サーボコントローラ

# ＳＣ２０００シリーズ用サポートツール

## 日本語版ソフト簡易説明書

ScTool v2.0.9.7


株式会社　ワコー技研

・目次

## １．はじめに

このたび、汎用コントローラＳＣ２０００シリーズサポートツールをお買い上げいただきまして、有難うございます。

ご使用の前に、この取扱説明書を良くお読みいただき、本システムを正しくご利用ください。

＊このセットの構成は下記の通りです。

ＳＣ２０００サポートツールインストールＣＤ　１枚
インストール説明書（本書）１冊
メニュー概略説明書　１冊

＊本ソフトウェアが動作するハードウェア構成は、下記の通りです。

ＩＢＭ－ＰＣ／ＡＴ互換ハードウェア
（INTEL又はAMD製CPU搭載機種）

RS232C通信ポート
（ＲＳ２３２Ｃ通信ポートが無いＰＣの場合、USB-SERIAL変換器が御利用になれます）

800x600 ピクセル表示が可能なVGAカード（オンボード可）及び表示装置
（推奨：１０２４ｘ７６８ピクセル）

CD-ROMドライブ
（ソフトインストール時のみ使用します）

ハードディスクに 10MBytes以上の空き容量

＊本ソフトウェアは下記ＯＳで動作致します。

Ｗｉｎｄｏｗｓ９８（制限有り(*1)）
Ｗｉｎｄｏｗｓ９８ＳＥ（制限有り(*1)）
Ｗｉｎｄｏｗｓ２０００ Ｐｒｏｆｅｓｓｉｏｎａｌ
ＷｉｎｄｏｗｓＸＰ　Ｐｒｏ／Ｈｏｍｅ

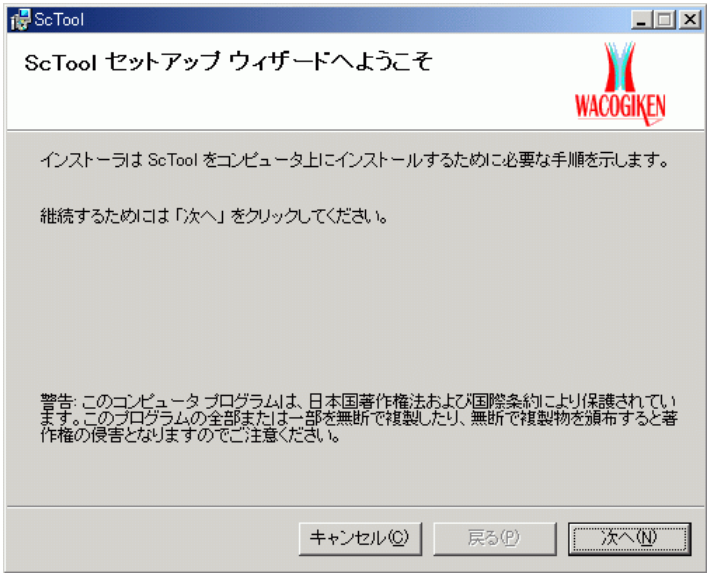
他のＯＳでは正しくご利用になれませんので、御注意下さい。
（エミュレータ上では上記動作保証ＯＳでも、動作保証外とさせて頂きます）

(*1)最大プログラムサイズが 32KBytesで制限されます。
Windows2000/XPに、この制限は有りません。

## 2．インストール

ＳＣ２０００シリーズサポートツール（ＳｃＴｏｏｌ）をインストールするには以下の操作を行います。
（注意！！ 既にインストールされているシステムに新しいバージョンをインストールする場合は、
先にアンインストールを行ってください）

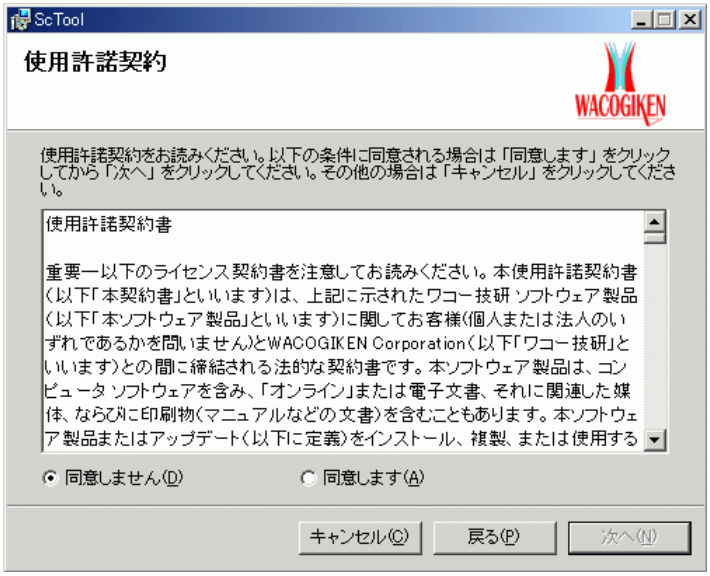
・ＳＣ２０００シリーズサポートツールのＣＤ－ＲＯＭをＣＤ－ＲＯＭドライブに挿入します。
インストーラが自動的に起動し、下記画面が表示されます。




もし表示されない場合は，ＣＤ－ＲＯＭのルートディレクトリにあるＳＥＴＵＰ．ＥＸＥをエクスプローラ等からダブルクリックして起動が可能です。

お使いのシステムに「WindowsInstaller」がインストールされていない場合、上記画面と違う場合があります。この場合は画面の指示に従ってください。

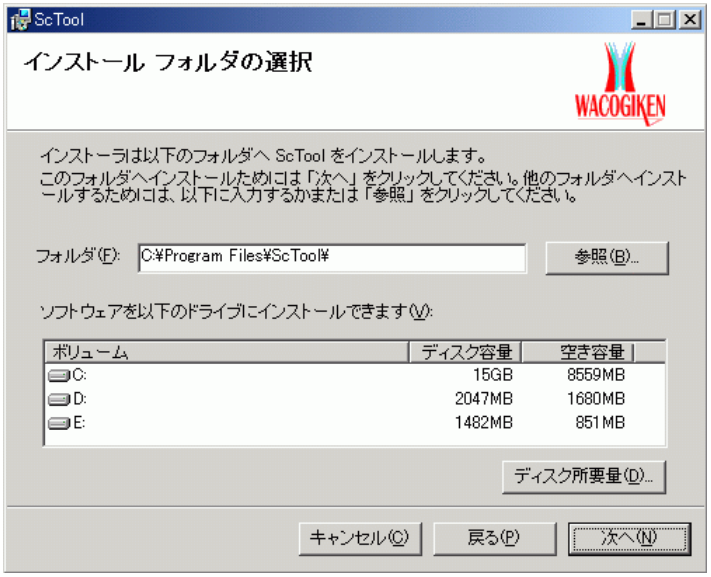
「次へ(N)」を押すと次の画面に進みます。




使用許諾契約画面が表示されます。良くお読みになって、同意される場合は「同意します(A)」をクリックしてください。同意されない場合には、インストールを継続することは出来ませんので、「キャンセル」を押してインストールを中止してください。

「同意します(A)」をクリックすると「次へ(N)」のボタンが押せるようになりますので、押して次の画面に移動してください。

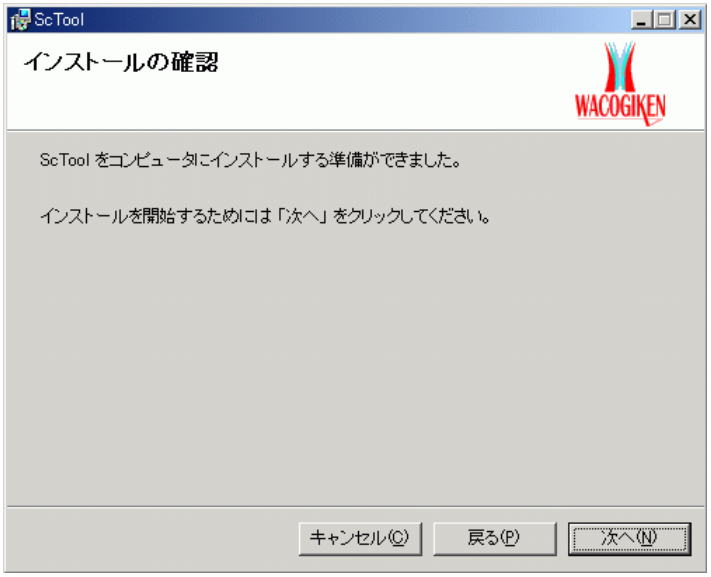
## 2．インストール（続き）




インストール先フォルダ選択画面が表示されます。お使いのシステムに搭載されているハードディスクの容量及び空き容量が表示されます。
（ハードディスクの表示容量はシステムに依存する為、上記画面と異なります）

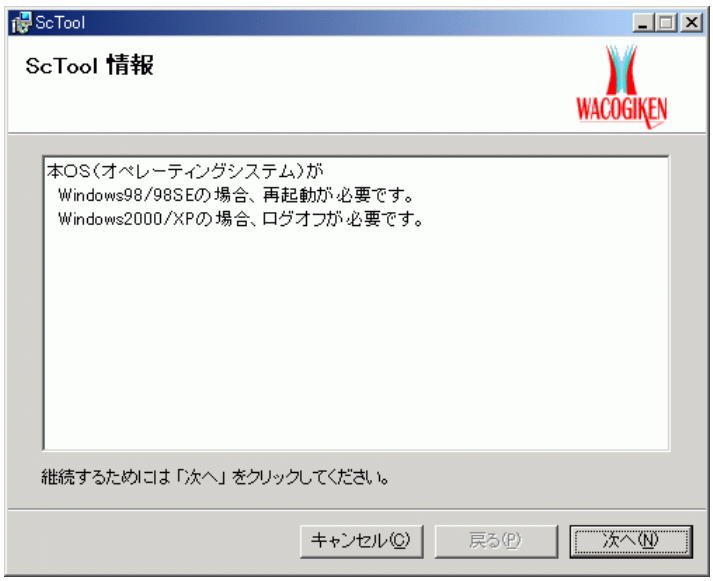
特に問題が無い限り、ディレクトリの変更は行わない事を推奨します。

「次へ(N)」を押すと次の画面に進みます。



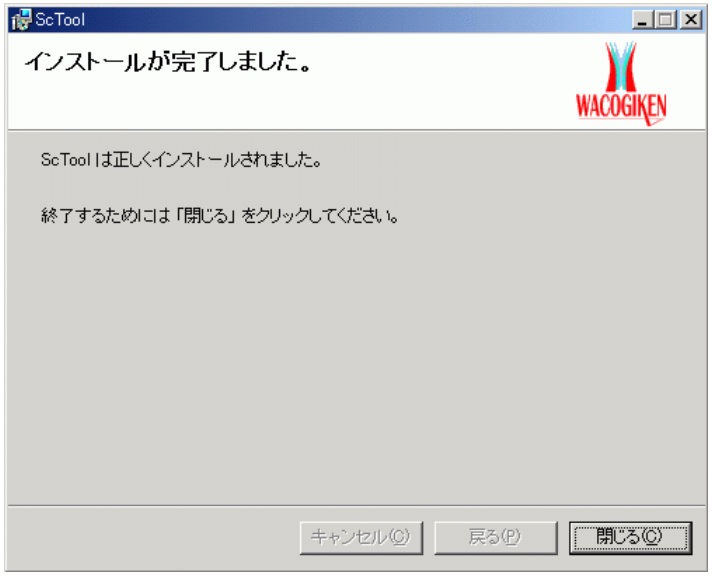

インストールの準備が出来ました。
「次へ(N)」を押すとインストールを開始します。

## 2．インストール（続き）




インストール途中で、注意点が表示されます。
初回のインストール時に限り上記注意書きに従う必要があります。
（動作しているＯＳがWindows98 系の場合は再起動、Windows2000 系の場合はログオフが必要になります）

「次へ(N)」を押すと次の画面を表示します。




上図画面が表示されましたらインストールは終了しました。「閉じる(C)」を押してインストーラを終了させてください。

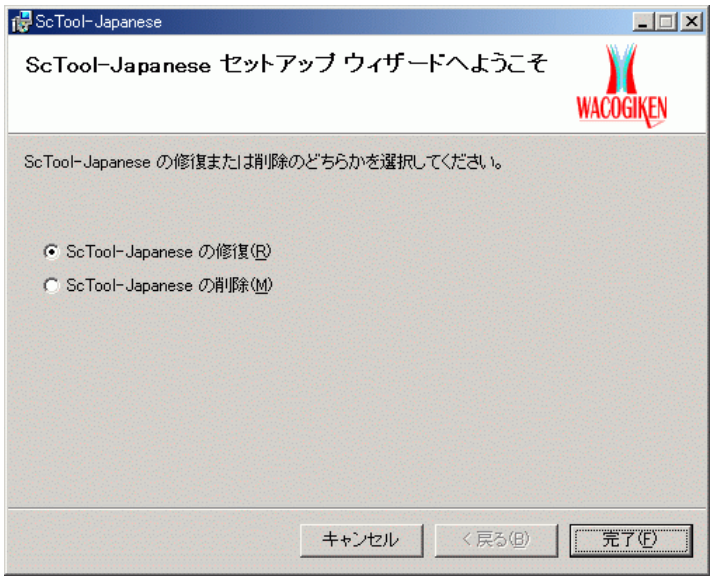
終了後、初回のインストールに限り、前ページの注意書きに従って下さい。
（2 回目以降は必要ありません）

## ３．アンインストール

本ＳｃＴｏｏｌが不要となった場合に、アンインストールする方法です。

ＳＣ２０００シリーズサポートツールのＣＤ－ＲＯＭをＣＤ－ＲＯＭドライブに挿入します。
ＣＤ－ＲＯＭの自動起動機能が有効であれば、数秒で下記画面が表示されます。

表示されない場合は、ＣＤ－ＲＯＭのルートディレクトリにあるＳＥＴＵＰ．ＥＸＥを
エクスプローラ等からダブルクリックして起動して下さい。

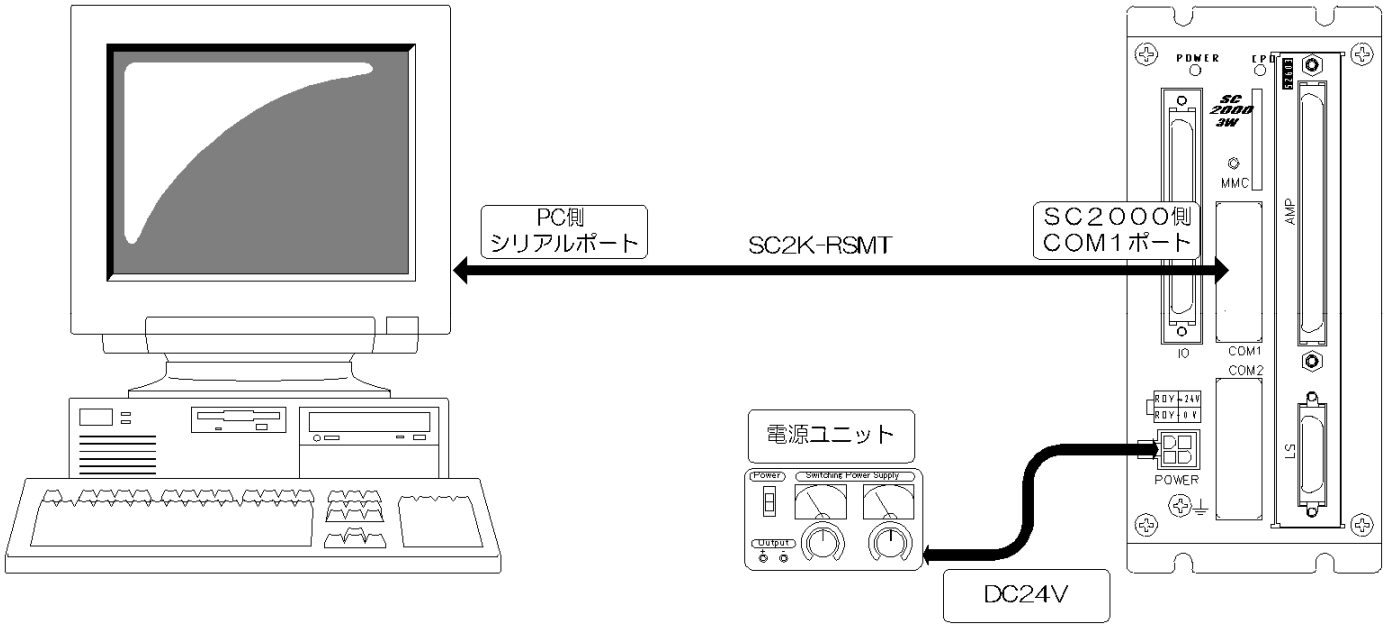



「ＳｃＴｏｏｌ－Ｊａｐａｎｅｓｅの削除(M)」にチェックをつけて、「完了(F)」を押せば
アンインストールが開始されます。

コントロールパネルの「アプリケーションの追加と削除」から「ScTool-Japanese」を選択しても
同じ事が行えます。

## 4．ハードウェア

・最小構成での接続図




最小構成の接続に必要なハードウェアは、下記の通りです。
- ・ＳＣ２０００本体
- ・ＳＣ２Ｋ－ＲＳＭＴケーブル
- ・ＤＣ２４Ｖの安定化電源

ＰＣにシリアルポートが無い場合、ＵＳＢ－SERIAL変換アダプタ(*1)を使用する事により、上記と同等な接続が可能です。（ＰＣ本体にＵＳＢポートが必要となります）

