0 & S\theta & 0 \\ 0 & 1 & 0 & 0 \\ -S\theta & 0 & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R(Z,\theta) = \begin{bmatrix} C\theta & -S\theta & 0 & 0 \\ S\theta & C\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3)$$

行列T（a、b、c）は、X、Y、Z座標方向にそれぞれa、b、cの量だけ平行移動することを意味します。 R行列は、それぞれX、Y、Z座標軸を中心とした角度シータの回転を意味します。 C記号とS記号は、それぞれ余弦関数と正弦関数を表します。

15.5 テーブル回転/傾斜5軸構成

これらの工作機械では、2つの回転軸が機械の作業台に取り付けられています。 通常、次の2つの形式が使用されます。

- X軸またはY軸（AまたはB回転、一次）を中心に回転する傾斜テーブルに取り付けられた垂直Z軸（C回転、二次）を中心に回転する回転台。 ワークは回転台に取り付けられます。
- X軸またはY軸（AまたはB回転、二次）を中心に回転する傾斜台を、Z軸（C回転、一次）を中心に回転する回転台に取り付け、ワークを傾斜させます。 テーブル。

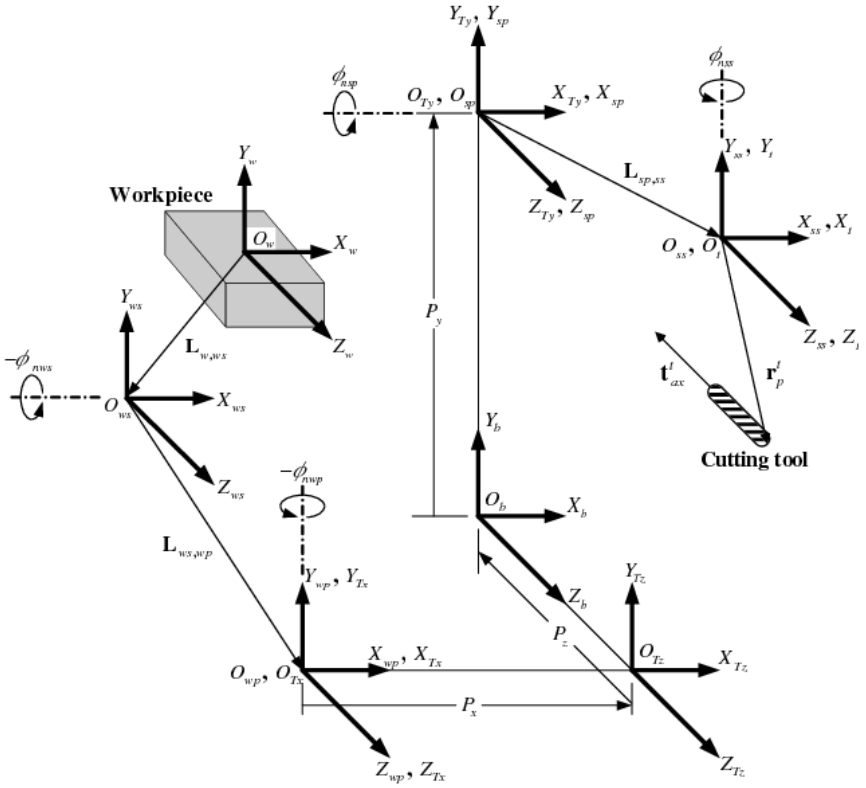


図 17-126

多軸機械は、ジョイントで接続された一連のリンクで構成されていると見なすことができます。 マシンの各リンクに座標フレームを埋め込み、均一な変換を使用することで、これらの座標フレーム間の相対的な位置と方向を記述できます。

ワーク座標系と工具座標系の関係を説明する必要があります。 これは、変換行列 wAt によって定義できます。 これは、それぞれが独自に定義された座標系を持つ、マシンのさまざまな構造要素またはリンク間の後続の変換によって見つけることができます。 一般に、このような変換は次のようになります。

$$ {}^{w}A_t = {}^{w}A_1 \cdot {}^{1}A_2 \cdot {}^{2}A_3 \cdots {}^{n}A_t \qquad (4) $$

ここで、各行列 i-1Aj は、形式（2,3）の変換行列 T または回転行列 R です。

行列の乗算は、左側の行列 A の各行の要素に右側の行列 B の各列の要素を乗算し、合計して結果の行列 C の要素を取得する単純なプロセスです。

$$ C_{ij} = \sum_{k=1,n}^{n} A_{ik} B_{kj}; \quad i = 1,n; \quad j = 1,n $$

図 2 に、座標系を使用した一般的な構成を示します[4]。 これには、テーブルの回転/傾斜軸とスピンドルの回転/傾斜軸が含まれます。 工作機械で実際に使用される回転軸は 2 つだけです。

最初に、上記の最初のタイプの構成の変換を開発します。 回転軸オフセットのないテーブル傾斜/回転（trt）タイプ。 xyzac-trtconfiguration という名前を付けることができます。

同じタイプ（xyzac-trt）の変換も開発しますが、回転軸のオフセットを使用します。

次に、回転軸オフセットを使用した xyzbc-trt 構成の変換を開発します。

15.5.1 作業オフセットのある xyzac-trt 工作機械の変換

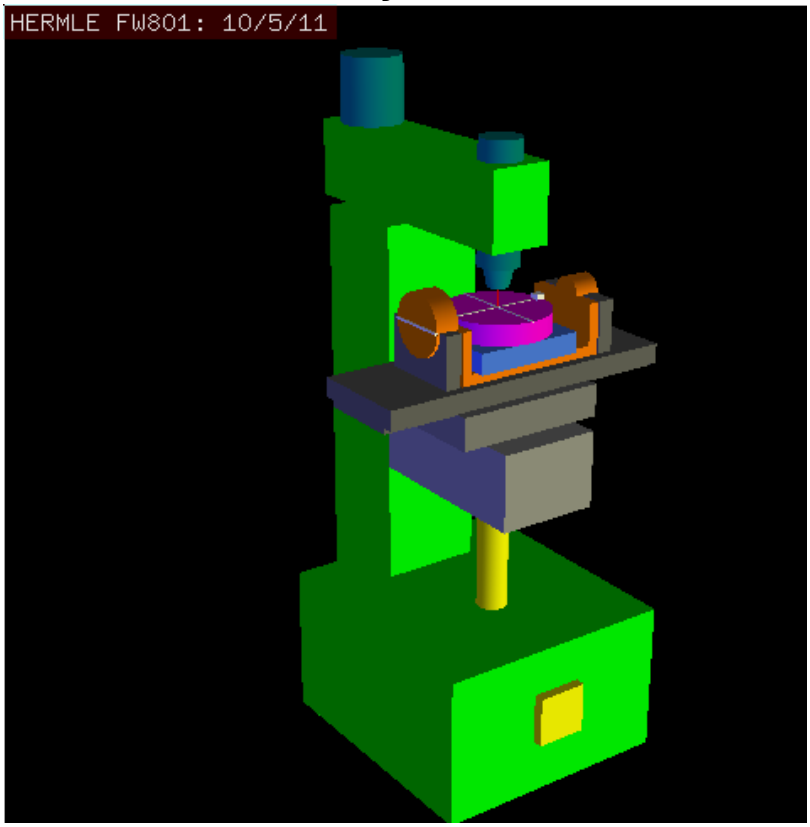


図 17-127

ここでは、図 4 に示すように、傾斜軸と回転軸がピボット点と呼ばれる点で交差する単純化された構成を扱います。したがって、図 2 の 2 つの座標系 Ows と Owp は一致します。

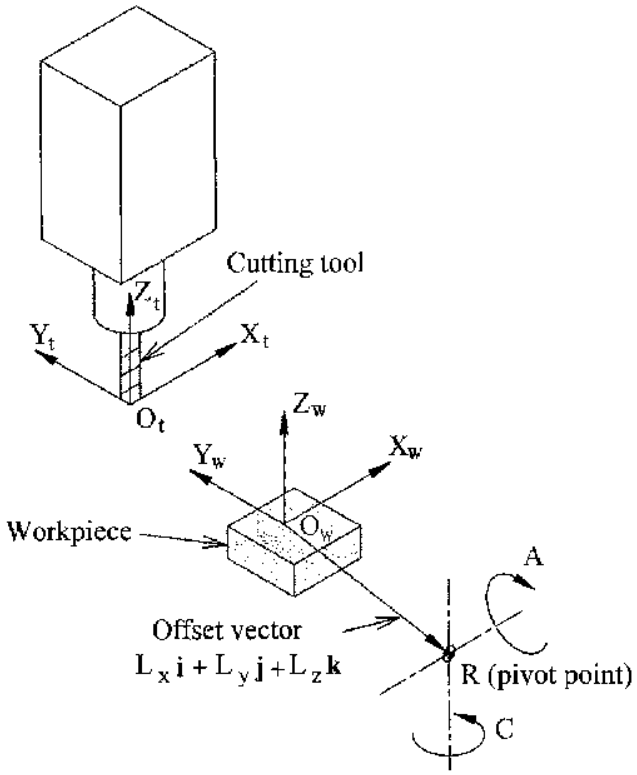


図 17-128

1.1.1.1 フォワードトランスフォーメーション

変換は、行列の順次乗算によって定義できます。

$$ {}^{w}A_t = {}^{w}A_C \cdot {}^{C}A_A \cdot {}^{A}A_P \cdot {}^{P}A_t \qquad (5) $$