(*1)USBポートを１つ使い、シリアルポートとして使用出来る変換アダプタです。
ATEN社(*2)等から発売されています。

(*2)http://www.aten.com/

## 5．メニュー

本ツールを起動すると、下記のメニューが表示されます。
各メニューの詳細は、ツールのオンラインヘルプ機能を参照願います。
本説明書では概略のみ説明致します。

・ファイルメニュー


このモードでは、下記の操作が行えます。

・レジスタ・プログラム等のウィンドウを開いて編集（新規作成）

ＳＣ２０００の内部レジスタ（又はプログラム）を編集する為に、目的のウィンドウを開く事が出来ます。

・ワークスペース（作業ディレクトリ）の作成／変更／設定

バックアップ・リストア等に使用されるディレクトリを指定します。
ファイルの保存／読込にも使用されます。

・プリンタの設定／印刷（レジスタ・プログラム類のウィンドウを開いている場合）

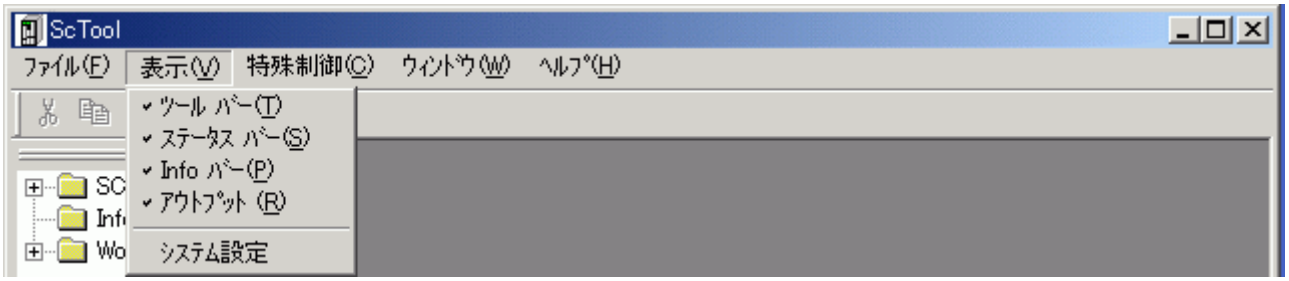
レジスタ／プログラムを印刷する為のメニューです。

・ソフトの終了

本ソフトウェアを終了します。

5．メニュー（続き）

・表示メニュー




このモードでは、下記の操作が行えます。

・各種ツールバー（又は情報ウィンドウ）の表示／非表示設定

各ツールバー（又は情報ウィンドウ）の表示／非表示を設定します。
閉じてしまったウィンドウは、このメニューから表示させる事が可能です。

・システム設定

本ソフトウェアの動作設定が行えます。

５．メニュー（続き）

・特殊制御メニュー


このモードでは、下記の操作が行えます。

・ＳＣ２０００との回線接続／回線切断／回線設定

御使いのＰＣとＳＣ２０００の通信を開始／終了します。
レジスタの編集やプログラムの書き込み時には、回線接続が必須となります。
通信手段（ＲＳ２３２Ｃ／その他）の設定もこのメニューから行えます。

・ＳＣ２０００のデータをバックアップ／ＳＣ２０００へデータをリストア

接続されているＳＣ２０００から全てのデータ(*1)をバックアップ、又は
バックアップされているデータをＳＣ２０００にリストアする為のメニュー
です。
(*1)ファームウェア以外

・フラッシュＲＯＭ操作（保存／呼び出し）

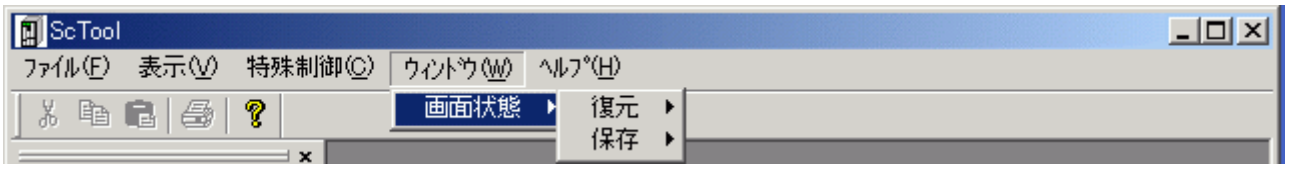
ＳＣ２０００のデータは起動時にＲＡＭ（揮発性メモリー）に転送され，これに
対しての編集が行えます。　しかし電源を切ると編集内容は全て破棄されてしまうので、
搭載しているフラッシュＲＯＭ（不揮発性メモリー）に対して書き込む事により、
レジスタ、プログラムの保存を行えます。
逆に、起動中のＳＣ２０００のレジスタ、又はプログラムを電源投入時に戻したい場合は、
呼び出しも行えます。

・特殊機能（消去／再起動／モーション動作状態閲覧）

特定のレジスタ、プログラムの消去が行えます。（消去）
ＳＣ２０００の電源を再投入した状態に戻す事が可能です。（再起動）
現在動作しているモーションプログラムの状況が閲覧出来ます（モーション動作状態閲覧）

5．メニュー（続き）

・ウィンドウメニュー

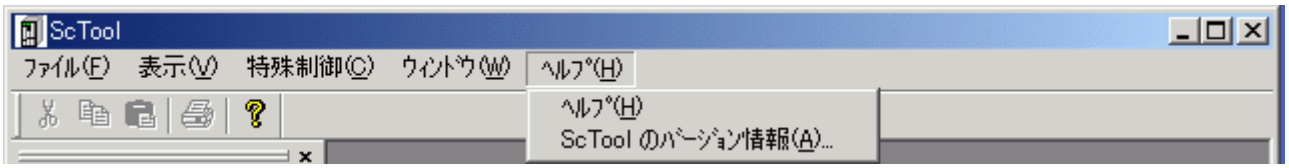



このモードでは、下記の操作が行えます。

・ウィンドウ状態の復元／保存

ウィンドウを毎回大量に開いていたのでは、編集の作業性効率が落ちる場合が有ります。
その様な場合に、「良く使うウィンドウ配列」を記憶しておく事が可能です。
（最大４パターン）
記憶された状態は、呼び出す事により「保存時のウィンドウ状態」に戻ります。
（戻るのはウィンドウ状態のみで，データが戻る訳では有りません）

・ヘルプメニュー




このモードでは、下記の操作が行えます。

・オンラインヘルプ閲覧

ＳｃＴｏｏｌのヘルプは基本的にオンラインマニュアルのみとなっております。
プログラム作成時に必要な関数リファレンス、メニューの構成、その他情報等から
構成されています。

・本ソフトのバージョン情報表示

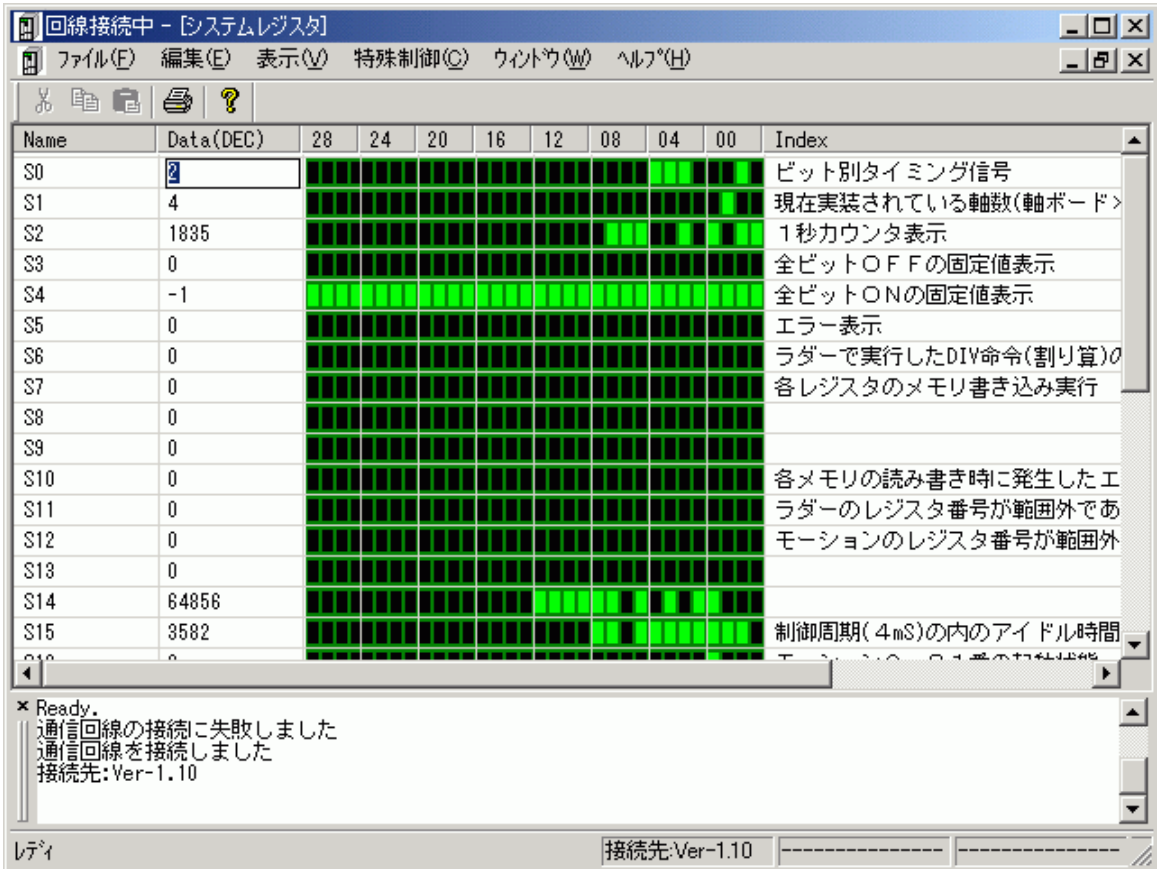
本ＳｃＴｏｏｌのバージョン番号が閲覧出来ます。

・ＳＣ２０００本体ファームウェアのアップデート

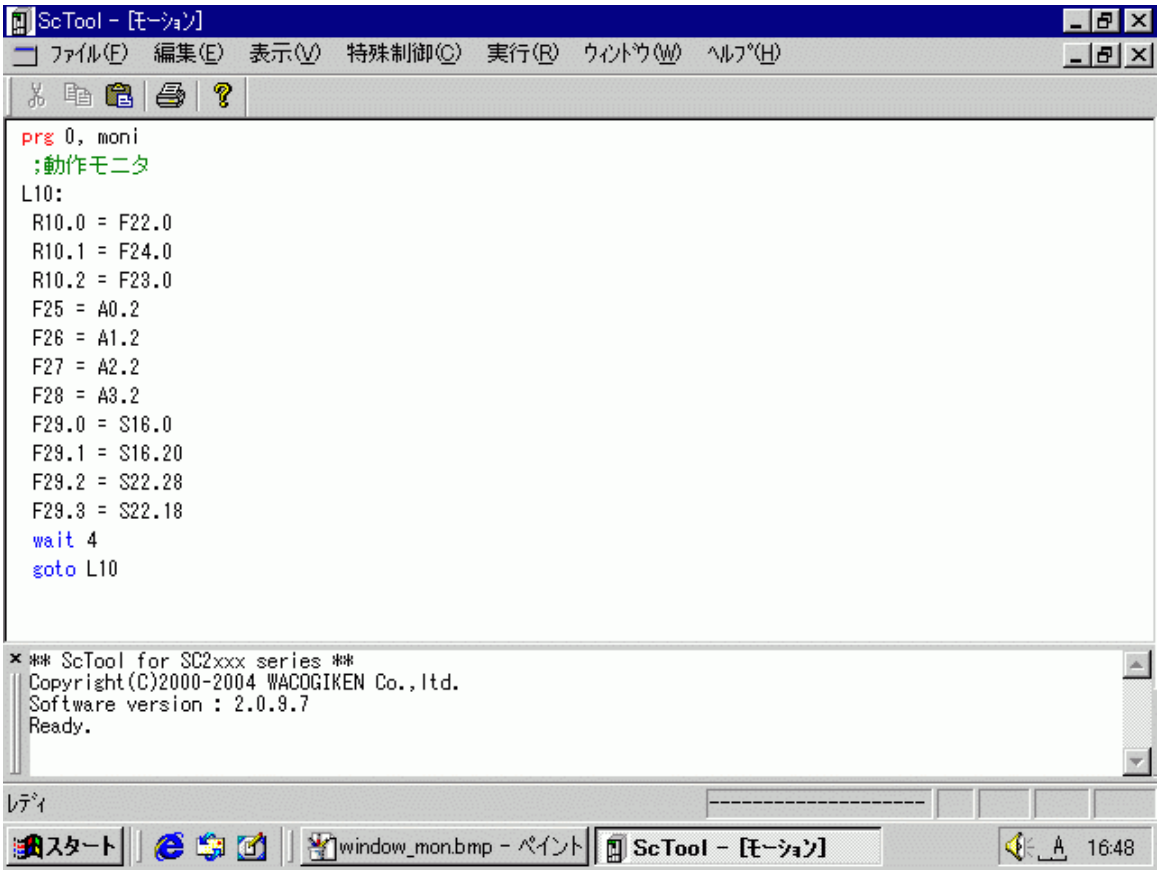
ＳＣ２０００本体のファームウェアをアップデート可能です。
ファームウェア自体は、当社から御送りするファイルか、インターネット経由での取得が
行えます。（インターネットからの取得には、その環境から当社ホームページが閲覧出来る
必要があります）

## 6．画面イメージ

・レジスタ編集中

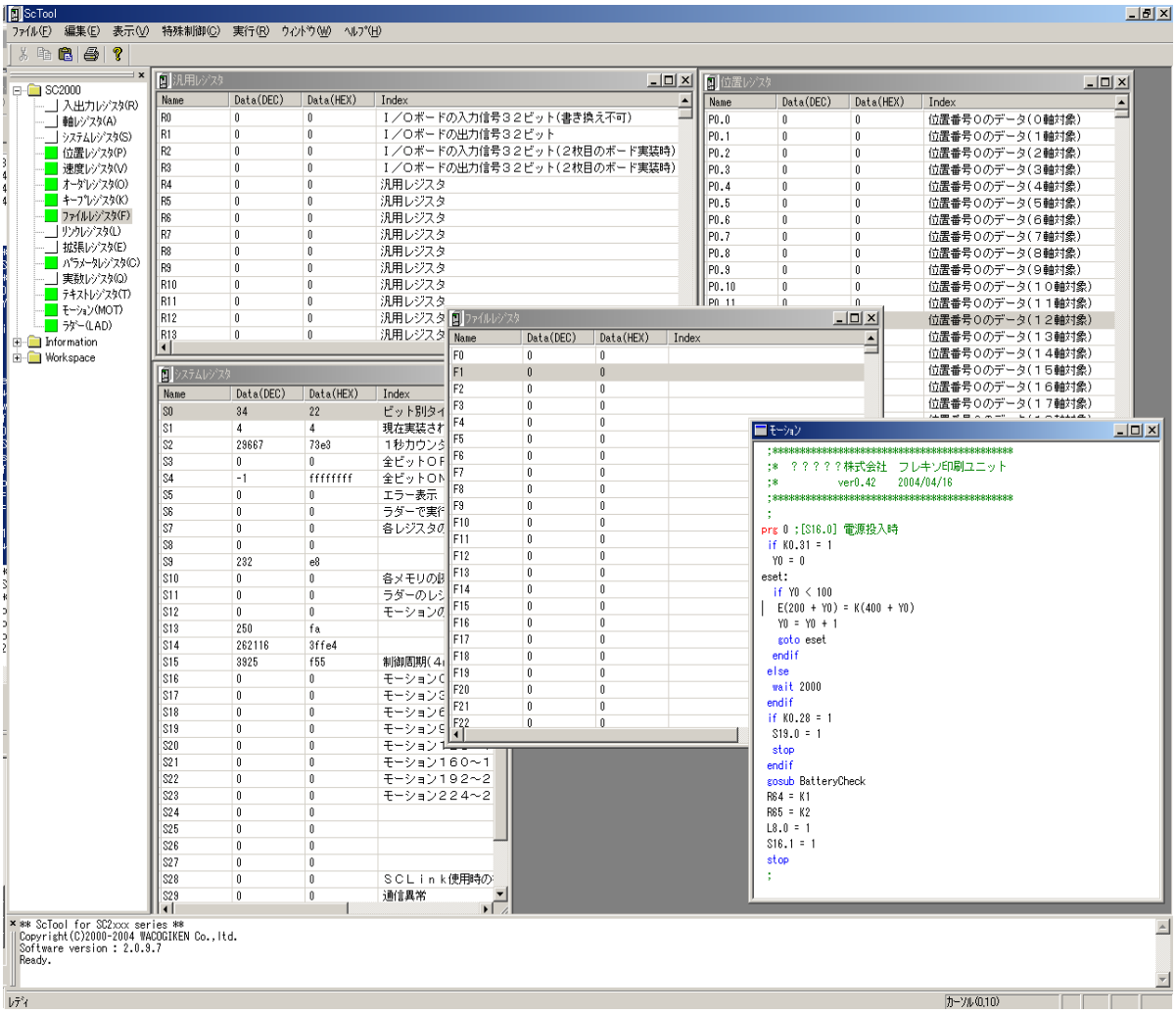



・プログラム（モーション）編集中



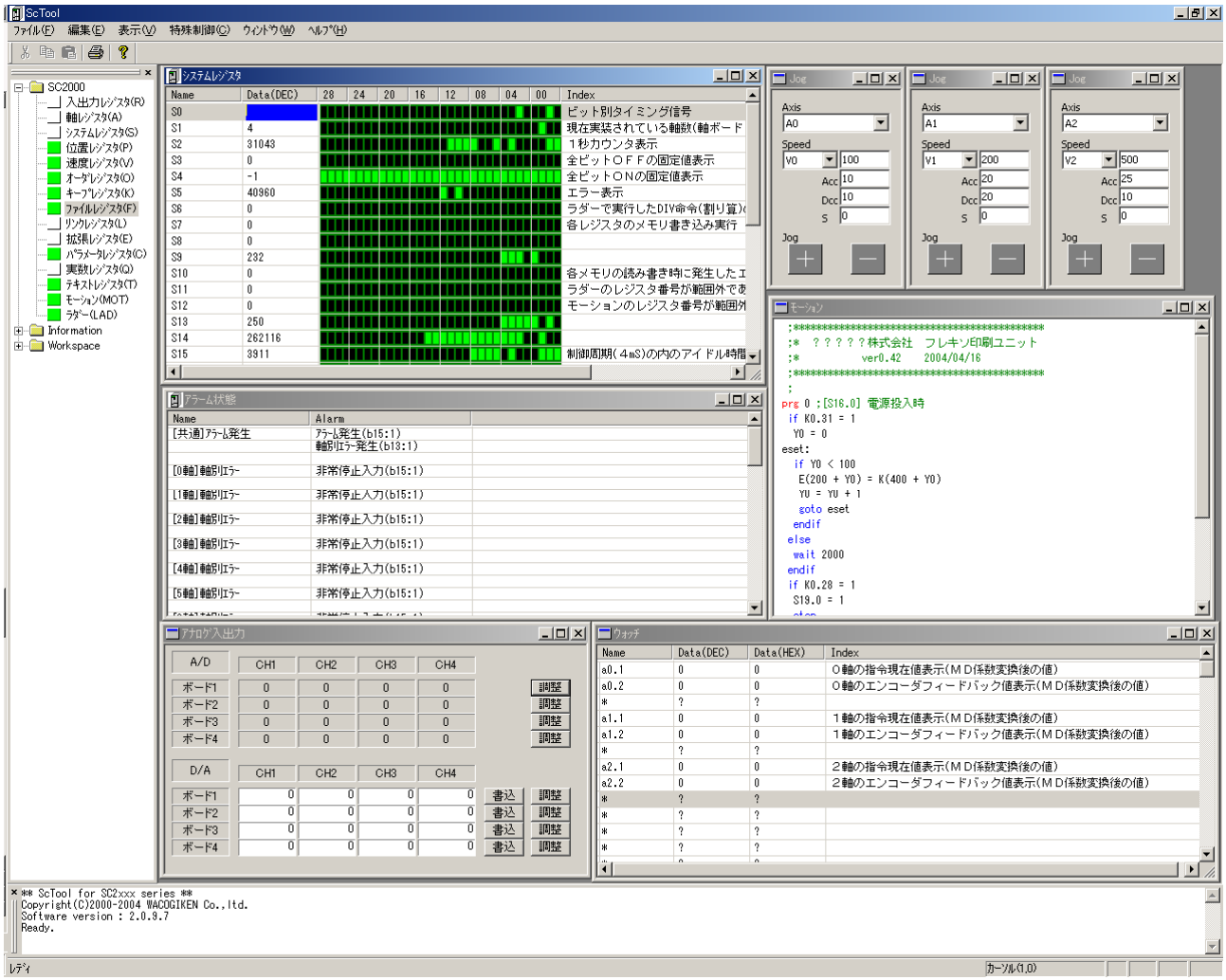
# ６．画面イメージ（続き）

・マルチウィンドウでの編集中画面１



# 6．画面イメージ（続き）

・マルチウィンドウでの編集中画面２



## ７．参考付録　Ｈｅｌｐファイル内容

　次ページ以降は、ソフトウェアに付属のオンラインＨｅｌｐの内容を印刷した物です。複数ページの同時参照などにお使い下さい。

# 概要

本ツールの概要
本ツールは「SC2xxxシリーズ」のサポート用ツールです。
（現時点では、SC2000のみが対象）
本ツールを使用する事により、「SC2xxxシリーズ」上で動作するプログラムの開発等を行う事が可能になります。

# 動作環境

ハードウェア

| 項目 | 内容 |
|---|---|
| パソコン種類 | PC／AT互換機 |
| CPU | PentiumII-300MHz相当以上の物<br>（推奨：PentiumIII-700MHz相当以上の物） |
| メモリー | 128MBytes以上<br>（推奨：OS起動後に空きメモリーが128M以上） |
| ビデオ | 800x600ピクセル以上<br>（推奨：1024x768ピクセル以上） |
| CDROM | 2倍速以上（インストール時のみ使用） |
| 通信ポート | RS232Cポートを1つ使用<br>USB-RS232C変換でも動作可 |
| ハードディスク | 10MBytesの空き領域 |

ソフトウェア

| 項目 | 内容 |
|---|---|
| オペレーティングシステム<br>（OS） | Windows98（非推奨）<br>Windows98SE（非推奨）<br>Windows Me（非推奨）<br>Windows2000Pro（推奨環境）<br>Windows2000Server(動作可)<br>Windows2000AdvancedServer(動作可)<br>WindowsXP home-edition（推奨環境）<br>WindowsXP Professional-edition（推奨環境）<br><br>上記OS の何れかが正常稼動している環境<br>（エミュレータ（ ※1）上での動作保証は致しかねます）<br>注意！Windows95では動作しません |

（※1）VMware/VirtualPC2004等のPC/ATエミュレータ

# オプションモジュール（拡張機能）

オプションモジュール（拡張機能）

| 型番 | 名称 | 内容 |
|---|---|---|
| （提供中） | 拡張JOG運転DLL | 複数軸（最大4軸）同時運転が可能なJOG運転用拡張モジュール |
| （作成中/BETA版提供中） | モーションウィザード拡張DLL | モーション作成時、駆動命令を簡単に入力出来る拡張モジュール |
| （要問い合わせ） | イーサネット通信DLL | SC2000とイーサネット経由で通信を行う為の拡張モジュール<br>SC2000用イーサネット通信基板に付属します |
| （作成中） | 軸レジスタ制御拡張DLL | 軸レジスタに対して、サーボ制御、各種簡易運転を行う為の拡張モジュール |
| （作成中） | 簡易エミュレータ拡張DLL | SC2000実機が無い環境でのプログラム・レジスタ作成用の拡張モジュール（実機と完全同一ではありません） |

オプションモジュール（拡張機能）のインストール方法

ScTool本体のインストール先ディレクトリ（規定値：C:¥Program files¥ScTool）の下に Pluginディレクトリがあるので、
そこに拡張機能のDLLをコピーし、ScTool本体を起動しなおせば完了となります。

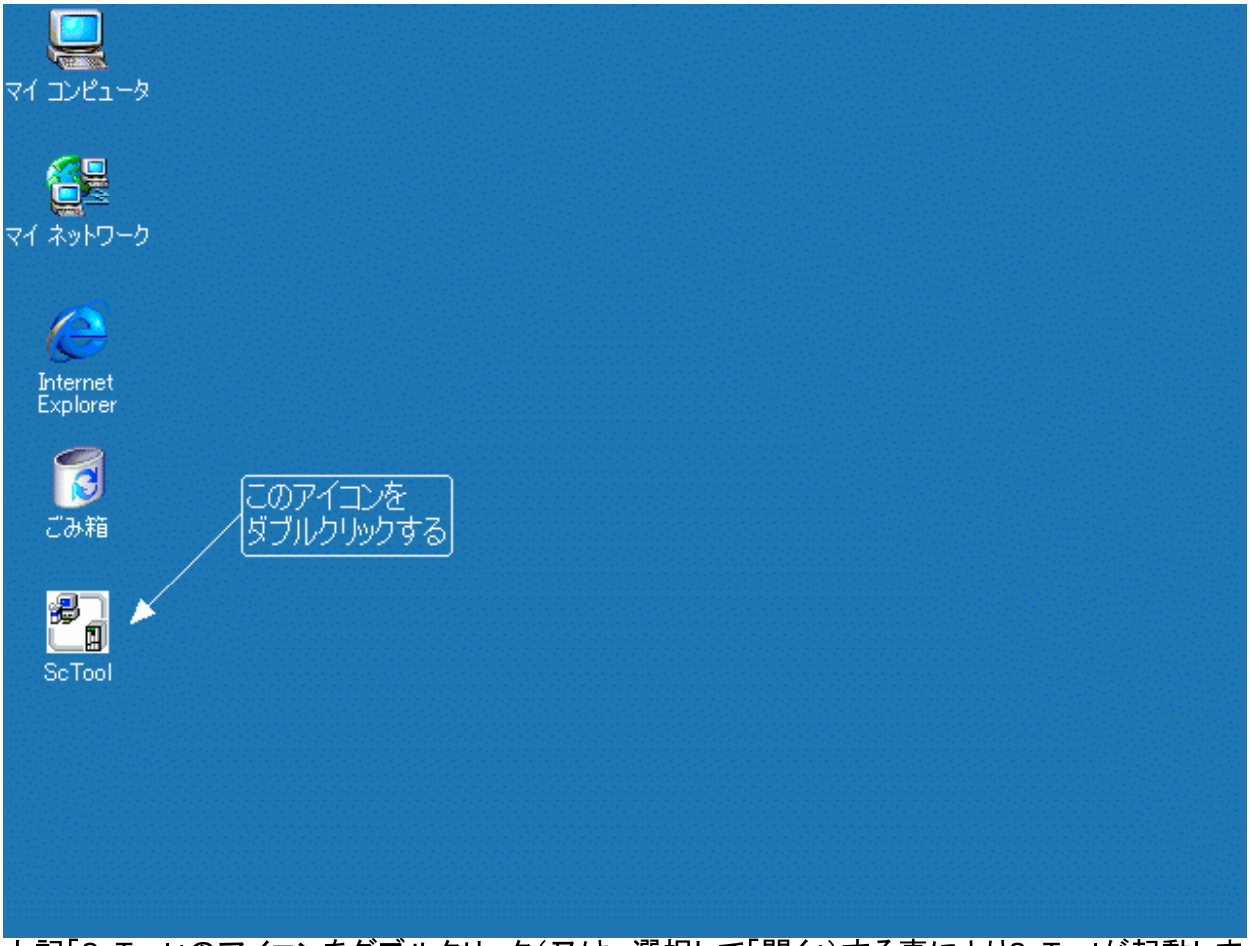
幾つかの拡張機能には設定項目が有り、ScToolの「表示(V)」→「システム設定」→「DLL」タブ画面で設定する事が可能です
（設定が無い拡張機能もあります）

又、導入する事によりメニューの項目が変化する場合も有りますので、各拡張機能のマニュアルを参照願います。

# 起動方法

---

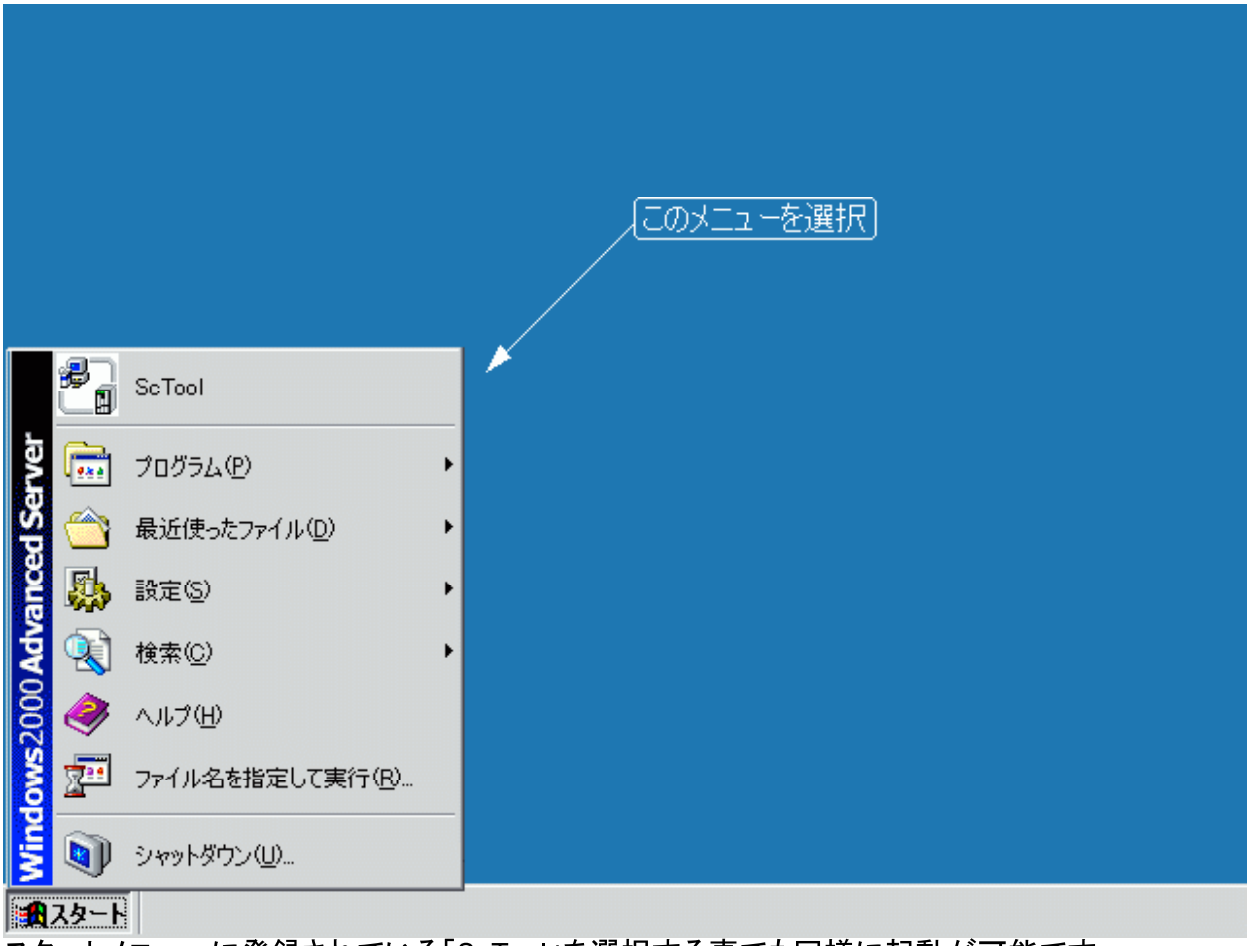
起動方法1
1. デスクトップのアイコンから起動




上記「ScTool」のアイコンをダブルクリック（又は、選択して「開く」）する事によりScToolが起動します。

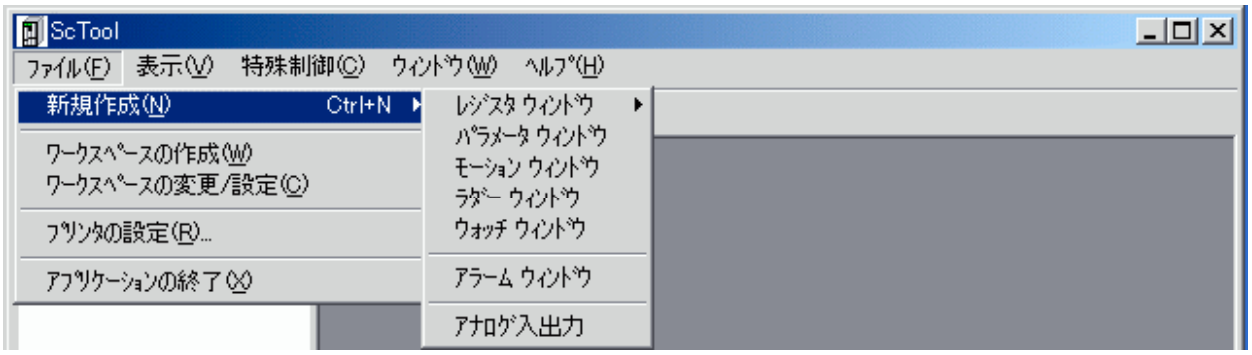
---

起動方法2
2. スタートメニューから起動




スタートメニューに登録されている「ScTool」を選択する事でも同様に起動が可能です。

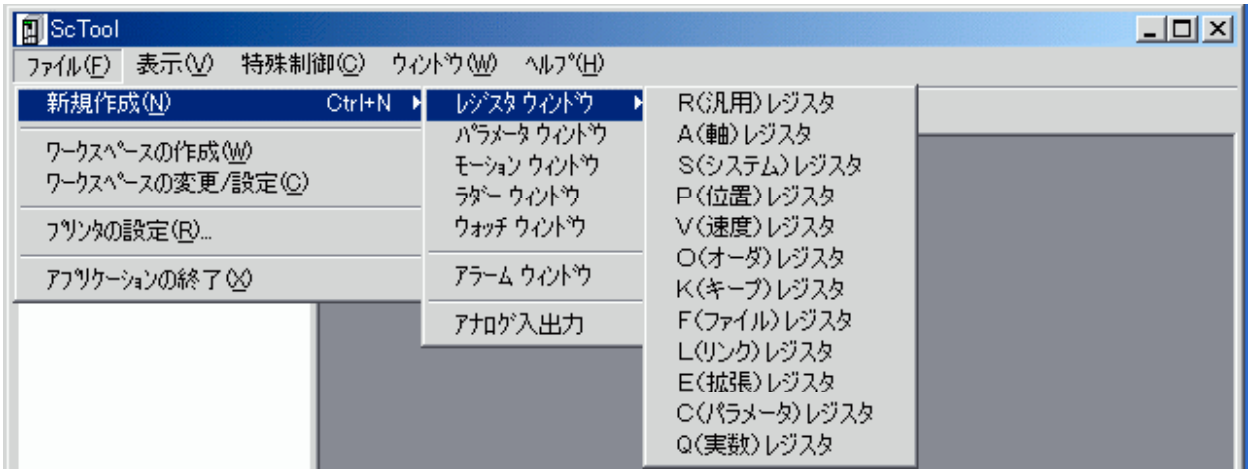
# ファイル(F)・新規作成(N)