次のように構築されたマトリックスを使用します。

$$ {}^{}A_C = \begin{bmatrix} 1 & 0 & 0 & L_x \\ 0 & 1 & 0 & L_y \\ 0 & 0 & 1 & L_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{C}A_A = \begin{bmatrix} C_C & S_C & 0 & 0 \\ -S_C & C_C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6) $$

$$ {}^{A}A_P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_A & S_A & 0 \\ 0 & -S_A & C_A & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{P}A_t = \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (7) $$

これらの方程式 Lx、Ly、Lz は、ワーク座標系の原点に対する 2 つの回転軸 A および C のピボットポイントのオフセットを定義します。 さらに、Px、Py、Pz は、ピボットポイントからカッターチップ位置までの相対距離で

あり、ピボットポイントの「ジョイント座標」とも呼ばれます。 ピボットポイントは、2 つの回転軸の交点にあります。 SA および SC 項の符号は、[2,3]の符号とは異なります。これは、テーブルの回転がワークピースの座標軸に対して負であるためです（sin（-theta）= -sin（theta）、cos（-theta）に注意してください）。 ）= cos（theta））。

（5）に従って乗算すると、次のようになります。

$$ {}^{w}A_t = \begin{bmatrix} C_C & S_C C_A & S_C S_A & C_C P_x + S_C C_A P_y + S_C S_A P_z + L_x \\ -S_C & C_C C_A & C_C S_A & -S_C P_x + C_C C_A P_y + C_C S_A P_z + L_y \\ 0 & -S_A & C_A & -S_A P_y + C_A P_z + L_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (8) $$

これで、このマトリックスの 3 番目の列を、指定されたツールの方向ベクトル K と同等にすることができます。

$$ K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} = \begin{bmatrix} S_C S_A \\ C_C S_A \\ C_A \\ 0 \end{bmatrix} \qquad (9) $$

これらの方程式から、回転角シータ、シータを解くことができます。 3 行目から次のことがわかります。

$$ \theta_A = cos^{-1}(K_z) \qquad (0 < \theta_A < \pi) \qquad (10) $$

そして、最初の行を 2 番目の行で割ると、次のようになります。

$$ \theta_C = tan2^{-1}(K_x, K_y) \qquad (-\pi < \theta_C < \pi) \qquad (11) $$

これらの関係は通常、CAM ポストプロセッサで、ツールの方向ベクトルを回転角に変換するために使用されます。

（8）の最後の列を工具位置ベクトル Q と等しくすると、次のように書くことができます。

$$ \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C P_x + S_C C_A P_y + S_C S_A P_z + L_x \\ -S_C P_x + C_C C_A P_y + C_C S_A P_z + L_y \\ -S_A P_y + C_A P_z + L_z \\ 1 \end{bmatrix} \qquad (12) $$

右側のベクトルは、行列とベクトルの積として記述して、次のようにすることもできます。

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C & S_C C_A & S_C S_A & L_x \\ -S_C & C_C C_A & C_C S_A & L_y \\ 0 & -S_A & C_A & L_z \\ 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = {}^{Q}A_P \cdot P \qquad (13) $$

これを拡張して、

$$\begin{aligned} Q_x &= C_c P_x + S_C C_A P_y + S_C S_A P_z + L_x \\ Q_y &= -S_C P_x + C_C C_A P_y + C_C S_A P_z + L_y \\ Q_z &= -S_A P_y + C_A P_z + L_z \end{aligned} \qquad (14)$$

これは運動学の順方向変換です。

15.5.1.1 逆変換

式（13）から P を P =（QAP）-1 * Q として解くことができます。正方行列は、回転行列 R と並進ベクトル q を含む均質な 4x4 行列であり、その逆行列は次のように記述できます。

$${}^{q}A_p = \begin{bmatrix} R & q \\ 0 & 1 \end{bmatrix} \qquad ({}^{Q}A_P)^{-1} = \begin{bmatrix} R^T & -R^T q \\ 0 & 1 \end{bmatrix} \qquad (15)$$

ここで、R^T は R の転置です（行と列が入れ替わっています）。 したがって、次のようになります。

$$\begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C & -S_C & 0 & -C_C L_x + S_C L_y \\ S_C C_A & C_C C_A & -S_A & -S_C C_A L_x - C_C C_A L_y + S_A L_z \\ S_C S_A & C_C S_A & C_A & -S_C S_A L_x - C_C S_A L_y - C_A L_z \\ 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \qquad (16)$$

したがって、運動学の逆変換に必要な方程式は、次のように書くことができます。

$$\begin{aligned} P_x &= C_c(Q_x - L_x) - S_C(Q_y - L_y) \\ P_y &= S_C C_A(Q_x - L_x) + C_C C_A(Q_y - L_y) - S_A(Q_z - L_z) \\ P_z &= S_C S_A(Q_x - L_x) + C_C S_A(Q_y - L_y) + C_A(Q_z - L_z) \end{aligned} \qquad (17)$$

15.5.2 回転軸オフセットを使用した xyzac-trt マシンの変換

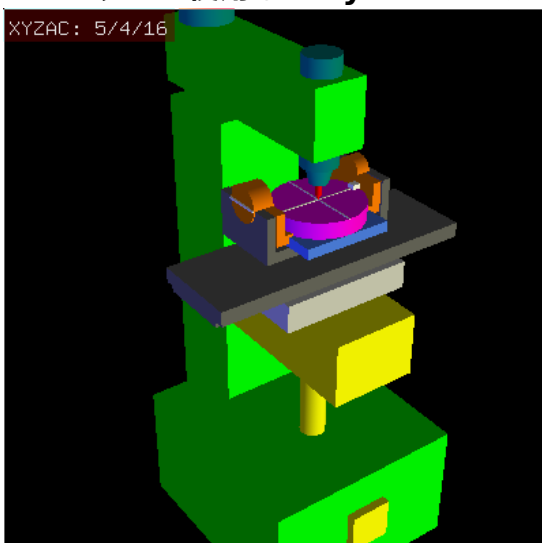


図 17-129　回転軸オフセット（正）を使用した XYZAC-TRT の VISMACH モデル

ここでは、傾斜軸と回転軸が 1 点で交差せず、オフセット Dy を持つ拡張構成を扱います。 さらに、図 2 の 2 つの座標系 Ows と Owp の間には、Dz と呼ばれる z オフセットもあります。 vismach モデルを図 5 に示し、オフセットを図 6 に示します（この例では正のオフセット）。 構成を簡素化するために、前のケースのオフセット Lx、Ly、Lz は含まれていません。 LinuxCNC で「タッチ」機能を使用して G54 オフセットを使用する場合は、おそらくこれらは必要ありません。

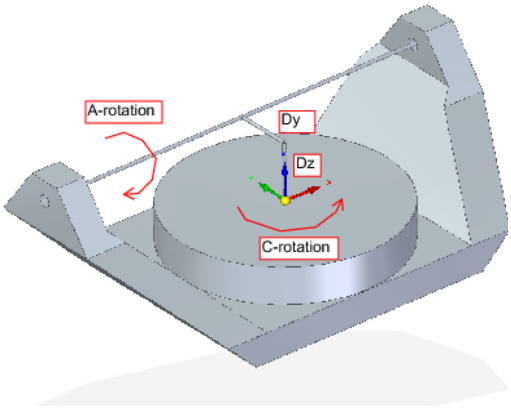


図 17-130　テーブル傾斜/回転 XYZAC-TRT 構成、軸オフセットあり

1.1.1.1　フォワードトランスフォーメーション

変換は、行列の順次乗算によって定義できます。

$$ {}^{w}A_t = {}^{w}A_O \cdot {}^{O}A_A \cdot {}^{A}A_P \cdot {}^{P}A_t \qquad (18) $$

次のように構築されたマトリックスを使用します。

$$ {}^{w}A_O = \begin{bmatrix} C_C & S_C & 0 & 0 \\ -S_C & C_C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{O}A_A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & D_y \\ 0 & 0 & 1 & D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (19) $$

$$ {}^{A}A_P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_A & S_A & 0 \\ 0 & -S_A & C_A & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{P}A_t = \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y - D_y \\ 0 & 0 & 1 & P_z - D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (20) $$

これらの方程式 Dy で、Dz は、ワーク座標系の原点に対する回転軸 A のピボットポイントのオフセットを定義します。 さらに、Px、Py、Pz は、ピボットポイントからカッターチップ位置までの相対距離であり、ピボットポイントの「ジョイント座標」とも呼ばれます。 ピボットポイントは A 回転軸上にあります。

（18）に従って乗算すると、次のようになります。

$$ {}^{w}A_t = \begin{bmatrix} C_C & S_C C_A & S_C S_A & C_C P_x + S_C C_A (P_y - D_y) + S_C S_A (P_z - D_z) + S_C D_y \\ -S_C & C_C C_A & C_C S_A & -S_C P_x + C_C C_A (P_y - D_y) + C_C S_A (P_z - D_z) + C_C D_y \\ 0 & -S_A & C_A & -S_A (P_y - D_y) + C_A (P_z - D_z) + D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (21) $$

これで、このマトリックスの 3 番目の列を、指定されたツールの方向ベクトル K と同等にすることができます。

$$K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} = \begin{bmatrix} S_C S_A \\ C_C S_A \\ C_A \\ 0 \end{bmatrix} \qquad (22)$$

これらの方程式から、回転角シータ、シータを解くことができます。 3 行目から次のことがわかります。

$$\theta^{-1} = \cos_{-1}(K^z) \qquad (0 < \theta^{-1} < \pi) \qquad (53)$$

2 番目の行を最初の行で割ると次のようになります。

$$\theta_C = tan2^{-1}(K_x, K_y) \qquad (-\pi < \theta_C < \pi) \qquad (24)$$

これらの関係は通常、CAM ポストプロセッサで、ツールの方向ベクトルを回転角に変換するために使用されます。

（21）の最後の列を工具位置ベクトル Q と等しくすると、次のように書くことができます。

$$Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C P_x + S_C C_A (P_y - D_y) + S_C S_A (P_z - D_z) + S_C D_y \\ -S_C P_x + C_C C_A (P_y - D_y) + C_C S_A (P_z - D_z) + C_C D_y \\ -S_A (P_y - D_y) + C_A (P_z - D_z) + D_z \\ 1 \end{bmatrix} \qquad (25)$$