このメニューには、下記7点の項目があります
（下記項目をクリックすると、詳細説明に移動します）


レジスタ ウィンドウ
パラメータ ウィンドウ
モーション ウィンドウ
ラダー ウィンドウ
ウォッチ ウィンドウ
アラーム ウィンドウ
アナログ入出力


# ファイル(F)・新規作成(N)・レジスタウィンドウ




このメニューは、指定されたレジスタのウィンドウ（以下レジスタウィンドウ）を開きます。
開く事が出来るレジスタウィンドウは下記の通りです。

R（汎用）レジスタ
　汎用レジスタ（入出力レジスタ及び汎用として使用出来るレジスタ）のウィンドウを開きます

A（軸）レジスタ
　軸レジスタ（各軸の制御・状態等のレジスタ）のウィンドウを開きます

S（システム）レジスタ

システムレジスタ（SC2xxxの制御・状態等のレジスタ）のウィンドウを開きます

Ｐ（位置）レジスタ

位置レジスタ（各軸で使用する位置情報が格納されたレジスタ）のウィンドウを開きます
（このレジスタはフラッシュROMに保存が可能です）

Ｖ（速度）レジスタ

速度レジスタ（各軸で使用する速度情報が格納されたレジスタ）のウィンドウを開きます
（このレジスタはフラッシュROMに保存が可能です）

Ｏ（オーダ）レジスタ

オーダレジスタ（各軸で使用する駆動情報が格納されたレジスタ）のウィンドウを開きます
（このレジスタはフラッシュROMに保存が可能です）

Ｋ（キープ）レジスタ

キープレジスタ（不揮発性記憶（電源を切っても残ります）領域のレジスタ）のウィンドウを開きます

Ｆ（ファイル）レジスタ

ファイルレジスタ（フラッシュメモリーに値を保持出来るレジスタ）のウィンドウを開きます

Ｌ（リンク）レジスタ

リンクレジスタ（外部機器と通信を行う為のレジスタ）のウィンドウを開きます

Ｅ（拡張）レジスタ

拡張レジスタ（タッチパネル等と通信を行う為のレジスタ）のウィンドウを開きます

Ｃ（パラメータ）レジスタ

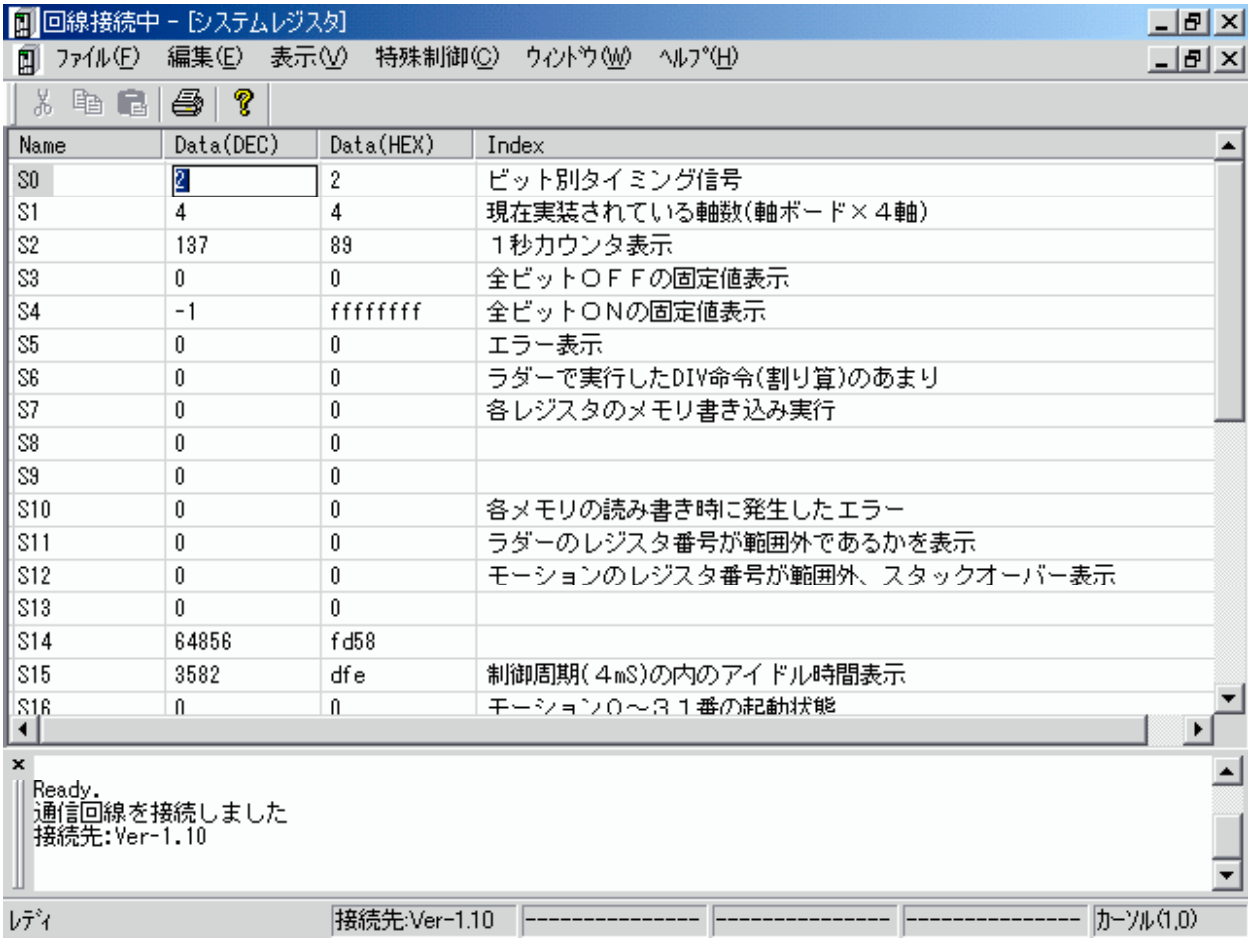
パラメータレジスタ（軸単位及び共通の設定項目レジスタ）のウィンドウを開きます
（パラメータウィンドウでは設定不可な項目も変更可能です）

Ｑ（実数）レジスタ

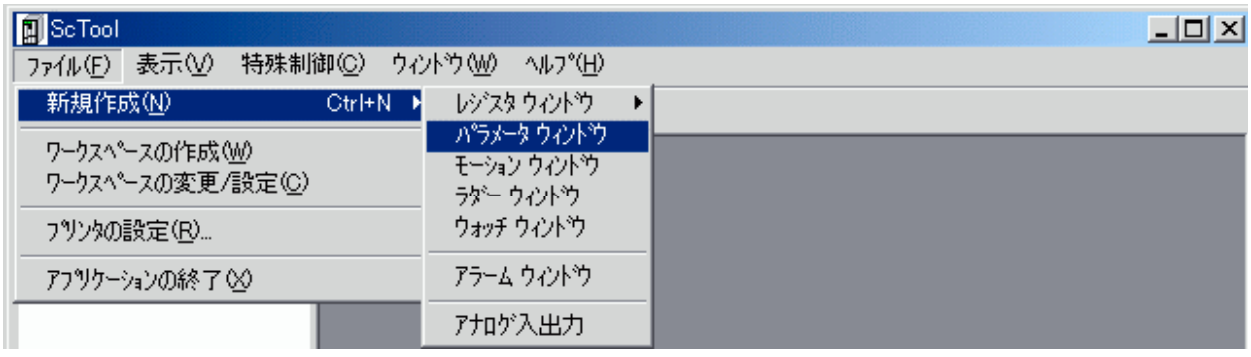
実数レジスタ（モーションから利用可能な実数が扱えるレジスタ）を開きます

---

レジスタウィンドウの例（システムレジスタ）




# ファイル(F)・新規作成(N)・パラメータウィンドウ




パラメータウィンドウを開きます

（パラメータのウィンドウを最大表示で開いています）

---

各個所の説明

「Name」

軸番号及びパラメータ名称が表示されます

「Data」

パラメータの現在の値が10進数で表示しています
この場所に直接数値を入力する事も可能です

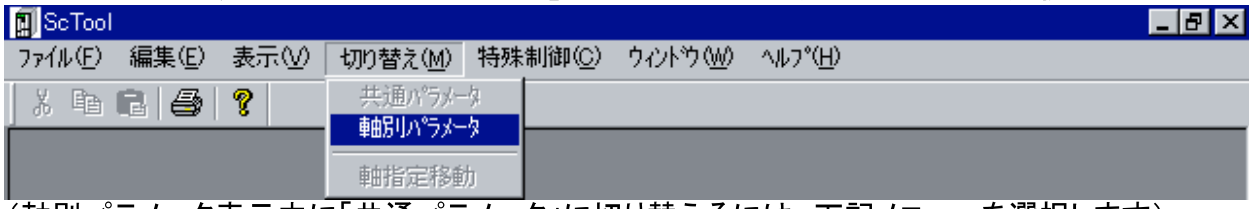
「Argument」

選択肢が表示されています。現在選択されている項目は、左側に「＊」マークが付きます
項目を選択するには、選択肢をクリックするか、「＊」の位置をクリックして下さい

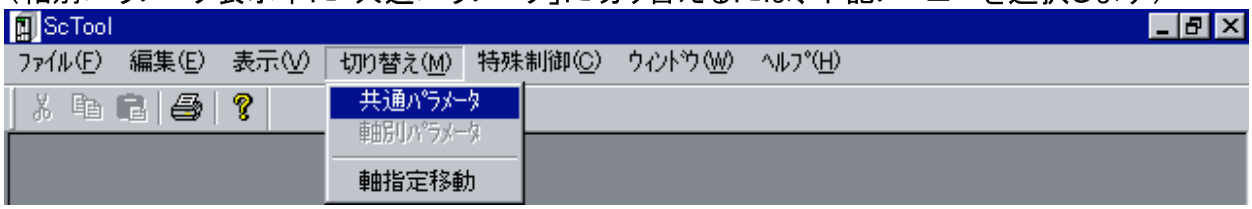
---

共通パラメータと軸パラメータの切り替え方

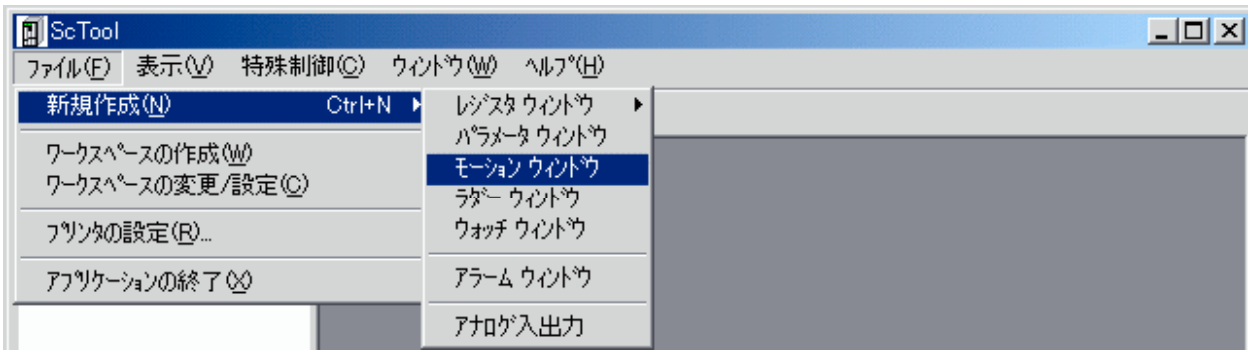
（共通パラメータ表示中に「軸別パラメータ」に切り替えるには、下記メニューを選択します）



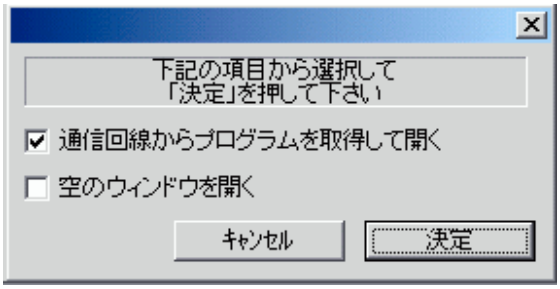

（軸別パラメータ表示中に「共通パラメータ」に切り替えるには、下記メニューを選択します）



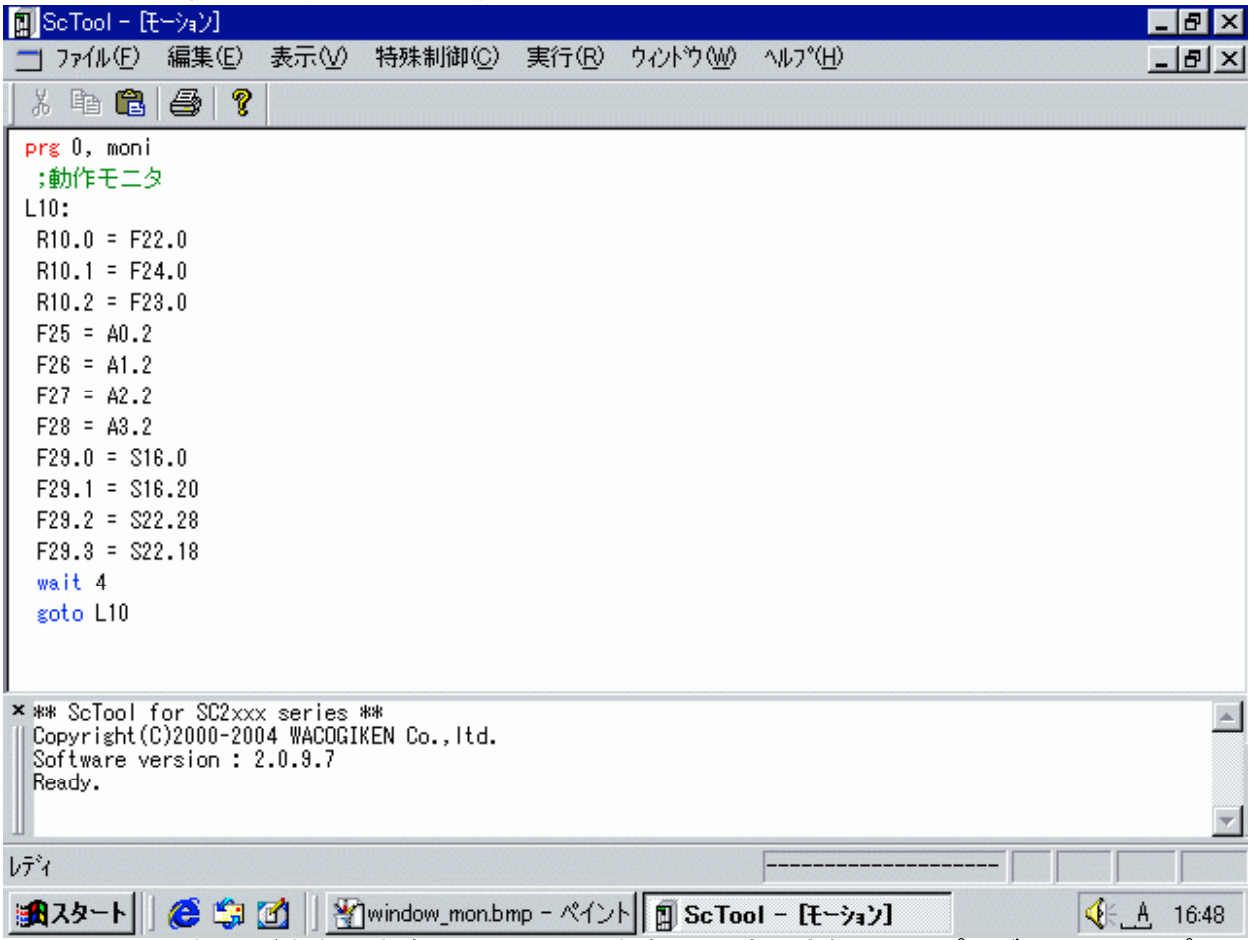
# ファイル(F)・新規作成(N)・モーションウィンドウ




モーションを開きます
メニューを選択すると、下記ウィンドウが表示されます。



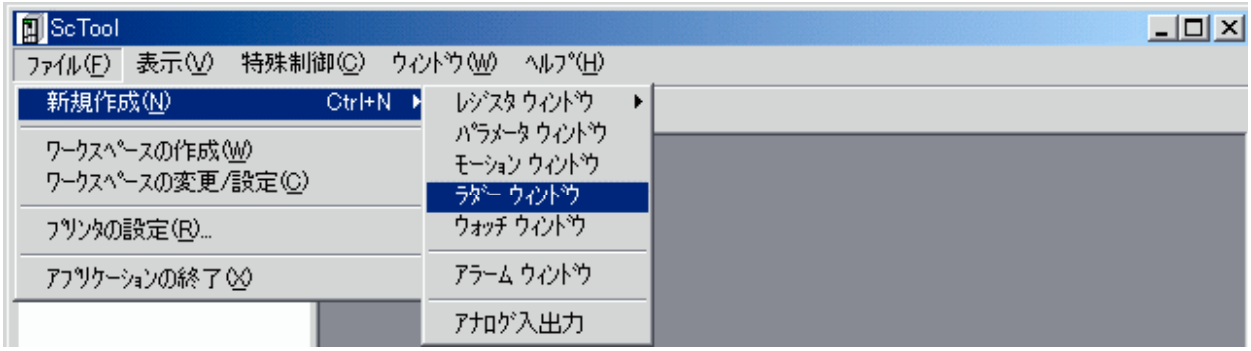

編集対象を選択すると、編集画面が表示されます



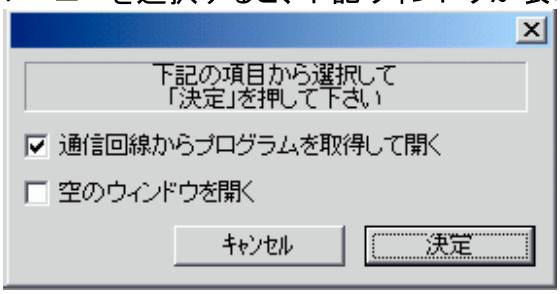

（モーションのウィンドウを最大表示で開いています。又、表示されているプログラムはサンプルと

して入力した物です）

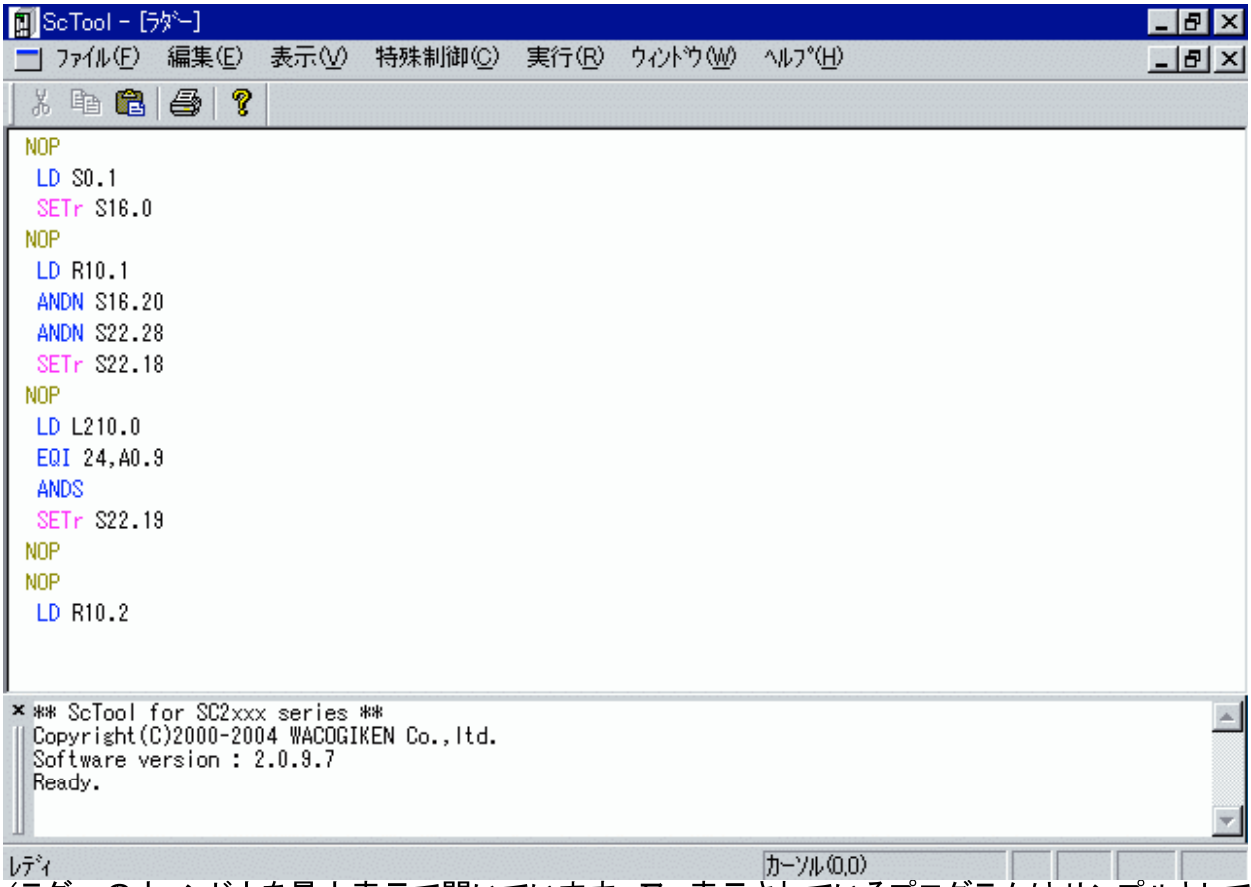
# ファイル(F)・新規作成(N)・ラダーウィンドウ




ラダーを開きます
メニューを選択すると、下記ウィンドウが表示されます。

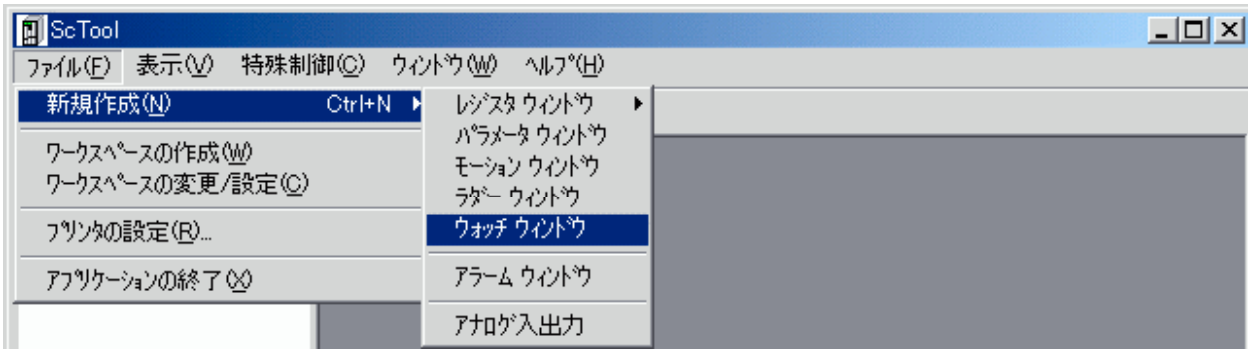



編集対象を選択すると、編集画面が表示されます

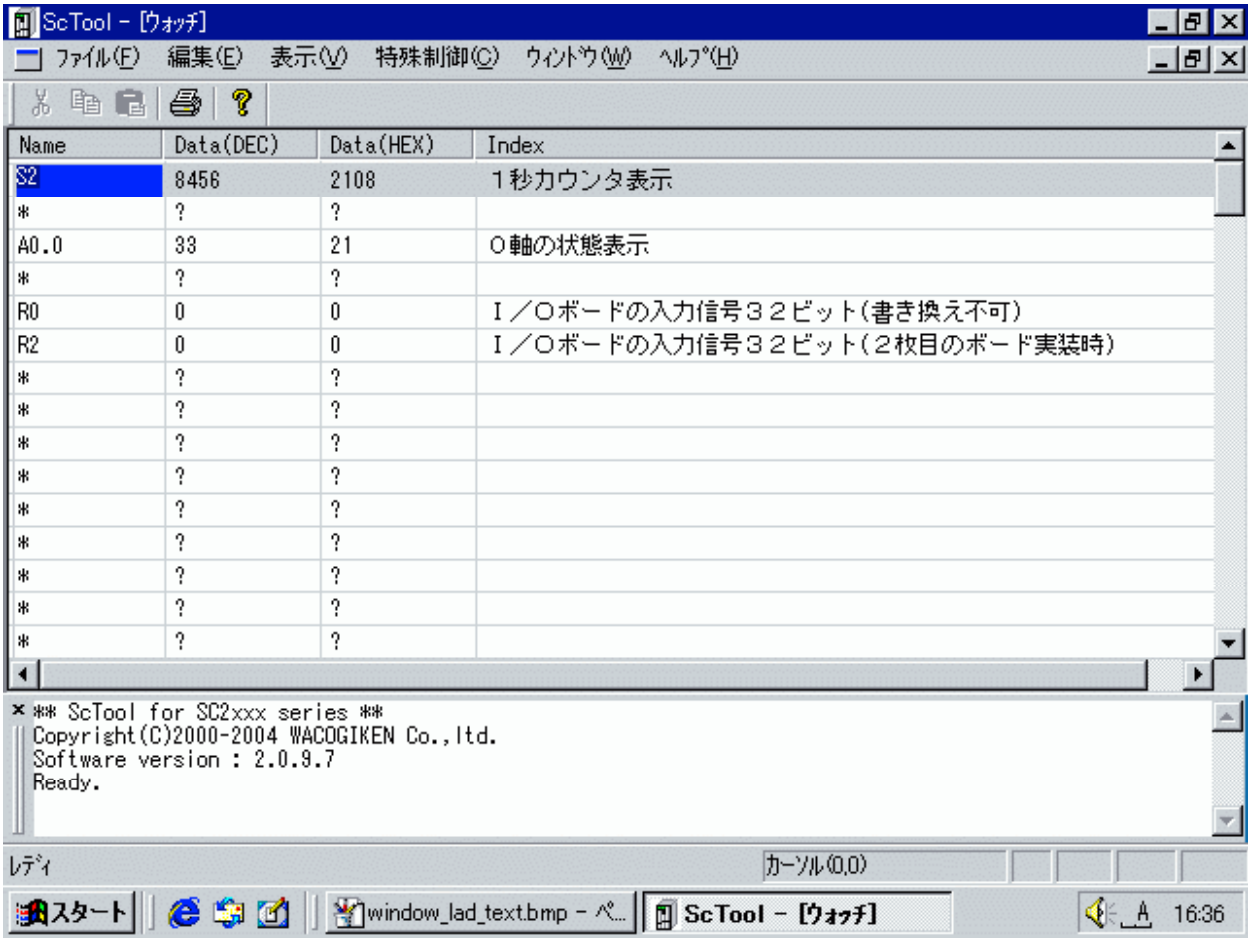


（ラダーのウィンドウを最大表示で開いています。又、表示されているプログラムはサンプルとして入力した物です）

# ファイル(F)・新規作成(N)・ウォッチウィンドウ



ウォッチウィンドウを開きます




（ウォッチウィンドウを最大表示で開いています。又、入力済みの個所はサンプルとして入力した物です）

---

各個所の説明

「Name」

監視するレジスタ名が表示されます（入力欄）

「Data(DEC)」

監視するレジスタの現在の値を10進数で表示しています

「Data(HEX)」

監視するレジスタの現在の値を16進数で表示しています

「Index」

監視するレジスタに対する説明を表示しています

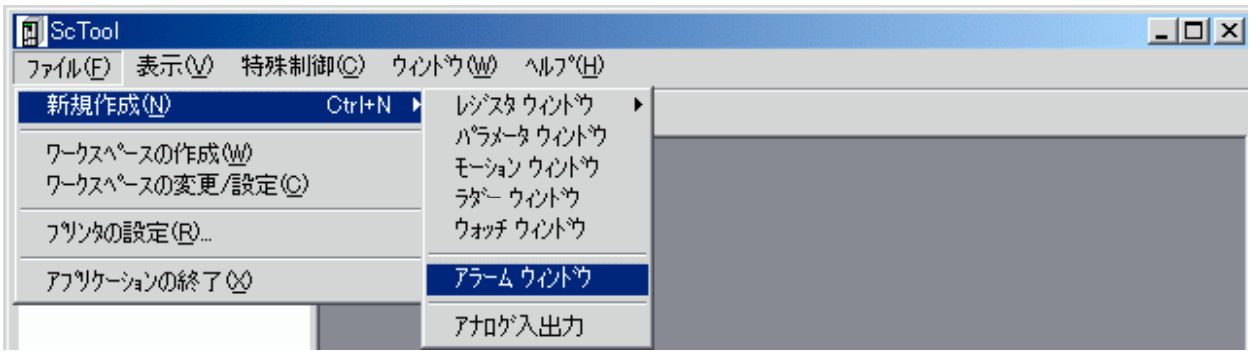
---

キー入力関係の説明

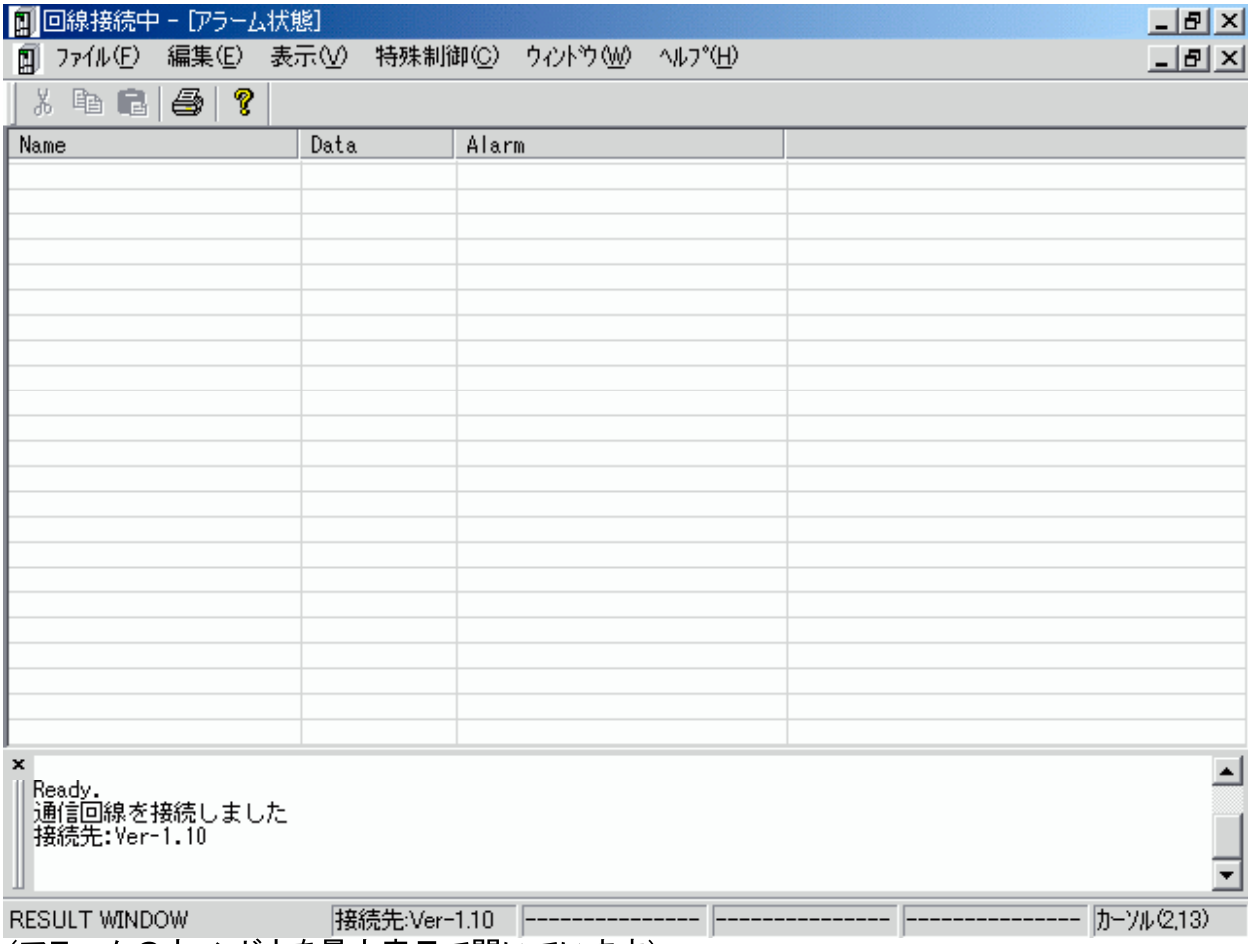
| キー操作 | 挙動 |
|---|---|
| リターンキー | 入力したデータをカーソル位置に書き込み |
| シフトキーを押しながらリターンキー | 入力したデータをカーソル位置に挿入 |
| 枠内を消してリターンキー | カーソル位置を消去し、それより下にある項目を上に移動する |

---

# ファイル(F)・新規作成(N)・アラームウィンドウ

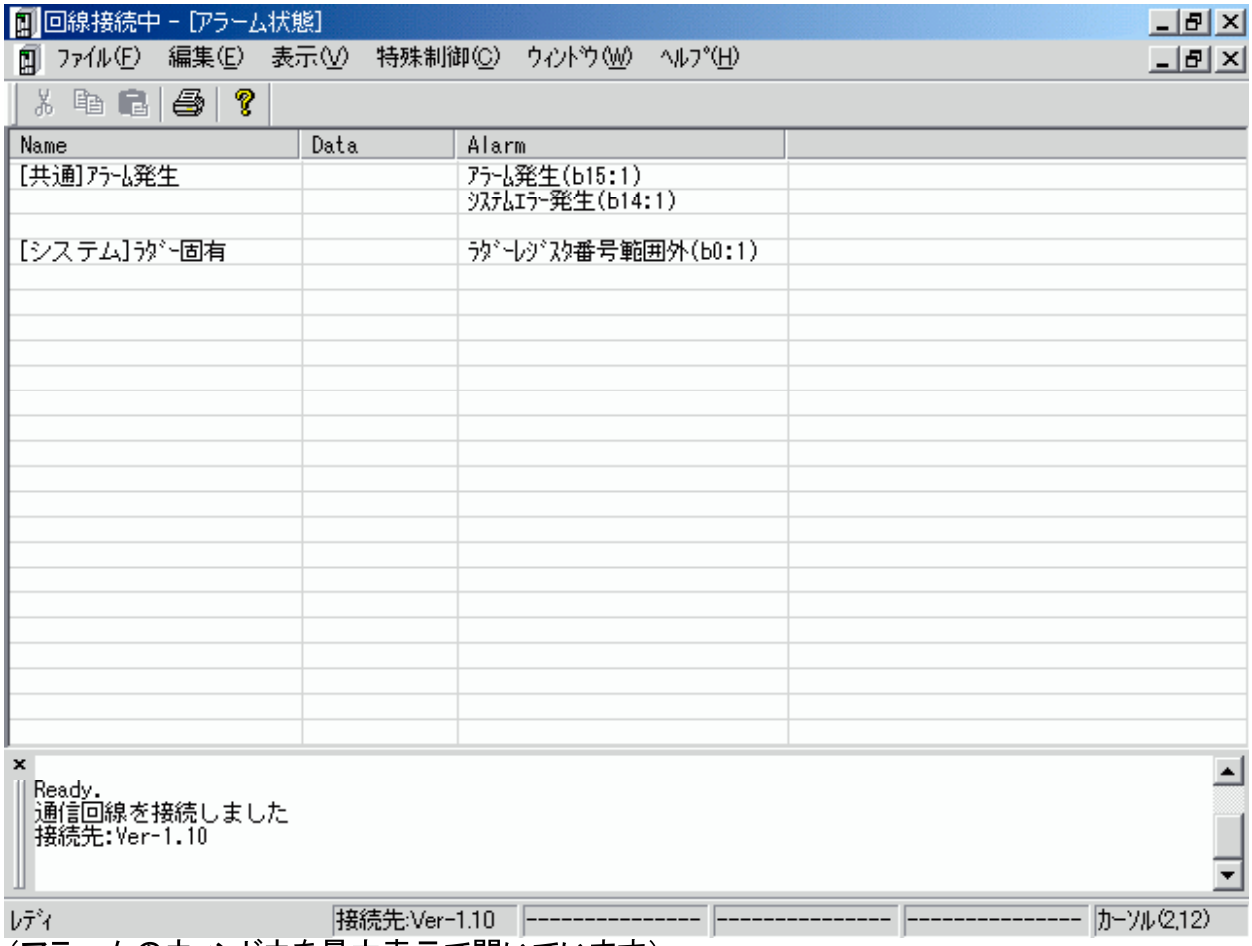



アラームウィンドウを開きます
アラームが発生していない状態でのアラームウィンドウ




（アラームのウィンドウを最大表示で開いています）

アラームが発生している状態でのアラームウィンドウ



（アラームのウィンドウを最大表示で開いています）

---

各個所の説明

「Name」

アラーム分類が表示されます

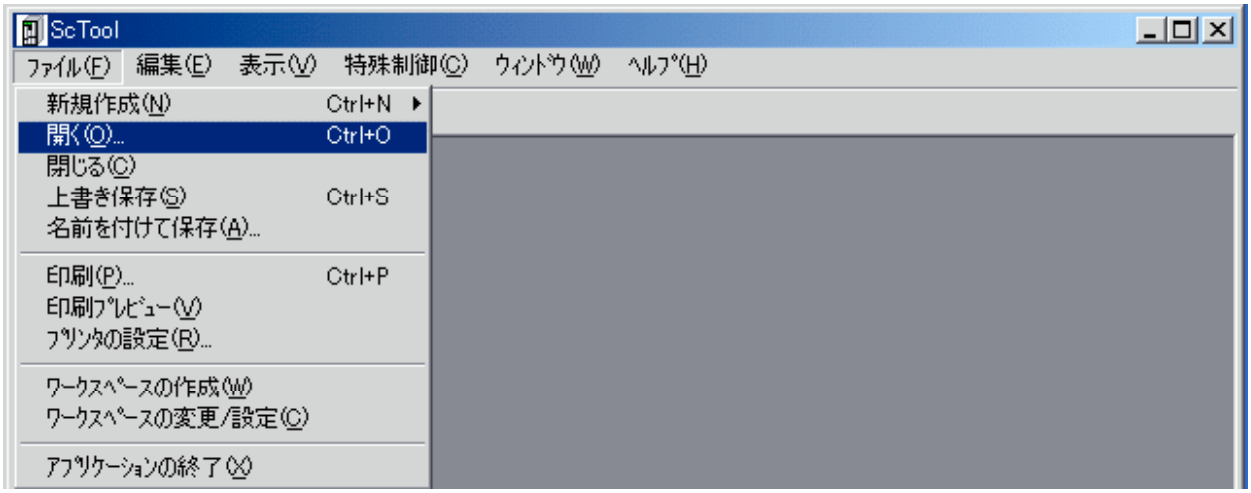
「Alarm」

発生しているアラームの詳細を表示しています

---

# ファイル(F)・開く(O)

---




開いているウィンドウのモードによって、読み込めるファイルに相違があります。
（詳細は下記）

| ウィンドウのモード | 読み込めるファイル |
|---|---|
| レジスタ（全レジスタ対応） | 各モードのレジスタファイル<br>テーブルファイル |
| モーション | モーションファイル |
| ラダー | ラダーファイル |
| パラメータ | パラメータファイル |
| ウォッチ | ウォッチファイル |