右側のベクトルは、行列とベクトルの積として記述して、次のようにすることもできます。

$$Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C & S_C C_A & S_C S_A & -S_c C_A D_y - S_C S_A D_z + S_C D_y \\ -S_C & C_C C_A & C_C S_A & -C_C C_A D_y - C_C S_A D_z + C_C D_y \\ 0 & -S_A & C_A & S_A D_y - C_A D_z + D_z \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} =^Q A_P \cdot P \qquad (26)$$

これは運動学の順方向変換です。

15.5.2.1 逆変換

前と同じように（15）を使用して、式（25）から P =（QAP）-1 * Q として P を解くことができます。 これにより、次のものが得られます。

$$\begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C & -S_C & 0 & 0 \\ S_C C_A & C_C C_A & -S_A & -C_A D_y + S_A D_z + D_y \\ S_C S_A & C_S S_A & C_A & -S_A D_y - C_A D_z + D_z \\ 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \qquad (27)$$

したがって、運動学の逆変換に必要な方程式は、次のように書くことができます。

$$\begin{aligned} P_x &= C_c Q_x - S_C Q_y \\ P_y &= S_C C_A Q_x + C_C C_A Q_y - S_A Q_z - C_A D_y + S_A D_z + D_y \\ P_z &= S_C S_A Q_x + C_S S_A Q_y + C_A Q_z - S_A D_y - C_A D_z + D_z \end{aligned} \qquad (28)$$

15.5.3 回転軸オフセットを使用した xyzbc-trt マシンの変換

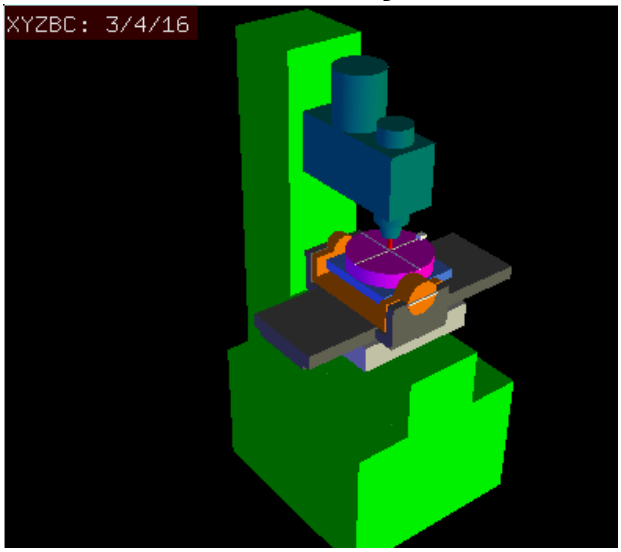


図 17-131 回転軸オフセット（負）を使用した XYZBC-TRT の VISMACH モデル

ここでも、傾斜軸（y 軸を中心に）と回転軸が点で交差せず、オフセット Dx を持つ拡張構成を扱います。 さらに、図 2 の 2 つの座標系 Ows と Owp の間には、Dz と呼ばれる z オフセットもあります。 vismach モデルを図 7（この例では負のオフセット）に示し、正のオフセットを図 8 に示します。

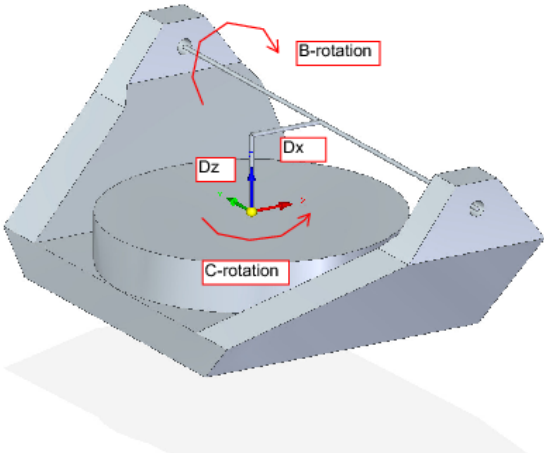


図 17-132 テーブル傾斜/回転 XYZBC-TRT 構成、軸オフセットあり

1.1.1.1 フォワードトランスフォーメーション

変換は、行列の順次乗算によって定義できます。

$$ {}^{w}A_t = {}^{w}A_O \cdot {}^{O}A_B \cdot {}^{B}A_P \cdot {}^{P}A_t \qquad (29) $$

次のように構築されたマトリックスを使用します。

$$ {}^{W}A_{O} = \begin{bmatrix} C_C & S_C & 0 & 0 \\ -S_C & C_C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{O}A_{B} = \begin{bmatrix} 1 & 0 & 0 & D_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{30} $$

$$ {}^{B}A_{P} = \begin{bmatrix} C_B & 0 & -S_B & 0 \\ 0 & 1 & 0 & 0 \\ S_B & 0 & C_B & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{P}A_{t} = \begin{bmatrix} 1 & 0 & 0 & P_x - D_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z - D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{31} $$

これらの方程式 Dx で、Dz は、ワーク座標系の原点に対する回転軸 B のピボットポイントのオフセットを定義します。 さらに、Px、Py、Pz は、ピボットポイントからカッターチップ位置までの相対距離であり、ピボットポイントの「ジョイント座標」とも呼ばれます。 ピボットポイントは B 回転軸上にあります。

（29）に従って乗算すると、次のようになります。

$$ {}^{W}A_{t} = \begin{bmatrix} C_C C_B & S_C & -C_C S_B & C_C C_B(P_x - D_x) + S_C P_y - C_C S_B(P_z - D_z) + C_C D_x \\ -S_C C_B & C_C & S_C S_B & -S_C C_B(P_x - D_X) + C_C P_y + S_C S_B(P_z - D_z) - S_C D_x \\ S_B & 0 & C_B & S_B(P_x - D_x) + C_B(P_z - D_z) + D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{32} $$

これで、このマトリックスの 3 番目の列を、指定されたツールの方向ベクトル K と同等にすることができます。

$$ K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} = \begin{bmatrix} -C_C S_B \\ S_C S_B \\ C_B \\ 0 \end{bmatrix} \tag{33} $$

これらの方程式から、回転角 thetaB、thetaC を解くことができます。 3 行目から次のことがわかります。＋

$$ \theta_B = cos^{-1}(K_z) \qquad (0 < \theta_B < \pi) \tag{34} $$

2 番目の行を最初の行で割ると次のようになります。

$$ \theta_C = tan2^{-1}(K_y, K_x) \qquad (-\pi < \theta_C < \pi) \tag{35} $$

これらの関係は通常、CAM ポストプロセッサで、ツールの方向ベクトルを回転角に変換するために使用されます。

（32）の最後の列を工具位置ベクトル Q と等しくすると、次のように書くことができます。

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C C_B(P_x - D_x) + S_C P_y - C_C S_B(P_z - D_z) + C_C D_x \\ -S_C C_B(P_x - D_X) + C_C P_y + S_C S_B(P_z - D_z) - S_C D_x \\ S_B(P_x - D_x) + C_B(P_z - D_z) + D_z \\ 1 \end{bmatrix} \tag{36} $$

右側のベクトルは、行列とベクトルの積として記述して、次のようにすることもできます。

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C C_B & S_C & -C_C S_B & -C_C C_B D_x + C_C S_B D_z + C_C D_x \\ -S_C C_B & C_C & S_C S_B & S_C C_B D_x - S_C S_B D_z - S_C D_x \\ S_B & 0 & C_B & -S_B D_x - C_B D_z + D_z \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = {}^{Q}A_{P} \cdot P \tag{37} $$

これは運動学の順方向変換です。

15.5.3.1 逆変換

式（37）から P を P =（QAP）-1 * Q として解くことができます。前と同じアプローチで、次のようになります。

$$\begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C C_B & -S_C C_B & S_B & -C_B D_x - S_B D_z + D_x \\ S_C & C_C & 0 & 0 \\ -C_C S_B & S_C S_B & C_B & S_B D_x - C_B D_z + D_z \\ 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \quad (38)$$

したがって、運動学の逆変換に必要な方程式は、次のように書くことができます。

$$\begin{aligned} P_x &= C_c C_B Q_x - S_C C_B Q_y + S_B Q_z - C_B D_x - S_B D_z + D_x \\ P_y &= S_C Q_x + C_C Q_y \\ P_z &= -C_C S_B Q_x + S_C S_B Q_y + C_B Q_z + S_B D_x - C_B D_z + D_z \end{aligned} \quad (39)$$

15.6 表回転/傾斜の例

LinuxCNC には、上記の数学で説明されている xyzac-trt および xyzbc-trt トポロジ用のキネマティクスモジュールが含まれています。 興味のあるユーザーの場合、ソースコードは src / emc / kinematics /ディレクトリの git ツリーにあります。

xyzac-trt および xyzbc-trt シミュレーション構成の例は、サンプル構成（configs / sim / axis / vismach / 5axis tablerotary-tilting /）ディレクトリにあります。

構成例には、必要な ini ファイルと gode（.ngc）ファイルを含む subdirectory の例が含まれています。 これらの sim 構成は、LinuxCNCvismach 機能を使用して現実的な 3 次元モデルを呼び出します。