### レジスタファイル読み込み時の詳細

ウィンドウのモードがレジスタであれば、レジスタの種類には関係なく「レジスタファイル」を読み込む事が可能です。
読み込んだファイルは、即座に接続中のSC2xxxに転送され、書き込まれます。
但し、フラッシュROMに保存するには、別途フラッシュROMに対する操作が必要 となるので注意が必要です。
（Kレジスタを除く）

### モーション／ラダーファイル読み込み時の詳細

モーションのファイルを読み込んだ場合、編集領域にはファイルの内容が表示されますが、SC2xxxには転送されていません。
「実行」メニューから操作して、初めて転送されます。
レジスタと同様にフラッシュROMへの操作が必要 です。

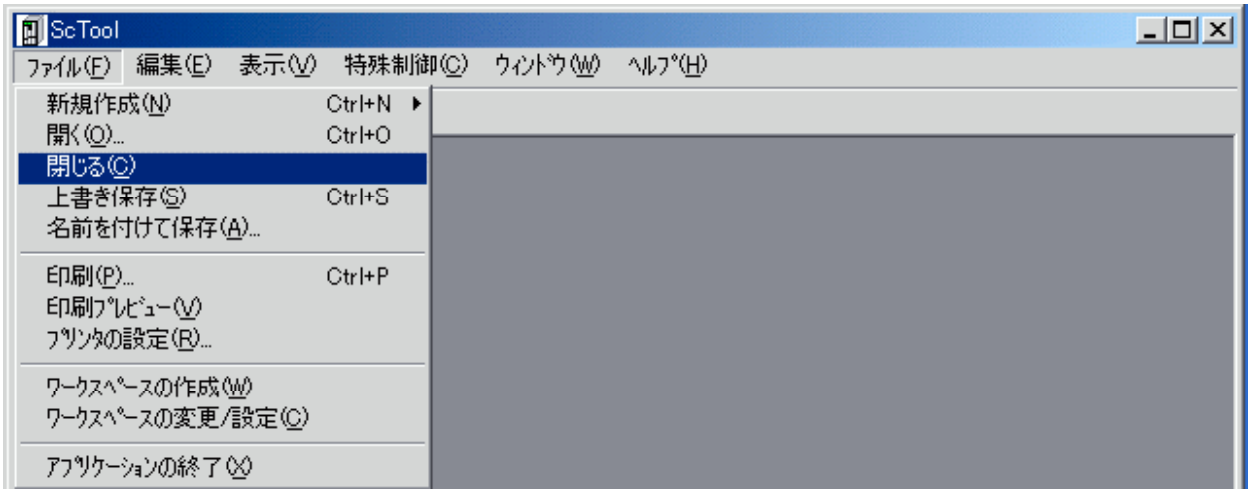
### ウォッチファイル読み込み時の詳細

読み込んだ時点から即監視が始まります。

### テーブルファイル読み込み時の詳細

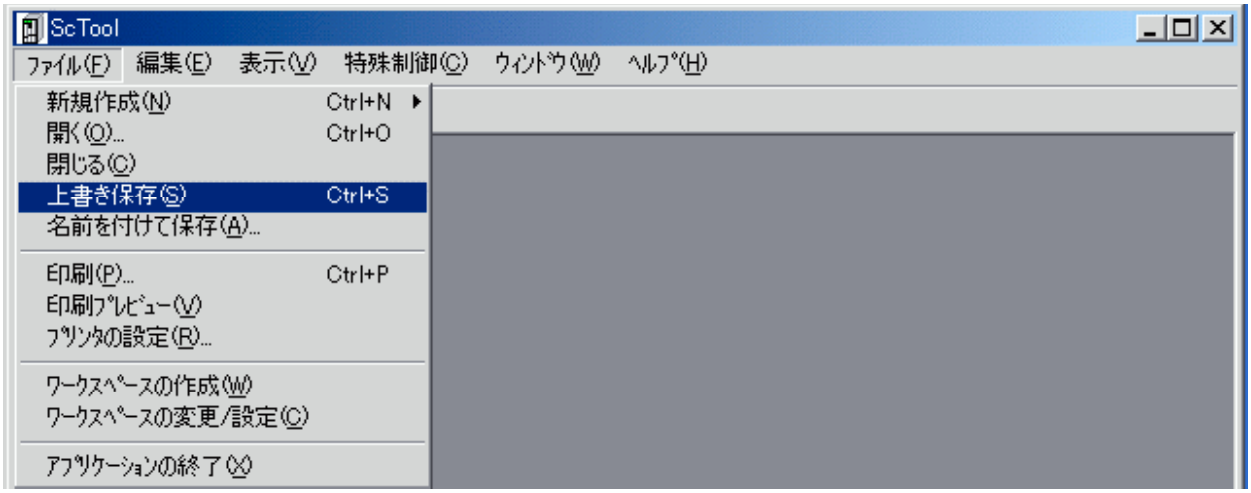
テーブルファイル（旧ツール類で作成したプロジェクト、又はバックアップで作成されたプロジェクト形式のデータに含まれるファイル）は、
読み込まれた時点で、「定義内容が空白でない場合は、現在の定義に上書き」されます。

# ファイル(F)・閉じる(C)




ウィンドウを閉じます。
ウィンドウを閉じる事により、編集内容が破棄されてしまう場合には、編集内容を反映させる為の選択肢が表示される場合もあります。

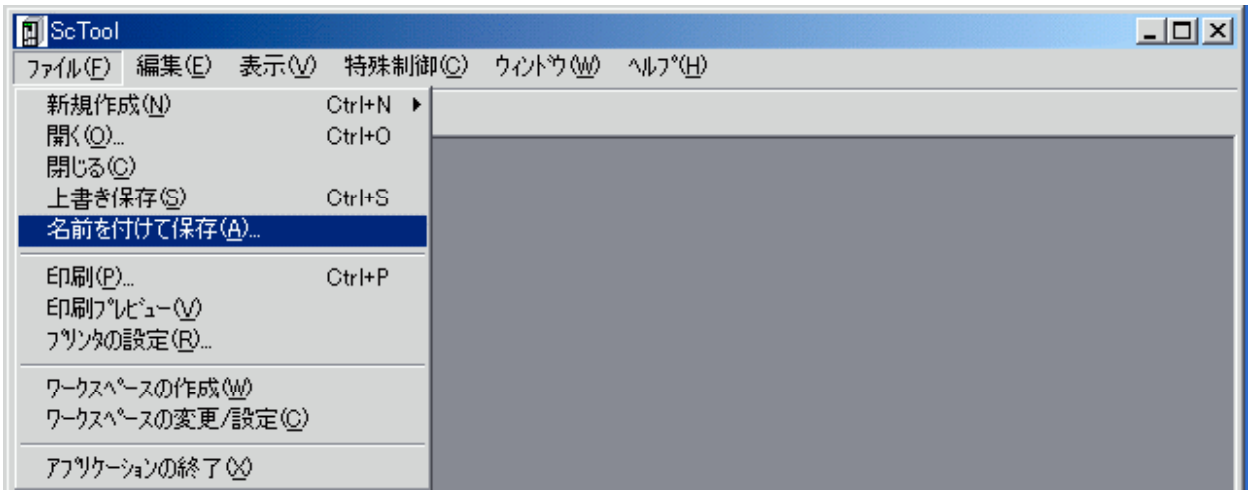
# ファイル(F)・上書き保存(S)




編集中のデータをファイルに保存します
保存されるファイルは、開いているウィンドウのモードによって変わります（下記）

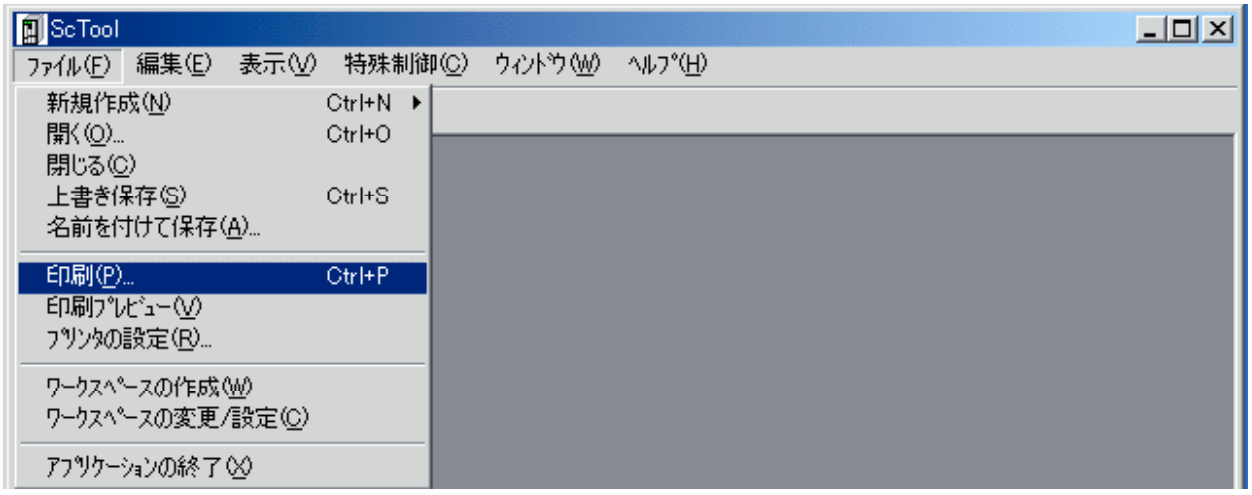
| ウィンドウのモード | 作成されるファイル |
|---|---|
| レジスタ（全レジスタ対応） | 各モードに対応したレジスタファイル<br>作成されたファイルには、「定義」も一緒に格納される。<br>（テキストファイル） |
| モーション | モーションプログラム（テキストファイル） |
| ラダー | ラダープログラム（テキストファイル） |
| ウォッチ | ウォッチファイル（テキストファイル） |

# ファイル(F)・名前を付けて保存(A)




編集中のデータを任意のファイル名を指定して保存します
詳細は、「上書き保存(S)」を参照願います。

# ファイル(F)・印刷(P)

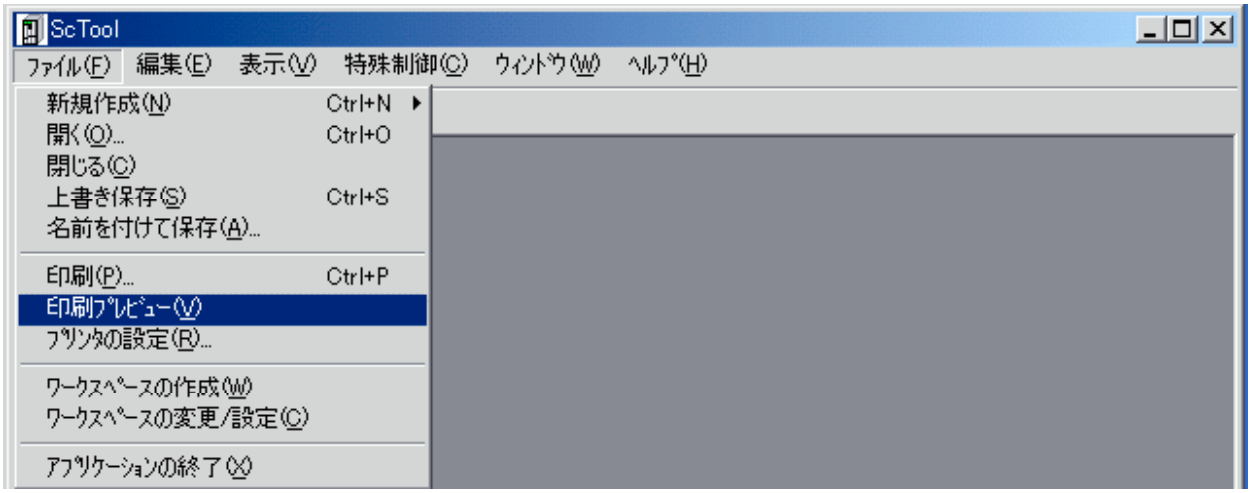



編集中のデータ又は接続対象のSC2xxxのデータを印刷します。
印刷される対象は、ウィンドウのモードによって変わります。(下記)

| ウィンドウのモード | 指定1 | 指定2 | 印刷詳細 |
|---|---|---|---|
| レジスタ<br>(全レジスタ) | 開始レジスタ名 | 終了レジスタ名 | 接続対象のレジスタ |
| モーション | 開始行番号 | 終了行番号 | 編集中のモーション |
| ラダー | 開始行番号 | 終了行番号 | 編集中のラダー |
| ウォッチ | 無し | 無し | 監視中のレジスタ名及びデータ |
| | | | |

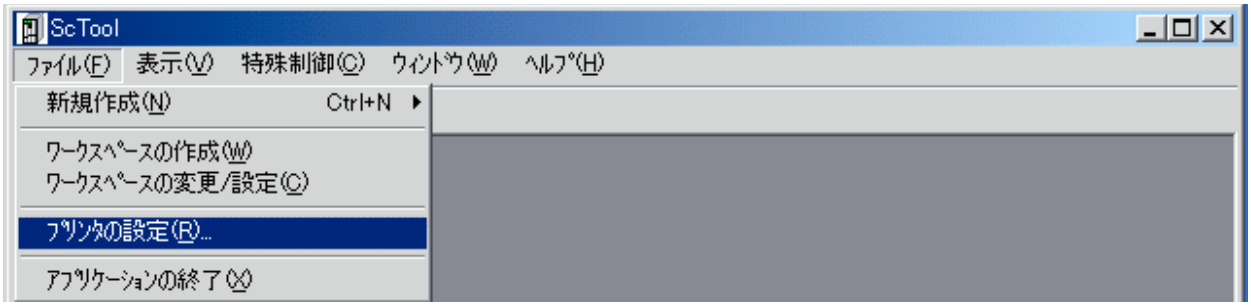
| | | | |
|---|---|---|---|
| アラーム | 無し | 無し | 発生中のアラーム |

# ファイル(F)・印刷プレビュー(V)

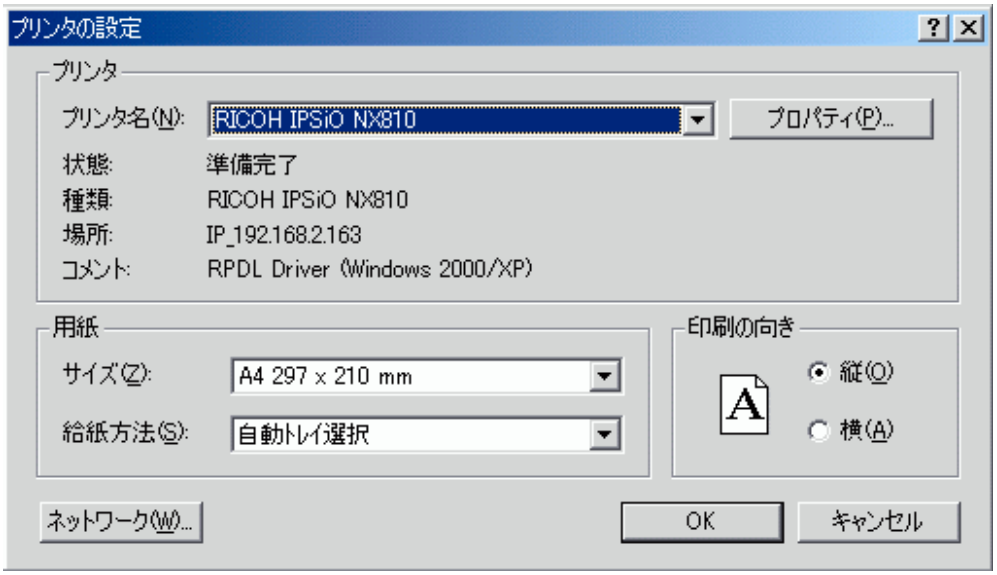



印刷結果を画面上で確認出来ます。
詳細は「印刷(P)」を参照願います。

# ファイル(F)・プリンタの設定(R)

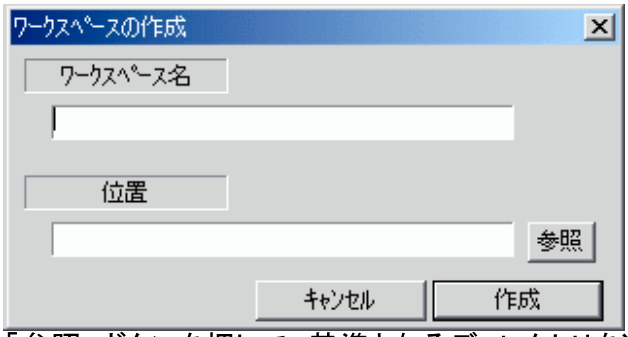



印刷設定ウィンドウを開きます（下記はWindows2000での表示）



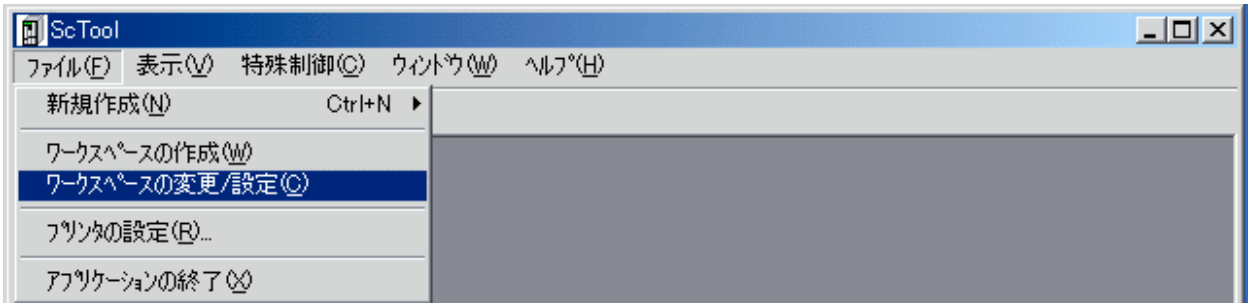

# ファイル(F)・ワークスペースの作成(W)


ワークスペース（作業ディレクトリ）の作成を行います。
起動すると下記ウィンドウが表示されます




「参照」ボタンを押して、基準となるディレクトリを選択する事が可能です。
ディレクトリは、「位置」＋「ワークスペース名」で作られます。

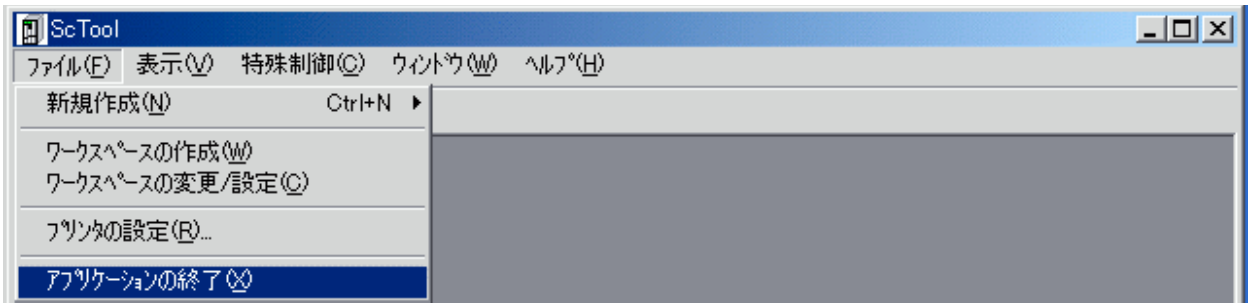
# ファイル(F)・ワークスペースの変更/設定(C)




ワークスペース（作業ディレクトリ）の変更（又は指定）を行います。
起動すると下記ウィンドウが表示されます


「参照」ボタンを押して、ディレクトリを選択する事が可能です。

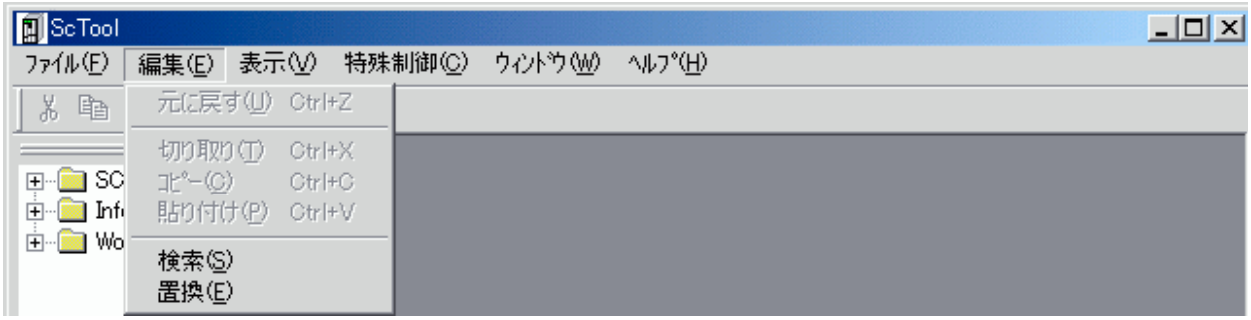
# ファイル(F)・アプリケーションの終了(X)



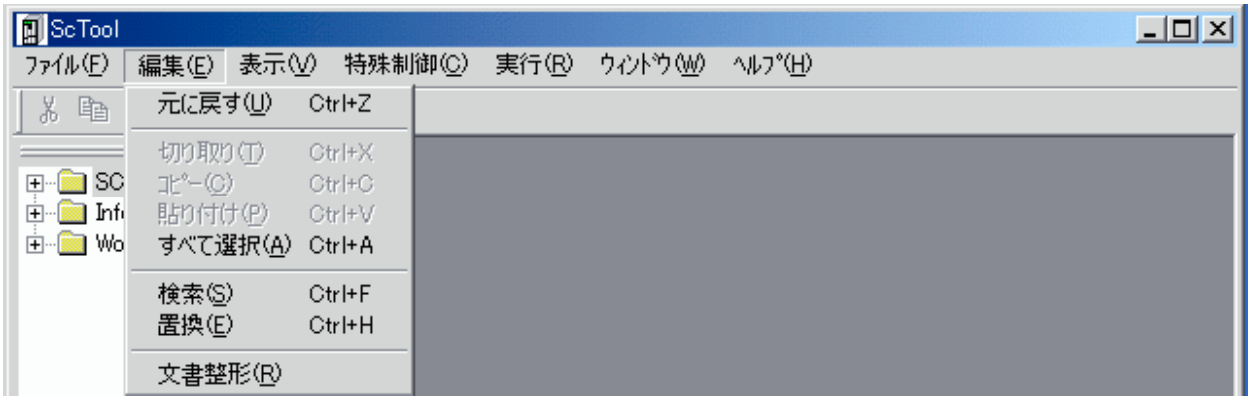

本ソフトを終了します
保存されていないデータが存在する場合には、該当するウィンドウが警告を表示します。
（警告された時点で保存する事も可能です）

# 編集(E)メニュー

レジスタ類ウィンドウ選択時



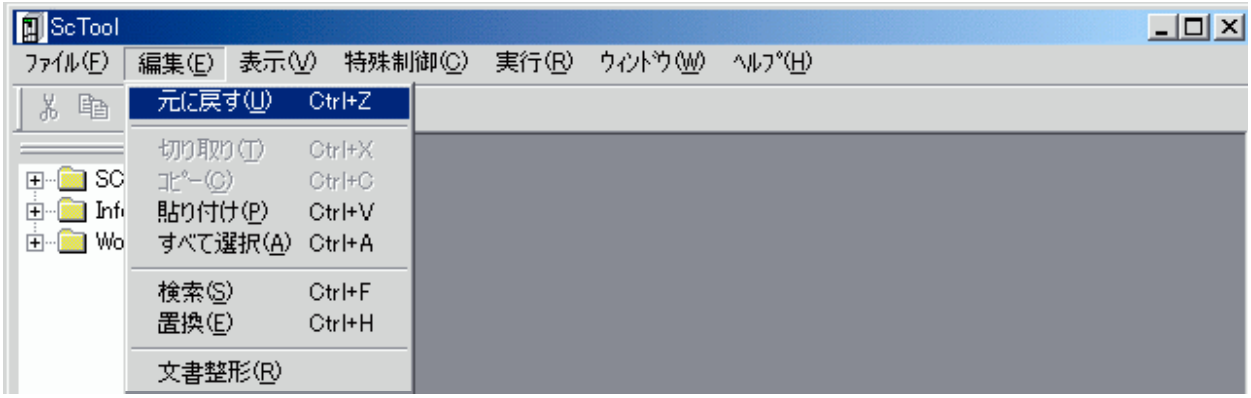

プログラム類ウィンドウ選択時




このメニューには、下記８点の項目があります

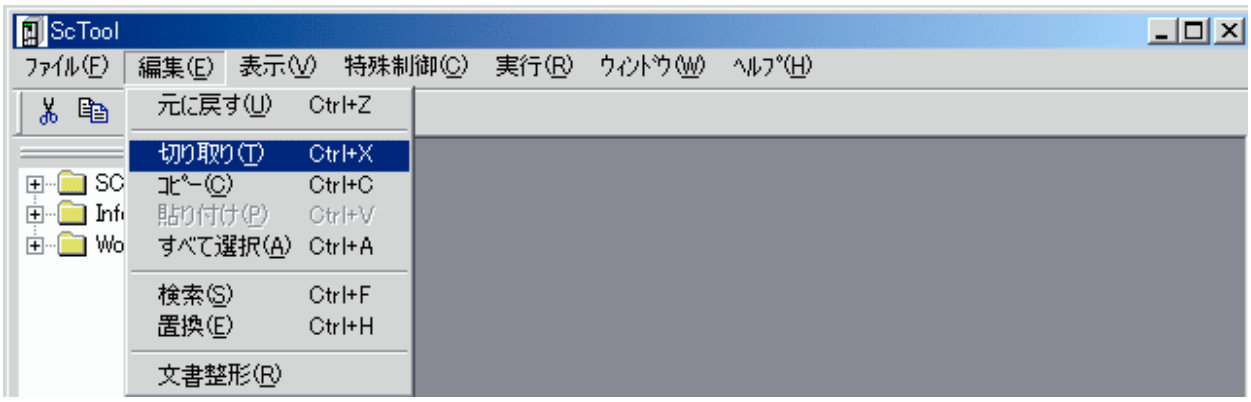
元に戻す
切り取り
コピー
貼り付け
検索
置換
文書整形

# 編集(E)・元に戻す



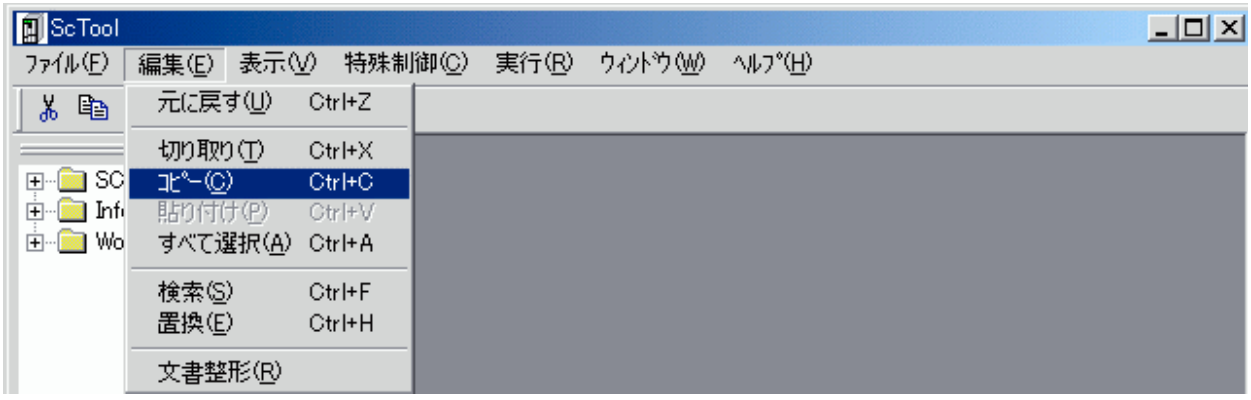

直前に行った作業をキャンセルします。
現在殆どの機能で動作しません。御迷惑をお掛けします

# 編集(E)・切り取り(T)



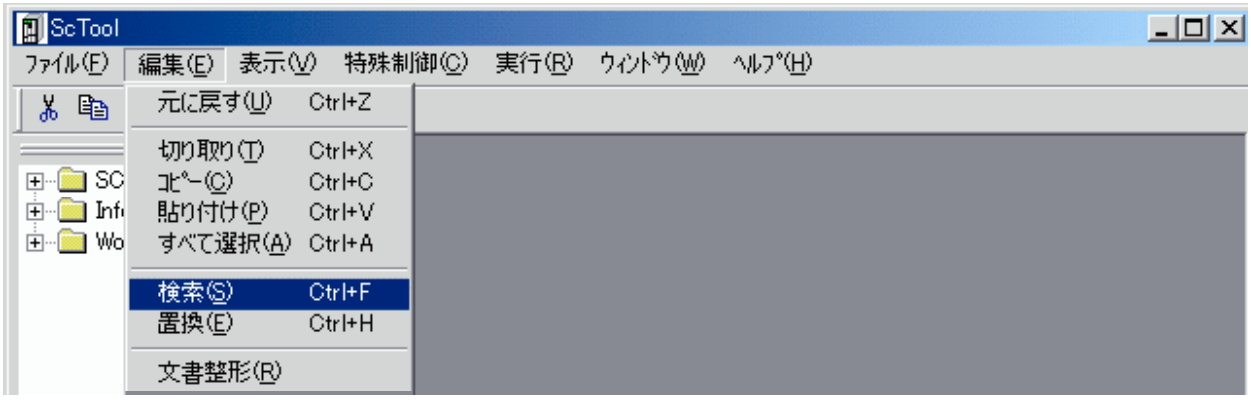

指定された領域を切り取ります

# 編集(E)・コピー(C)

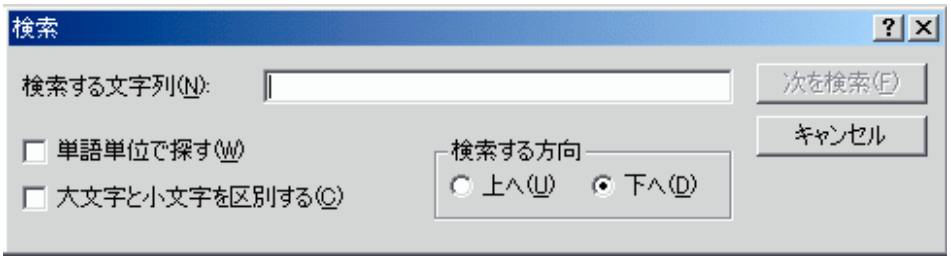



指定された領域をクリップボードにコピーします

# 編集(E)・検索



文字列を検索します。
選択すると、下記ウィンドウが表示されます。




「検索する文字列(N)」
　　検索を行う文字列を入力します
「単語単位で探す(W)」
　　現在使用出来ません。御迷惑をお掛けします
「大文字と小文字を区別する(C)」
　　検索対象と比較する時に、入力された文字を厳密に比較します
「検索する方向」
　　現在の編集位置から、上方向、又は下方向のどちらに向かって検索を行うか設定します

---

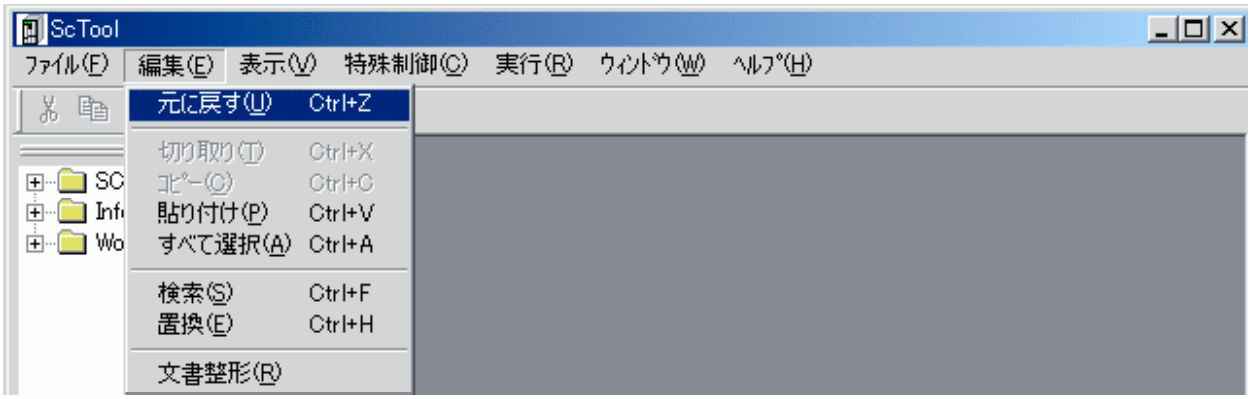
検索対象となる個所
検索対象となる個所は、開いているウィンドウのモードによって変わります（下記）

| ウィンドウのモード | 検索対象 |
|---|---|
| レジスタ（全レジスタ対応） | レジスタ名 |
| モーション | モーションプログラム |
| ラダー | ラダープログラム |
| パラメータ | パラメータ名 |
| ウォッチ | レジスタ名 |