15.6.1 Vismach シミュレーションモデル

Vismach は、PC 画面に CNC マシンの動的シミュレーションを表示する Python ルーチンのライブラリです。 特定のマシンの Python スクリプトは HAL にロードされ、データは HAL ピン接続によって渡されます。 ユーザースペース vismach モデルは、次のような hal コマンドによってロードされます。

```
loadusr -W xyzac-trt-gui
```

接続は、次のような HAL コマンドを使用して行われます。

```
net :table-x joint.0.pos-fb xyzac-trt-gui.table-x
net :saddle-y joint.1.pos-fb xyzac-trt-gui.saddle-y
...
```

vismach モデルに使用される HAL 接続の詳細については、シミュレーション ini ファイルを参照してください。

15.6.2 工具長補正

工具長補正が自動的に適用された工具テーブルの工具を順番に使用するには、さらにZオフセットが必要です。通常はツールの長さがゼロである「マスター」ツールよりも長いツールの場合、LinuxCNCには「motion.tooloffset.z」という変数があります。この変数がキネマティックコンポーネント（およびvismach pythonスクリプト）に渡される場合、新しいツールに必要な追加のZオフセットは、コンポーネントステートメントを追加することで説明できます。次に例を示します。

$$D_z = D_z + \text{tool-offset}$$

必要なHAL接続（xyzac-trtの場合）は次のとおりです。

```
net :tool-offset motion.tooloffset.z xyzac-trt-kins.tool-offset
```

どこ：

```
:tool-offset ---------------- signal name
motion.tooloffset.z --------- output HAL pin from LinuxCNC motion module
xyzac-trt-kins.tool-offset -- input HAL pin to xyzac-trt-kins
```

15.7 カスタムキネマティクスコンポーネント

LinuxCNCは、LinuxCNCの起動時にロードされるHALコンポーネントを使用してキネマティクスを実装します。最も一般的なキネマティクスモジュールであるtrivkinsは、軸座標文字とモータージョイントの間に1対1の対応があるアイデンティティ（トリビアル）キネマティクスを実装します。より複雑なシステム（上記のxyzac-trtおよびxyzbc-trtを含む）用の追加のキネマティクスモジュールが利用可能です。

使用可能なキネマティクスモジュールの簡単な説明については、kinsのマンページ（\$ man kins）を参照してください。

LinuxCNCが提供するキネマティクスモジュールは、通常、C言語で記述されています。標準構造を使用しているため、既存のソースファイルを新しい名前のユーザーファイルにコピーして変更し、インストールすることで、カスタムキネマティクスモジュールの作成が容易になります。

インストールはhalcompileを使用して行われます。

```
sudo halcompile --install kinsname.c
```

ここで、「kinsname」はコンポーネントに付ける名前です。インストールにはsudoプレフィックスが必要であり、rootパスワードの入力を求められます。詳細については、halcompileのmanページを参照してください（\$ man halcompile）

コンパイルしてインストールしたら、マシンの構成セットアップで参照できます。これは、configディレクトリのiniファイルで行われます。たとえば、一般的なiniの仕様は次のとおりです。

```
[KINS]
KINEMATICS = trivkins
```

に置き換えられます

```
[KINS]
KINEMATICS = kinsname
```

ここで、「kinsname」は、kins プログラムの名前です。 xyzac-trt キネマティクスモジュールで使用される Dx、Dy、Dz、ツールオフセットなどの可変構成アイテム用に、モジュールによって追加の HAL ピンが作成される場合があります。 これらのピンは、動的制御用の信号に接続することも、次のような HAL 接続で 1 回設定することもできます。

```
# set offset parameters
net :tool-offset motion.tooloffset.z xyzac-trt-kins.tool-offset
setp xyzac-trt-kins.y-offset 0
setp xyzac-trt-kins.z-offset 20
```

15.8 フィギュア

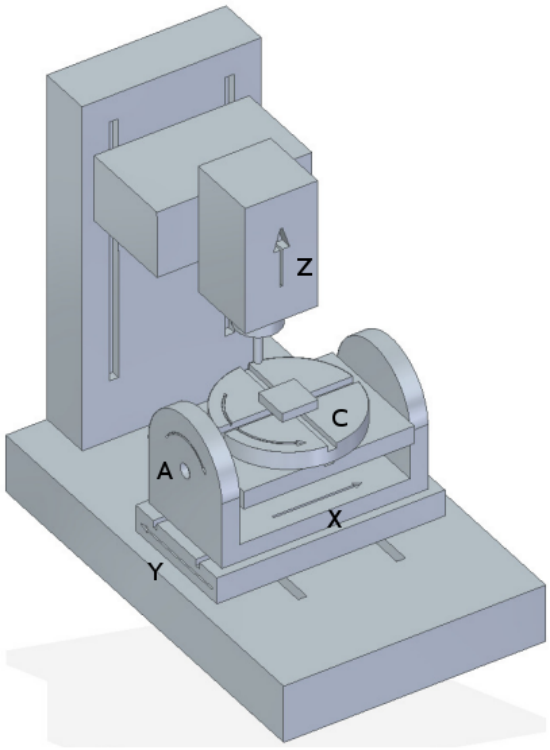


図 17-133 テーブルの傾斜/回転構成

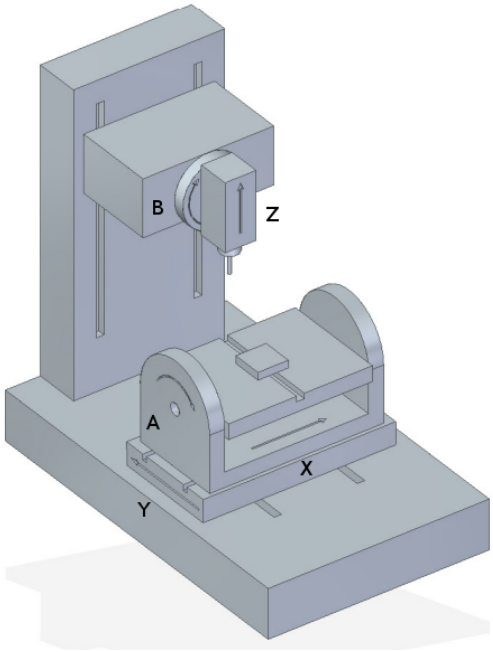


図 17-134　スピンドル/テーブル傾斜構成

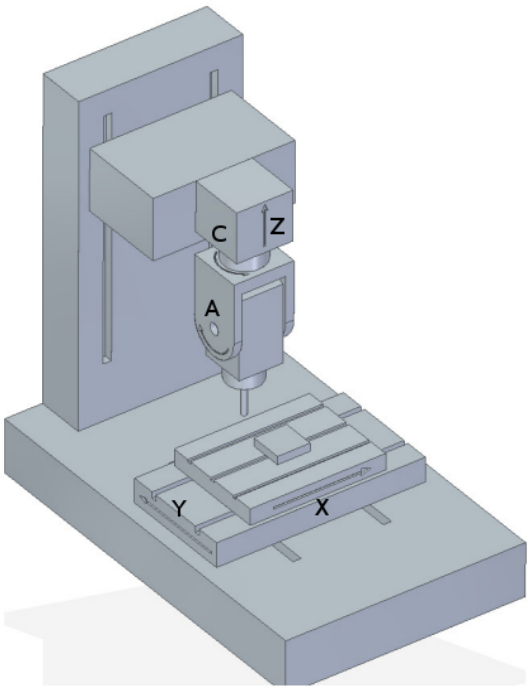


図 17-135　スピンドル傾斜/回転構成

15.9　参考文献

1. A Postprocessor Based on the Kinematics Model for General Five-Axis machine Tools: C-H She, R-S Lee, J Manufacturing

Processes, V2 N2, 2000.

2. NC Post-processor for 5-axis milling of table-rotating/tilting type: YH Jung, DW Lee, JS Kim, HS Mok, J Materials

Processing Technology,130-131 (2002) 641-646.

3. 3D 6-DOF Serial Arm Robot Kinematics, RJ du Preez, SA-CNC-CLUB, Dec. 5, 2013.

4. Design of a generic five-axis postprocessor based on generalized kinematics model of machine tool: C-H She, C-C Chang,

Int. J Machine Tools & Manufacture, 47 (2007) 537-545.

16章 PID チューニング

16.1 PID コントローラ

比例-積分-微分コントローラー（PID コントローラー）は、産業用制御システムの一般的なフィードバックループコンポーネントです。1

コントローラは、プロセス（通常は工業プロセス）からの測定値を基準設定値と比較します。 次に、差（またはエラー信号）を使用して、プロセスへの操作可能な入力の新しい値を計算し、プロセスの測定値を目的の設定値に戻します。

より単純な制御アルゴリズムとは異なり、PID コントローラーはエラー信号の履歴と変化率に基づいてプロセス出力を調整できるため、より正確で安定した制御が可能になります。 （単純な比例制御に定常状態エラーがあるか、プロセスが振動する場合、PID ループが正確で安定した制御を生成することを数学的に示すことができます）。

NOTE

1 このサブセクションは、HTTP：//EN.WIKIPEDIA.ORG/WIKI/PID_CONTROLLER にあるはるかに広範な記事から抜粋したものです。

16.1.1 制御ループの基本

直感的に、PID ループは、ゲージとコントロールノブを備えたインテリジェントなオペレーターが行うことを自動化しようとします。 オペレーターは、プロセスの出力測定値を示すゲージを読み取り、プロセスの出力測定値がゲージの目的の値で安定するまで、ノブを使用してプロセスの入力（アクション）を調整します。

古い制御文献では、この調整プロセスはリセットアクションと呼ばれています。 ゲージ上の針の位置は、測定値、プロセス値、またはプロセス変数です。 ゲージの望ましい値は、設定値（設定値とも呼ばれます）と呼ばれます。 ゲージの針と設定値の違いが誤差です。

制御ループは、次の 3 つの部分で構成されます。

1. プロセスに接続されたセンサー（エンコーダーなど）による測定、
2. コントローラ要素の決定、
3. モーターなどの出力デバイスを介したアクション。

コントローラがセンサーを読み取ると、この測定値が設定値から差し引かれ、エラーが決定されます。 次に、エラーを使用してプロセスの入力変数（アクション）の修正を計算し、この修正によってプロセスの出力測定からエラーが削除されるようにします。

PID ループでは、修正は 3 つの方法でエラーから計算されます。現在のエラーを直接キャンセルする（比例）、エラーが修正されずに継続した時間（積分）、および変化率から将来のエラーを予測する 時間の経過に伴うエラー（微分）。

PID コントローラーを使用して、他のプロセス変数を操作することによって影響を受ける可能性のある測定可能な変数を制御できます。 たとえば、温度、圧力、流量、化学組成、速度、またはその他の変数を制御するために使用できます。 自動車のクルーズコントロールは、粗い PID 制御を利用する業界外のプロセスの一例です。

一部の制御システムは、カスケードまたはネットワークに PID コントローラーを配置します。つまり、マスターコントロールは、スレーブコントローラーが使用する信号を生成します。 一般的な状況の 1 つは、モーター制御です。多くの場合、モーターの速度を制御し、スレーブコントローラー（多くの場合、可変周波数ドライブに組み込まれています）が比例入力に基づいて速度を直接管理します。 このスレーブ入力は、関連する変数に基づいて制御しているマスターコントローラーの出力によって供給されます。

16.1.2 理論

PID は、3 つの修正計算にちなんで名付けられています。これらの計算はすべて、管理された数量に追加され、調整されます。 比率は通常負であるため、これらの加算は実際にはエラーの減算です。

比例現在を処理するために、誤差は（負の）定数 P（比例の場合）で乗算され、制御された量に加算（誤差を減算）されます。 P は、コントローラーの出力がシステムのエラーに比例する帯域でのみ有効です。 エラーがゼロの場合、比例コントローラーの出力はゼロであることに注意してください。

積分過去から学ぶために、誤差は一定期間にわたって積分（合計）され、次に（負の）定数 I（平均を作成）で乗算され、制御された量に加算（誤差を減算）されます。 測定された誤差を平均して、設定値からのプロセス出力の平均誤差を見つけます。 単純な比例システムは、オーバーシュート時にエラーを除去するものがないために設定点の周りを前後に移動するか、低すぎる値または高すぎる値で振動および/または安定します。 プロセス入力から平均誤差の負の割合を加算する（つまり、その一部を減算する）ことにより、プロセス出力と設定値の間の平均差は常に減少します。 したがって、最終的には、適切に調整された PID ループのプロセス出力が設定値に落ち着きます。

導関数将来を処理するために、時間の経過に伴う 1 次導関数（誤差の傾き）が計算され、別の（負の）定数 D が乗算され、制御された量に加算（誤差が減算）されます。 微分項は、システムの変更に対する応答を制御します。 微分項が大きいほど、コントローラーはプロセスの出力の変化により迅速に応答します。

より技術的には、PID ループは、複雑な周波数領域システムに適用されるフィルターとして特徴付けることができます。 これは、実際に安定した値に達するかどうかを計算するのに役立ちます。 値が正しく選択されていない場合、制御されたプロセス入力が振動する可能性があり、プロセス出力が設定値に留まらない可能性があります。

16.1.3 ループチューニング

制御ループの調整とは、その制御パラメーター（ゲイン/比例帯域、積分ゲイン/リセット、微分ゲイン/レート）を目的の制御応答の最適値に調整することです。 プロセス変更または設定値変更の最適な動作は、アプリケーションによって異なります。 一部のプロセスでは、設定値からのプロセス変数のオーバーシュートを許可してはなりません。 他のプロセスでは、新しい設定値に到達するために消費されるエネルギーを最小限に抑える必要があります。 一般に、応答の安定性が必要であり、プロセスは、プロセス条件と設定値の任意の組み合わせに対して振動してはなりません。

ループの調整は、プロセスの応答時間によってさらに複雑になります。 設定値を変更して安定した効果が得られるまで、数分から数時間かかる場合があります。 一部のプロセスにはある程度の非線形性があるため、プロセスが無負荷から起動している場合、全負荷状態で適切に機能するパラメータは機能しません。 このセクションでは、ループチューニングの従来の手動による方法について説明します。

PID ループを調整する方法はいくつかあります。 方法の選択は、チューニングのためにループをオフラインにできるかどうか、およびシステムの応答速度に大きく依存します。 システムをオフラインにできる場合、最良の調整方法は、多くの場合、システムに入力のステップ変化を与え、時間の関数として出力を測定し、この応答を使用して制御パラメーターを決定することです。

簡単な方法システムをオンラインのままにする必要がある場合、1 つの調整方法は、最初に I 値と D 値をゼロに設定することです。 ループの出力が発振するまで P を上げます。 次に、発振が停止するまで I を増やします。 最後に、ループがその基準に到達するのに許容できる速さになるまで、D を増やします。 通常、高速 PID ループ調整はわずかにオーバーシュートして、設定値にすばやく到達します。 ただし、一部のシステムはオーバーシュートを受け入れることができません。

Parameter	立ち上がり時間	オーバーシュート	整定時間	定常状態エラー
P	減少	増加	小さな変化	減少
I	減少	増加	増加	排除
D	小さな変化	減少	減少	小さな変化

パラメータの増加の影響

ZIEGLER-NICHOLS 法別の調整方法は、正式には ZIEGLER-NICHOLS 法として知られており、JOHN G.ZIEGLER と NATHANIELB.NICHOLS によって導入されました。 これは、前述の方法と同じ方法で開始します。最初に I ゲインと D ゲインをゼロに設定し、次に P ゲインを増やして、ループを外部干渉にさらします。たとえば、モーター軸をノックして、順番に平衡状態から外します。 ループの出力が発振し始めるまでの臨界ゲインと発振周期を決定します。 臨界ゲイン（Kc）と出力の発振周期（Pc）を書き留めます。 次に、表に示すように、P、I、および D コントロールを調整します。

Control type	P	I	D
P	$.5K_c$		
PI	$.45K_c$	$P_c/1.2$	
PID	$.6K_c$	$P_c/2$	$P_c/8$

最終ステップ軸を調整した後、HALSCOPE で次のエラーをチェックして、マシンの要件内にあることを確認します。 HALSCOPE の詳細については、HAL ユーザーマニュアルを参照してください。

19章　外部軸補正

外部軸オフセットは、テレオプ（ワールド）ジョグおよび調整（gcode）モーション中にサポートされます。外部軸オフセットは、ini ファイル設定によって軸ごとに有効になり、hal 入力ピンによって動的に制御されます。hal インターフェースは、ホイールジョギングに使用されるものと似ています。 このタイプのインターフェースは、通常、パルスをカウントするエンコーダー hal コンポーネントに接続された手動パルスジェネレーター（mpg）を使用して実装されます。

16.2　Ini ファイルの設定

各軸文字（xyzabcuvw の L）について：

各軸文字（xyzabcuvw の L）について：

```
[AXIS_L]OFFSET_AV_RATIO = value (controls accel/vel for external offsets)
```

1. 許可される値：0 <=値<= 0.9
2. 許可されていない値は、stdout へのメッセージとともに 0.1 に置き換えられます
3. デフォルト値：0（外部オフセットを無効にします）。 結果：省略[AXIS_L] OFFSET_AV_RATIO は、軸の外部オフセットを無効にします。
4. ゼロ以外の場合、OFFSET_AV_RATIO（r）は、[AXIS_L]制約を維持するために、従来の（計画）最大速度と加速度を調整します。

```
planning max velocity = (1-r) * MAX_VELOCITY
external offset velocity = ( r) * MAX_VELOCITY
```

```
planning max acceleration = (1-r) * MAX_ACCELERATIOIN
external offset acceleration = ( r) * MAX_ACCELERATION
```

16.3　HAL ピン

16.3.1　軸ごとのモーション HAL ピン

各軸の文字（xyzabcuvw の L）

1. axis.L.eoffset-enable Input（bit）：enable
2. axis.L.eoffset-scale Input（float）：スケールファクター
3. axis.L.eoffset-counts Input（s32）：カウントレジスタへの入力
4. axis.L.eoffset-clear Input（bit）：要求されたオフセットをクリアします

5. axis.L.eoffset Output（float）：現在の外部オフセット

6. axis.L.eoffset-request Output（float）：要求された外部オフセット

16.3.2 その他のモーション HAL ピン

1. motion.eoffset-active Output（bit）：ゼロ以外の外部オフセットが適用されます

2. motion.eoffset-limited Output（bit）：ソフト制限のためにモーションが禁止されました

16.4 使用法

軸入力 hal ピン（enable、scale、counts）は、ホイールジョギングに使用されるピンと同様です。

16.4.1 オフセット計算

各サーボ周期で、axis.L.eoffset-counts ピンが前の周期の値と比較されます。 axis.L.eoffset-counts ピンの増加または減少（正または負のデルタ）に現在の axis.L.eoffset-scale ピンの値を掛けます。 この製品は内部レジスタに蓄積され、axis.L.eoffset-requesthal ピンにエクスポートされます。 累積レジスタは、マシンがオンになるたびにゼロにリセットされます。

要求されたオフセット値は、L 座標に適用され、axis.L.eoffsethal ピンで表されるオフセットの移動を計画するために使用されます。 計画されたモーションは、割り当てられた速度と加速度の制約を尊重し、正味のモーション（オフセットとテレオプジョギングまたは協調モーション）が L 座標のソフト制限に達した場合に制限される可能性があります。

多くのアプリケーションでは、axis.L.eoffset-scale ピンは一定であり、net axis.L.eoffset-axis.L.eoffset-counts に対する応答を要求すると、axis.L.eoffset-の累積値の積に相当します。 カウントと（一定の）軸。L.eoffset-scale ピン値。

16.4.2 マシンオフ/マシンオン

機械の電源を切ると、外部オフセットのある現在の位置が維持されるため、電源を切ったり入れたりするときに予期しない動きが発生することはありません。

起動（マシンオン）ごとに、各 hal ピン軸の内部カウントが登録されます。L.eoffset-counts はゼロになり、対応する hal 出力ピン axis.L.eoffset はゼロにリセットされます。

言い換えると、外部オフセットは、axis.L.eoffsetcounts ピンの値に関係なく、各起動時（マシンオン）でゼロとして定義されます。 混乱を避けるために、マシンがオフのときは、すべての axis.L.eoffset-counts ピンをゼロに設定することをお勧めします。

16.4.3 ソフトリミット

外部軸のオフセット動作は、[AXIS_L] OFFSET_AV_で指定された速度と加速度の設定で個別に計画されます。オフセット動作は、テレオプジョグや調整（gcode）動作とは調整されません。 テロープジョギングおよび協調（gcode）モーション中、軸のソフト制限（[AXIS_L] MIN_LIMIT、MAX_LIMIT）は軸の移動を制限します。

外部オフセットが適用され、モーションがソフト制限に達すると（外部オフセットの増加、テレオプジョギング、または協調モーションによって）、hal pin motion.eoffset-limited がアサートされ、軸の値は名目上ソフト制限に保持されます。 この hal ピンは、関連する hal ロジックで使用して、追加の eoffset カウントを切り捨てたり、マシンを停止したりできます（たとえば、halui.machine.off に接続します）。 軸がソフト制限内で移動すると、motion.eoffset-limited ピンがリセットされます。

計画された軸の値を変更し続ける協調動作中にソフト制限で動作している場合、hal 出力ピン axis.L.eoffset は、現在のオフセット（計算されたオフセット要求ではなく、制限に到達するために必要な距離）を示します。 この表示値は、計画軸の値が変化すると変化します。

hal ピン axis.L.eoffset-request は、内部カウントレジスタと eoffset-scale の積である現在要求されているオフセットを示します。 一般に、axis.L.eoffset ピンの値は axis.L.eoffset-request 値よりも遅れています。これは、外部オフセットが加速制限の対象となるためです。 ソフト制限で動作している場合、axis.L.eoffset-counts の追加の更新は、axis.L.eoffset-requesthal ピンに反映されるように要求された外部オフセットに影響を与え続けます。

外部オフセットが有効でゼロ以外の値が適用されているテレオプジョギングの場合、ソフト制限に遭遇すると、減速間隔なしで問題のある軸の動きが停止します。 同様に、外部オフセットが有効になっている協調モーション中に、ソフトリミットに達すると、減速フェーズなしでモーションが停止します。 この場合、オフセットがゼロであるかどうかは関係ありません。

減速フェーズなしでモーションが停止すると、システムの加速制限に違反し、1）サーボモーターシステムの次のエラー（および/またはサンプ）、2）ステッピングモーターシステムのステップの損失につながる可能性があります。 一般に、ソフトリミットに近づかないように外部オフセットを適用することをお勧めします。

16.4.4 ノート

外部オフセットは、軸座標文字（xyzabcuvw）に適用されます。 外部軸オフセットが適用される前に、すべてのジョイントをホームにする必要があります。

axis.L.eoffset-enable hal ピンのオフセットがゼロ以外のときにリセットされた場合、オフセットは維持されます。 オフセットは次の方法でクリアできます。

1. マシンオフ/マシンオントグル
2. イネーブルピンを再アクティブ化し、軸をインクリメント/デクリメントします。L.eoffset-hal ピンをカウントしてオフセットをゼロに戻します。
3. 軸のパルス L.eoffset-クリア HAL ピン

外部オフセットは、ソフト制限範囲内で適用される小さなオフセットで使用することを目的としています。

外部オフセットが適用される場合、テレオプジョギングと協調モーションの両方でソフト制限が尊重されます。 ただし、協調モーション中にソフトリミットに達した場合、計画されたモーションが同じ方向に続くと、問題のある外部オフセットを減らしてもソフトリミットから離れない場合があります。 この状況は、オフセット除去の修正率（[AXIS_L] OFFSET_AV_RATIO で設定）が、反対の計画された動作速度よりも小さい場合があるために発生する可能性があります。 このような場合、計画された調整されたモーションを一時停止（または停止）すると、問題のある外部オフセットに変更が加えられたときに、ソフトリミットから離れる動きが可能になります。

16.4.5 警告

外部オフセットを使用すると、機械の動きを大幅に変えることができます。 hal コンポーネントと接続、および関連するユーザーインターフェイスを使用した外部オフセットの制御は、展開前に慎重に設計およびテストする必要があります。

16.5 関連する Hal コンポーネント

16.5.1 eoffset_per_angle.comp

測定された角度（回転座標またはスピンドル）に基づいて関数から外部オフセットを計算するコンポーネント。 詳細については、man ページを参照してください（$ man eoffset_per_angle）。

16.6 テスト

外部軸オフセット機能は、候補軸ごとに[AXIS_L]設定を追加することで有効になります。 例えば：

```
[AXIS_Z]
OFFSET_AV_RATIO = 0.2
```

テストには、sim_pingui を使用してジョグホイールインターフェイスをシミュレートすると便利です。 たとえば、ターミナルでは次のようになります。

```
$ sim_pin axis.z.eoffset-enable axis.z.eoffset-scale axis.z.eoffset-counts
```

外部オフセットの使用は、現在のオフセットに関連する情報を表示することによって支援されます。現在の eoffset 値と要求された eoffset 値、軸 pos-cmd、および（ID キネマティクスマシンの場合）対応するジョイントモーター pos-cmd と motor- オフセット。 提供されている sim 構成（以下を参照）は、軸 gui の pyvcp パネルの例を示しています。

カスタム表示がない場合、halshow は、カスタムウォッチリストを使用して補助アプリケーションとして開始できます。

hal pin eoffset 接続をシミュレートし、z 軸の eoffset 情報を表示するための ini ファイル設定の例（z == join2 の ID キネマティクスの場合）：

```
[APPLICATIONS]
APP = sim_pin \
axis.z.eoffset-enable \
axis.z.eoffset-scale \
axis.z.eoffset-counts \
axis.z.eoffset-clear
APP = halshow --fformat "%0.5f" ./z.halshow
```

ファイル z.halshow（構成ディレクトリ内）は次のとおりです。

```
pin+joint.2.motor-pos-cmd
pin+joint.2.motor-offset
pin+axis.z.pos-cmd
pin+axis.z.eoffset
pin+axis.z.eoffset-request
pin+motion.eoffset-limited
```

16.7 例

提供されているシミュレーション構成は、実際のハードウェアのユーザーカスタマイズの開始点を提供するための外部オフセットの使用を示しています

sim 構成では、ini 設定[HAL] HALFILE = LIB：basic_sim.tcl を使用して、ini ファイル[TRAJ] COORDINATES = 設定で指定された軸のすべてのルーチン hal 接続を構成します。 外部オフセット機能を示すために必要な hal ロジックと、pyvcp パネルの GUI hal ピン接続は、別々の hal ファイルで作成されます。 非シミュレーション構成では、マシンに適した LIB：basic_sim.tcl アイテムの HALFILE を置き換える必要があります。 提供されている pyvcp ファイル（.hal および.xml）は、アプリケーション固有の GUI インターフェイスの開始点になる可能性があります。

16.7.1 eoffsets_demo.ini

sim config sim / configs / axis / external_offsets / eoffsets_demo.ini は、任意の軸で外部オフセットを有効にするコントロールを備えたデカルト XYZ マシンを示しています。

すべての重要な位置とオフセット値を表示するためのディスプレイが用意されています。

sim_pin gui は、軸オフセットピンのコントロールを提供します：eoffset-scale＆eoffset-counts（信号 e：<L> counts 経由）、eoffset-clear（信号 e：clearall 経由）

スクリプト（eoffsets_monitor.tcl）を使用して、マシンオフ時に axis.L.counts ピンをゼロに設定します。

16.7.2 jwp_z.ini

sim config sim / configs / axis / external_offsets / jwp_z.ini は、単一（Z）座標の一時停止中のジョグ機能を示しています。

パネル LED は、重要なステータス項目を表示するために提供されています。

eoffset スケール係数を設定し、eoffset カウントをインクリメント/デクリメント/クリアするためのコントロールが提供されています。

16.7.3 dynamic_offsets.ini

この simconfig sim / configs / axis / external_offsets / dynamic_offsets.ini は、正弦波形を z 座標の外部オフセット入力に接続することにより、動的に適用されるオフセットを示します。

パネル LED は、重要なステータス項目を表示するために提供されています。

Z 軸の最大速度と最大加速度の ini ファイル設定を変更するためのコントロールが提供されています。

波形ジェネレータのパラメータを設定するためのコントロールが用意されています

halscope アプリが起動して、適用された波形、オフセット応答、およびモーター cmd 応答が表示されます。

注：プログラムの実行中は、z 座標の max-acceleration および max-velocity への変更は確認されません。

16.7.4 opa.ini（eoffset_per_angle）

opa.ini 構成では、hal コンポーネント eoffset_per_angle（$ man eoffset_per_angle）を使用して、C 座標（角度）から計算され、トランスバー（X）座標に適用される機能オフセットを持つ XZC マシンを示します。 オフセットの計算は、通常、motion.analog-out-NN ピンを制御するプログラム（または MDI）M68 コマンドによって設定される指定された参照半径に基づいています。

パネル LED は、重要なステータス項目を表示するために提供されています。

関数は、内側と外側のポリゴン（nsides> = 3）、正弦波、および方形波に対して提供されます。 関数は、fmul ピンを使用して周波数を乗算し、rfrac ピン（参照半径の分数）を使用して振幅を変更できます。

オフセット波形を開始/停止し、機能タイプとそのパラメータを設定するためのコントロールが用意されています。

17章　スタンドアロンインタープリター

rs274 スタンドアロンインタプリタは、コマンドラインから使用できます。

17.1　使用法

```
Usage: rs274 [-p interp.so] [-t tool.tbl] [-v var-file.var] [-n 0|1|2]
[-b] [-s] [-g] [input file [output file]]
-p: Specify the pluggable interpreter to use
-t: Specify the .tbl (tool table) file to use
-v: Specify the .var (parameter) file to use
-n: Specify the continue mode:
0: continue
1: enter MDI mode
2: stop (default)
-b: Toggle the ’block delete’ flag (default: OFF)
-s: Toggle the ’print stack’ flag (default: OFF)
-g: Toggle the ’go (batch mode)’ flag (default: OFF)
-i: specify the .ini file (default: no ini file)
-T: call task_init()
-l: specify the log_level (default: -1)
```

17.2　例

たとえば、ループの出力を確認するには、次のファイルで rs274 を実行し、ループが終了しないことを確認します。 ループから抜け出すには、Ctrl Z を使用します。例を実行するには、次の 2 つのファイルが必要です。

test.ngc

```
#<test> = 123.352
o101 while [[#<test> MOD 60 ] NE 0]
(debug,#<test>)
#<test> = [#<test> + 1]
o101 endwhile
M2
```

test.tbl

```
T1 P1 Z0.511 D0.125 ;1/8 end mill
```

```
T2 P2 Z0.1 D0.0625 ;1/16 end mill
T3 P3 Z1.273 D0.201 ;#7 tap drill
```

指図

```
rs274 -g test.ngc -t test.tbl
```

18章　著作権の保護期間

19章　GNU フリー文書利用許諾契約書

GNU Free Documentation License Version 1.1, March 2000

1. 前文

　このライセンスの目的は、マニュアル、教科書、またはその他の文書を自由の意味で「無料」にすることです。つまり、商業的または非商業的に、変更の有無にかかわらず、コピーおよび再配布するための効果的な自由をすべての人に保証します。 第二に、このライセンスは、他の人が行った変更に対して責任があるとは見なされない一方で、著者と発行者が自分の作品のクレジットを取得する方法を保持します。

　このライセンスは一種の「コピーレフト」です。つまり、ドキュメントの派生物自体も同じ意味で無料である必要があります。 これは、自由ソフトウェア用に設計されたコピーレフトライセンスである GNU GENERAL PUBLICLICENSE を補完します。

　自由ソフトウェアには無料のドキュメントが必要なため、このライセンスは自由ソフトウェアのマニュアルに使用するために設計されました。無料のプログラムには、ソフトウェアと同じ自由度を提供するマニュアルが付属している必要があります。 ただし、このライセンスはソフトウェアマニュアルに限定されません。 主題や印刷された本として出版されているかどうかに関係なく、あらゆるテキスト作業に使用できます。 このライセンスは、主に指示または参照を目的とする作品に推奨されます。

2. 適用性と定義

　このライセンスは、このライセンスの条件に基づいて配布できるという著作権所有者による通知を含むすべてのマニュアルまたはその他の作品に適用されます。 以下の「ドキュメント」とは、そのようなマニュアルまたは作業を指します。 一般の人々は誰でもライセンシーであり、「あなた」と呼ばれます。

　ドキュメントの「変更されたバージョン」とは、ドキュメントまたはその一部を含む、逐語的にコピーされた、または変更された、および/または別の言語に翻訳されたすべての作品を意味します。

　「二次セクション」とは、ドキュメントの発行者または作成者とドキュメントの全体的な主題（または関連事項）との関係のみを扱い、直接当てはまる可能性のあるものを何も含まない、ドキュメントの名前付き付録またはフロントマターセクションです。 その全体的な主題の中で。 （たとえば、ドキュメントの一部が数学の教科書である場合、二次セクションでは数学を説明できない場合があります。）関係は、主題または関連事項との歴史的関係、または法律、商業、哲学、 それらに関する倫理的または政治的立場。

「不変セクション」とは、ドキュメントがこのライセンスに基づいてリリースされることを示す通知で、タイトルが不変セクションのタイトルとして指定されている特定の二次セクションです。

「表紙のテキスト」は、ドキュメントがこのライセンスに基づいてリリースされることを示す通知に、表紙のテキストまたは裏表紙のテキストとして記載されている特定の短い文章です。

ドキュメントの「透明な」コピーとは、一般の人々が仕様を利用できる形式で表され、一般的なテキストエディタまたは（ピクセルで構成される画像の場合）コンテンツを直接簡単に表示および編集できる、機械で読み取り可能なコピーを意味します。 一般的なペイントプログラムまたは（図面の場合）広く利用可能ないくつかの描画エディタであり、テキストフォーマッタへの入力、またはテキストフォーマッタへの入力に適したさまざまな形式への自動変換に適しています。 読者によるその後の変更を阻止または阻止するようにマークアップが設計されている、その他の点では透過的なファイル形式で作成されたコピーは透過的ではありません。 「透明」でないコピーは「不透明」と呼ばれます。

トランスペアレントコピーに適した形式の例には、マークアップなしのプレーン ASCII、TEXINFO 入力形式、LATEX 入力形式、公開されている DTD を使用した SGML または XML、および人間による変更用に設計された標準準拠の単純な HTML が含まれます。 不透明な形式には、POSTSCRIPT、PDF、独自のワードプロセッサでのみ読み取りおよび編集できる独自の形式、DTD や処理ツールが一般に利用できない SGML または XML、および一部のワードプロセッサで生成されたマシン生成 HTML が含まれます。 出力目的のみ。

「タイトルページ」とは、印刷された本の場合、タイトルページ自体に加えて、このライセンスがタイトルページに表示するために必要な資料を読みやすく保持するために必要な次のページを意味します。 タイトルページがない形式の作品の場合、「タイトルページ」とは、テキストの本文の先頭に先行する、作品のタイトルの最も目立つ外観に近いテキストを意味します。

3. 逐語的コピー

このライセンス、著作権表示、およびこのライセンスがドキュメントに適用されることを示すライセンス通知がすべてのコピーで複製され、他の条件を追加しないことを条件として、商用または非商用を問わず、任意の媒体でドキュメントをコピーおよび配布できます。 このライセンスのものに。 作成または配布するコピーの読み取りまたはさらなるコピーを妨害または制御するために技術的手段を使用することはできません。 ただし、コピーと引き換えに補償を受け入れることはできます。 十分な数のコピーを配布する場合は、セクション 3 の条件にも従う必要があります。

上記と同じ条件でコピーを貸したり、コピーを公開したりすることもできます。

4. 数量でのコピー

100 を超える番号のドキュメントの印刷されたコピーを発行し、ドキュメントのライセンス通知でカバーテキストが必要な場合は、これらすべてのカバーテキストを明確かつ読みやすく記載したカバーでコピーを囲む必要があります。 裏表紙の裏表紙のテキスト。 両方の表紙はまた、あなたがこれらのコピーの発行者であることを明確かつ読みやすく識別する必要があります。 表紙には、タイトルのすべて

の単語が等しく目立つように表示された完全なタイトルを提示する必要があります。 さらに、カバーに他の素材を追加することもできます。 文書のタイトルを保持し、これらの条件を満たす限り、表紙に限定して変更を加えたコピーは、他の点では逐語的なコピーとして扱うことができます。

どちらかの表紙に必要なテキストが多すぎて読みにくい場合は、最初にリストされているテキスト（適度に収まる数）を実際の表紙に配置し、残りを隣接するページに続ける必要があります。

100 を超える番号のドキュメントの不透明なコピーを公開または配布する場合は、各不透明なコピーと一緒に機械可読の透明なコピーを含めるか、各不透明なコピーに、完全な透明なコピーを含む公的にアクセス可能なコンピュータネットワークの場所を記載する必要があります。 パブリックを使用する一般的なネットワークがパブリックスタンダードのネットワークプロトコルを使用して無料で匿名でダウンロードするためにアクセスできる、追加資料のないドキュメント。 後者のオプションを使用する場合は、不透明コピーの大量配布を開始するときに、この透明コピーが最後に配布してから少なくとも 1 年後まで、指定された場所でアクセスできるようにするために、合理的に慎重な手順を実行する必要があります。 その版の不透明なコピー（直接または代理店または小売業者を通じて）を一般に公開します。

多数のコピーを再配布する前に、ドキュメントの作成者に連絡して、ドキュメントの更新バージョンを提供する機会を与えることをお勧めしますが、必須ではありません。

5. 変更

上記のセクション 2 および 3 の条件の下で、ドキュメントの変更バージョンをコピーして配布することができます。ただし、このライセンスに基づいて変更バージョンをリリースし、変更バージョンがドキュメントの役割を果たし、ライセンスの配布および変更を行う場合に限ります。 それのコピーを持っている人への修正版。 さらに、変更バージョンでは次のことを行う必要があります。

A) タイトルページ（および存在する場合は表紙）で、ドキュメントのタイトルおよび以前のバージョンのタイトルとは異なるタイトルを使用します（存在する場合は、ドキュメントの履歴セクションにリストする必要があります）。 そのバージョンの元の発行者が許可を与えている場合は、前のバージョンと同じタイトルを使用できます。

B) タイトルページに、作成者として、変更バージョンの変更の作成に責任を負う 1 人以上の個人またはエンティティを、ドキュメントの少なくとも 5 人の主要な作成者（ドキュメントの主要な作成者が少ない場合はすべて）とともにリストします。 五）。

C) タイトルページに、変更バージョンの発行者の名前を発行者として記載します。

D) ドキュメントのすべての著作権表示を保持します。

E) 他の著作権表示に隣接して、変更に適切な著作権表示を追加します。

F) 著作権表示の直後に、このライセンスの条件に基づいて修正バージョンを使用することを一般に許可するライセンス通知を、以下の補遺に示す形式で含めます。

G) そのライセンス通知に、ドキュメントのライセンス通知に記載されている不変セクションと必要なカバーテキストの完全なリストを保存します。

H) このライセンスの変更されていないコピーを含めます。

I) 履歴」というタイトルのセクションとそのタイトルを保持し、タイトルページに記載されているように、少なくともタイトル、年、新しい著者、および変更バージョンの発行者を示すアイテムを追加します。 ドキュメントに「履歴」というタイトルのセクションがない場合は、タイトルページに記載されているように、ドキュメントのタイトル、年、作成者、発行者を記載したセクションを作成し、前の文で述べたように変更バージョンを説明するアイテムを追加します。

J) ドキュメントの透過コピーへのパブリックアクセスのために、ドキュメントに指定されているネットワークの場所を保持します。同様に、以前のバージョンのドキュメントに指定されているネットワークの場所を保持します。 これらは「履歴」セクションに配置できます。 ドキュメント自体の少なくとも 4 年前に公開された作品のネットワーク上の場所を省略できます。または、ドキュメントが参照するバージョンの元の発行者が許可を与えている場合は省略できます。

K) 「謝辞」または「献身」というタイトルのセクションでは、セクションのタイトルを保持し、そこに記載されている各寄稿者の謝辞および/または献身の内容とトーンをすべてセクションに保存します。

L) ドキュメントのすべての不変セクションを、テキストとタイトルを変更せずに保持します。 セクション番号または同等のものは、セクションタイトルの一部とは見なされません。

M) 「承認」というタイトルのセクションを削除します。 このようなセクションは、変更バージョンに含まれていない場合があります。

N) 既存のセクションのタイトルを「承認」に変更したり、不変セクションとタイトルを競合させたりしないでください。

変更バージョンに、セカンダリセクションとして適格であり、ドキュメントからコピーされた資料が含まれていない新しいフロントマターセクションまたは付録が含まれている場合、オプションでこれらのセクションの一部またはすべてを不変として指定できます。 これを行うには、変更バージョンのライセンス通知の不変セクションのリストにタイトルを追加します。 これらのタイトルは、他のセクションタイトルとは異なる必要があります。

「承認」というタイトルのセクションを追加できます。ただし、ピアレビューのステートメントや、テキストが標準の信頼できる定義として組織によって承認されているなど、さまざまな関係者による変更バージョンの承認のみが含まれている必要があります。

修正版の表紙テキストのリストの最後に、表紙テキストとして最大 5 語、裏表紙テキストとして最大 25 語のパッセージを追加できます。 フロントカバーテキストの 1 つのパッセージとバックカバーテキストの 1 つだけが、任意の 1 つのエンティティによって（またはそれによって行われた取り決めを通じて）追加できます。 ドキュメントに同じ表紙の表紙テキストがすでに含まれている場合、以前にあなたが追

加したか、あなたが代理を務める同じエンティティによって行われた取り決めによって追加された場合、別の表紙を追加することはできません。 ただし、古いものを追加した前の発行元からの明示的な許可があれば、古いものを置き換えることができます。

ドキュメントの作成者および発行者は、このライセンスにより、変更されたバージョンの宣伝、または承認を主張または暗示するために自分の名前を使用することを許可しません。

6. ドキュメントの組み合わせ

変更されていないすべての元のドキュメントのすべての不変セクションを組み合わせに含め、それらをすべてリストすることを条件として、変更されたバージョンについて上記のセクション 4 で定義された条件の下で、このライセンスの下でリリースされた他のドキュメントとドキュメントを組み合わせることができます。 ライセンス通知の結合された作業の不変セクションとして。

結合された作業には、このライセンスのコピーが 1 つだけ含まれている必要があり、複数の同一の不変セクションを 1 つのコピーに置き換えることができます。 同じ名前で内容が異なる複数の不変セクションがある場合は、そのセクションの最後に、そのセクションの元の作成者または発行者の名前（わかっている場合）を括弧で囲んで追加することにより、そのような各セクションのタイトルを一意にします。 一意の番号。 結合された作業のライセンス通知の不変セクションのリストのセクションタイトルに同じ調整を行います。

組み合わせでは、さまざまな元のドキュメントの「履歴」というタイトルのセクションを組み合わせて、「履歴」というタイトルの 1 つのセクションを形成する必要があります。 同様に、「謝辞」というタイトルのセクションと「献身」というタイトルのセクションを組み合わせます。 「承認」というタイトルのすべてのセクションを削除する必要があります。

7. 文書のコレクション

このライセンスに基づいてリリースされたドキュメントおよびその他のドキュメントで構成されるコレクションを作成し、このライセンスの規則に従うことを条件として、さまざまなドキュメント内のこのライセンスの個々のコピーを、コレクションに含まれる単一のコピーに置き換えることができます。 他のすべての点での各文書の逐語的なコピー。

このようなコレクションから単一のドキュメントを抽出し、抽出したドキュメントにこのライセンスのコピーを挿入し、そのドキュメントの逐語的なコピーに関する他のすべての点でこのライセンスに従うことを条件として、このライセンスの下で個別に配布することができます。

8. 独立した作品との集約

ストレージまたは配布メディアのボリューム内またはボリューム上での、他の別個の独立したドキュメントまたは作品とのドキュメントまたはその派生物の編集は、編集の著作権が主張されていない限り、ドキュメントの修正バージョンとして全体としてカウントされません。 コンパイル。 このような編集は「集約」と呼ばれ、このライセンスは、それ自体がドキュメントの派生物でない場合、このように編集されているため、ドキュメントとともに編集された他の自己完結型の作品には適用されません。

セクション 3 のカバーテキスト要件がドキュメントのこれらのコピーに適用される場合、ドキュメントがアグリゲート全体の 4 分の 1 未満である場合、ドキュメントのカバーテキストは、アグリゲート内のドキュメントのみを囲むカバーに配置できます。 それ以外の場合は、骨材全体のカバーに表示する必要があります。

9. 翻訳

   翻訳は一種の変更と見なされるため、セクション 4 の条件に基づいてドキュメントの翻訳を配布できます。不変セクションを翻訳に置き換えるには、著作権所有者からの特別な許可が必要ですが、一部またはすべての不変セクションの翻訳を含めることができます。 これらの不変セクションの元のバージョン。 このライセンスの元の英語版も含めることを条件に、このライセンスの翻訳を含めることができます。 翻訳と本ライセンスの元の英語版との間に不一致がある場合は、元の英語版が優先されます。

10. 終了

    このライセンスで明示的に規定されている場合を除き、ドキュメントをコピー、変更、サブライセンス、または配布することはできません。 ドキュメントをコピー、変更、サブライセンス、または配布するその他の試みは無効であり、このライセンスに基づくお客様の権利は自動的に終了します。 ただし、このライセンスに基づいてお客様からコピーまたは権利を受け取った当事者は、そのような当事者が完全に準拠している限り、ライセンスが終了することはありません。

11. このライセンスの将来の改訂

    FREE SOFTWARE FOUNDATION は、GNU FREE DOCUMENTATIONLICENSE の新しい改訂版を随時公開する場合があります。 このような新しいバージョンは、精神的には現在のバージョンと似ていますが、新しい問題や懸念に対処するために詳細が異なる場合があります。

    HTTP://WWW.GNU.ORG/COPYLEFT/を参照してください。

    ライセンスの各バージョンには、識別可能なバージョン番号が付けられています。 このライセンスの特定の番号付きバージョン「またはそれ以降のバージョン」が適用されることをドキュメントが指定している場合、その指定されたバージョンまたは公開された後のバージョンのいずれかの契約条件に従うオプションがあります（ ドラフト）フリーソフトウェアファウンデーションによる。 ドキュメントにこのライセンスのバージョン番号が指定されていない場合は、FREE SOFTWARE FOUNDATION によってこれまでに公開された（ドラフトとしてではなく）任意のバージョンを選択できます。

    補遺：このライセンスをドキュメントに使用する方法

    あなたが書いた文書でこのライセンスを使用するには、文書にライセンスのコピーを含め、次の著作権を入れてください タイトルページの直後のライセンス通知：

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".

不変セクションがない場合は、どのセクションが不変であるかを言う代わりに、「不変セクションなし」と書いてください。 表紙のテキストがない場合は、「表紙のテキストがリストになっている」の代わりに「表紙のテキストがない」と書いてください。 裏表紙のテキストについても同様です。

ドキュメントに重要なプログラムコードの例が含まれている場合は、GNU GENERAL PUBLIC LICENSE などの自由ソフトウェアライセンスを選択してこれらの例を並行してリリースし、自由ソフトウェアでの使用を許可することをお勧めします。