---

# 編集(E)・貼り付け

---

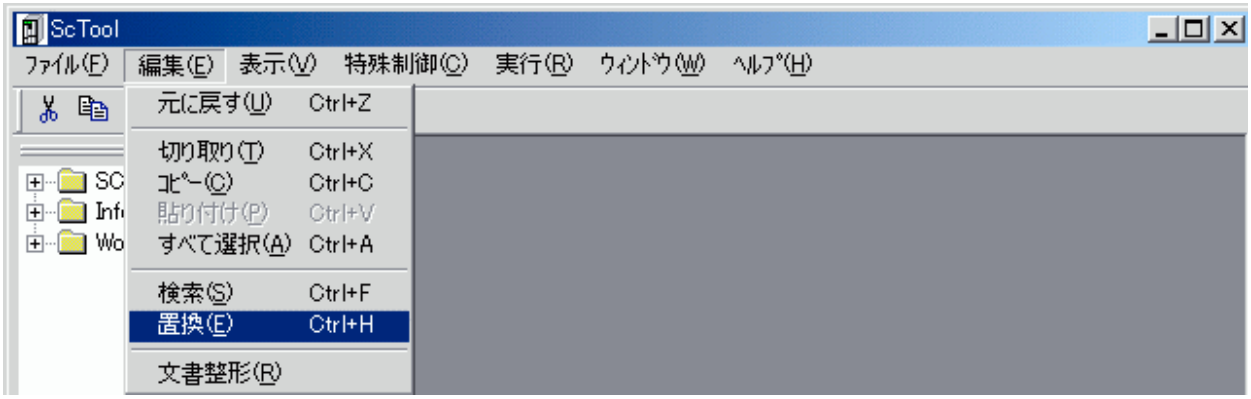



---

クリップボードに積まれたデータを貼り付けます

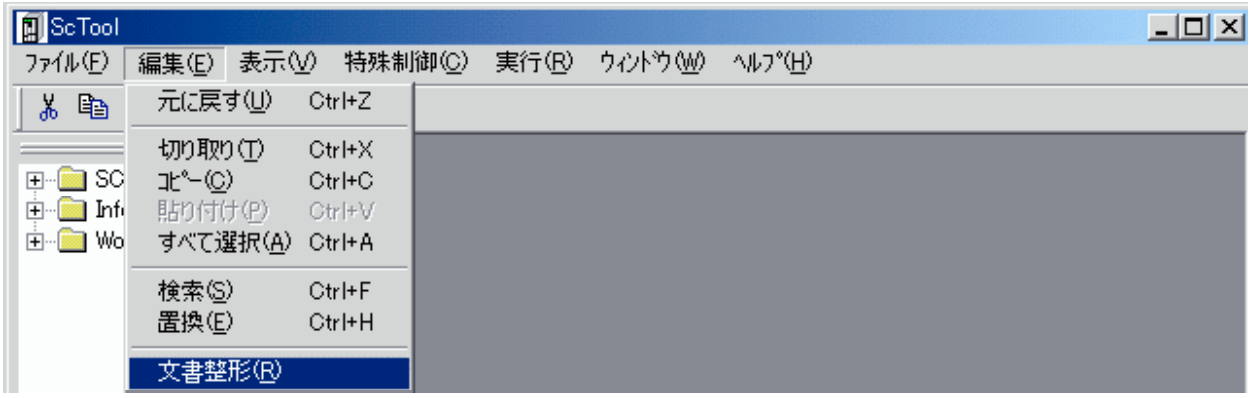
---

# 編集(E)・置換




文字列を置換します。
現在動作しません。御迷惑をお掛けします

# 編集(E)・文書整形




編集中プログラムを整形します。
モーションの場合、下記法則で整形されます

| 命令 | 法則 |
|---|---|
| prg / sub | この行は左端から始まり、次の行から繰り下がる |
| ラベル | この行は左端から始まり、次の行は前の繰り下がり位置に戻る |
| if | 次の行から繰り下がる |
| elseif / else | この行は繰り上がるが、次の行は元の繰り下がり位置に戻る |
| endif | この行から繰り上がる |
| その他の命令 | 現在の位置を継続 |

ラダーの場合
下記法則で整形されます

| 命令 | 法則 |
|---|---|

| | |
|---|---|
| NOP | この行は左端から始まる |
| その他の命令 | 左端から1つ繰り下がった位置 |

整形サンプル

モーション

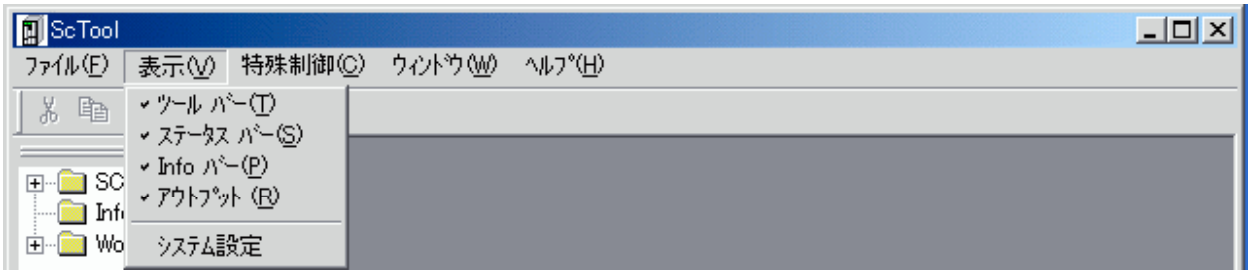
```
prg 1
 if r0.0 = 0
  if r0.1 = 0
   if r0.2 = 0
    f100 = 1
   endif
   f101=1
  endif
  f102=1
 elseif r0.3 = 0
  f103=0
 endif
 stop
```

ラダー

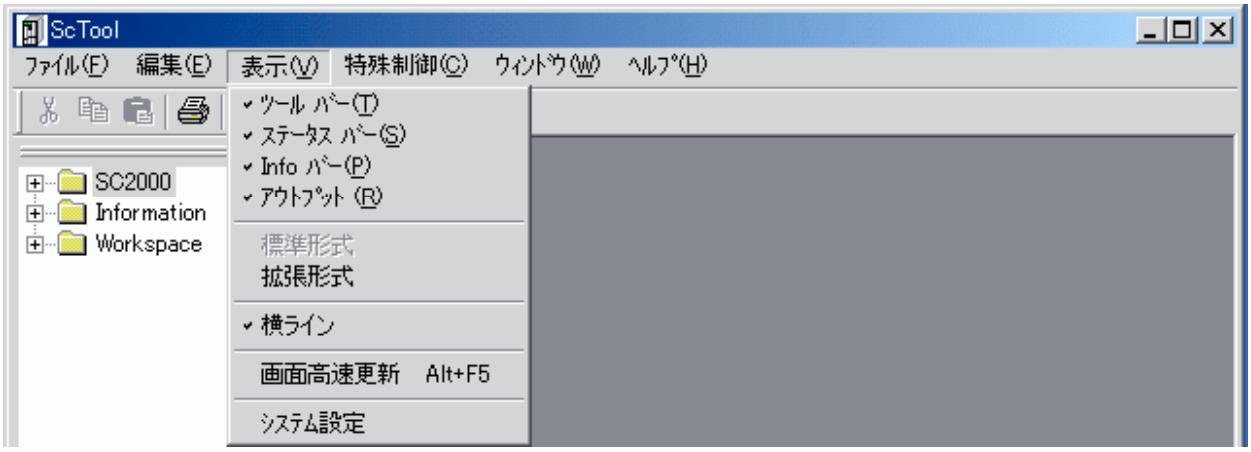
```
 LD R0.0
 SETr S16.0
 LDN R0.0
 RESr A0.24.0
NOP
 LD R0.1
 SETr S16.1
 LD R0.2
 SETr S16.2
 LD S0.6
 SETr S16.3
NOP
 LD R0.3
 SETr S16.4
```

# 表示(V)メニュー

レジスタ／プログラム類ウィンドウ未選択時



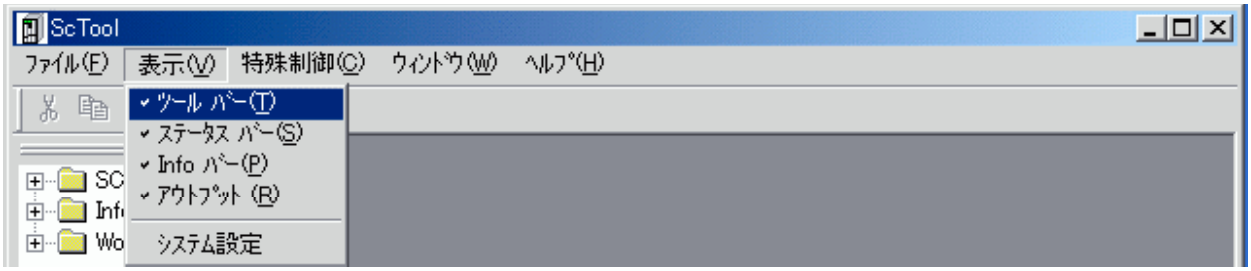

レジスタ／プログラム類ウィンドウ選択時




このメニューには、最大下記9点の項目があります
（項目名をクリックすると、詳細説明に移動します）

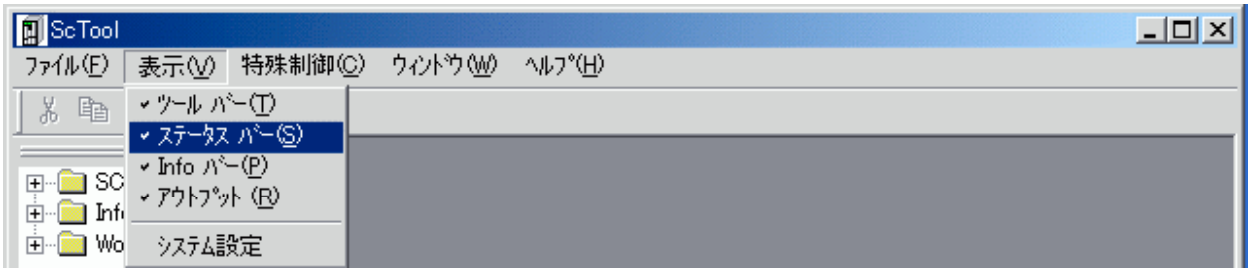
ツール バー
ステータス バー
Info バー（情報バー）
アウトプット
標準形式
拡張形式
横ライン
画面高速更新
システム設定

# 表示(V)・ツールバー



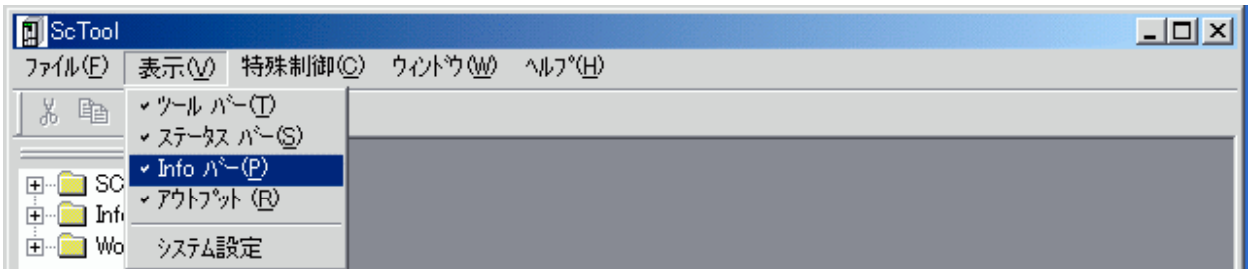

ツールバーの表示／非表示を制御します
2度選択すると元に戻ります。

# 表示(V)・ステータスバー



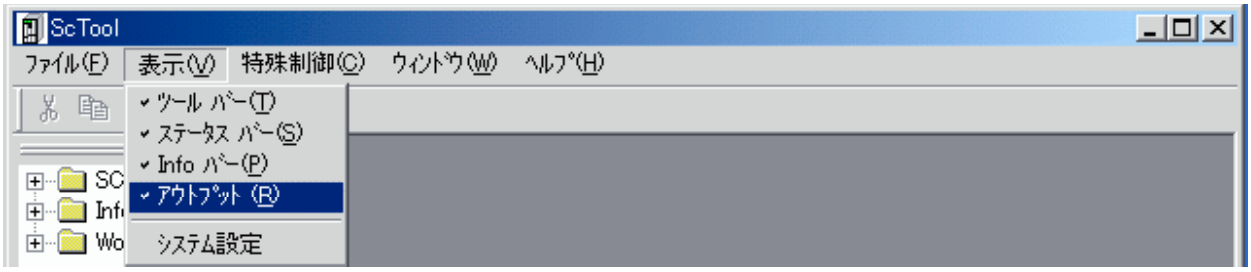

ステータスバーの表示／非表示を制御します
2度選択すると元に戻ります。

# 表示(V)・Infoバー(P)



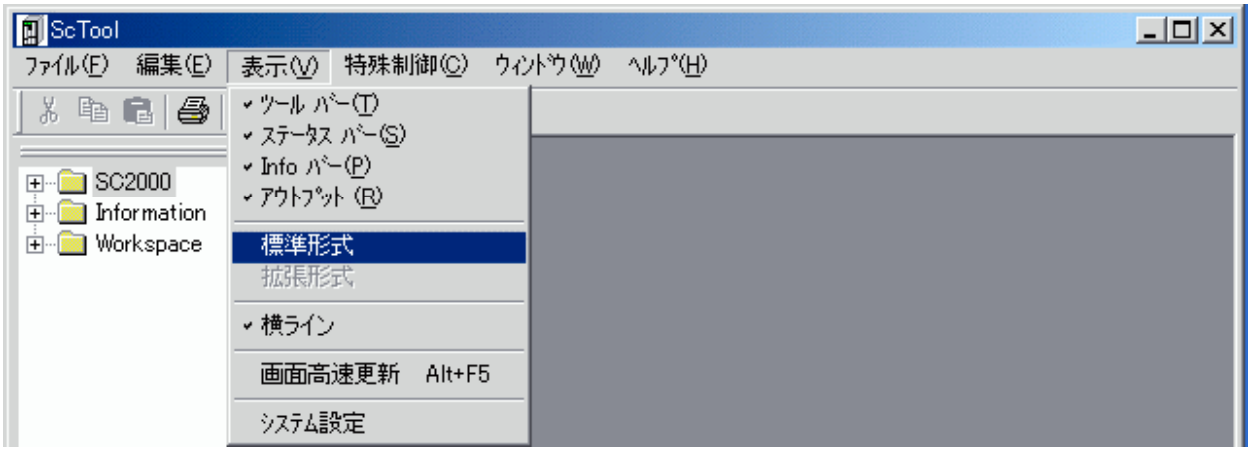

Infoバー（情報バー）の表示／非表示を制御します
2度選択すると元に戻ります。

# 表示(V)・アウトプット




アウトプット領域（リザルトウィンドウ）の表示／非表示を制御します
2度選択すると元に戻ります。

# 表示(V)・標準形式




レジスタ表示時に、表示内容を標準形式で表示します。
（データの一部を16進数で表示します）
拡張表示時にしか選択出来ません

（表示例）レジスタ・ウィンドウ（システムレジスタ）

回線接続中 - [システムレジスタ]

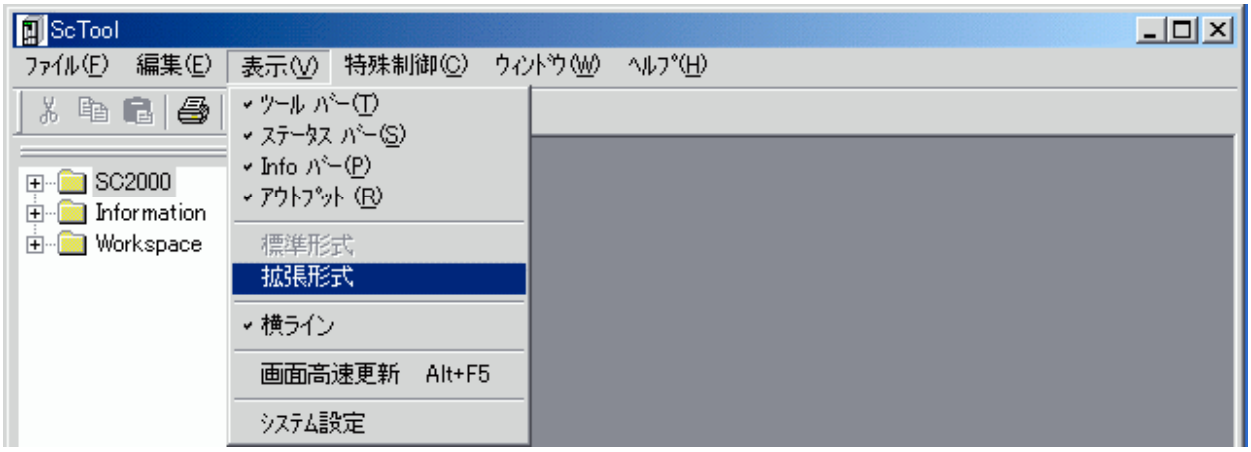
ファイル(F) 編集(E) 表示(V) 特殊制御(C) ウィンドウ(W) ヘルプ(H)

| Name | Data(DEC) | Data(HEX) | Index |
|---|---|---|---|
| S0 | 2 | 2 | ビット別タイミング信号 |
| S1 | 4 | 4 | 現在実装されている軸数(軸ボード×４軸) |
| S2 | 137 | 89 | １秒カウンタ表示 |
| S3 | 0 | 0 | 全ビットＯＦＦの固定値表示 |
| S4 | -1 | ffffffff | 全ビットＯＮの固定値表示 |
| S5 | 0 | 0 | エラー表示 |
| S6 | 0 | 0 | ラダーで実行したDIV命令(割り算)のあまり |
| S7 | 0 | 0 | 各レジスタのメモリ書き込み実行 |
| S8 | 0 | 0 | |
| S9 | 0 | 0 | |
| S10 | 0 | 0 | 各メモリの読み書き時に発生したエラー |
| S11 | 0 | 0 | ラダーのレジスタ番号が範囲外であるかを表示 |
| S12 | 0 | 0 | モーションのレジスタ番号が範囲外、スタックオーバー表示 |
| S13 | 0 | 0 | |
| S14 | 64856 | fd58 | |
| S15 | 3582 | dfe | 制御周期(４mS)の内のアイドル時間表示 |
| S16 | 0 | 0 | モーション０～３１番の起動状態 |

Ready.
通信回線を接続しました
接続先:Ver-1.10

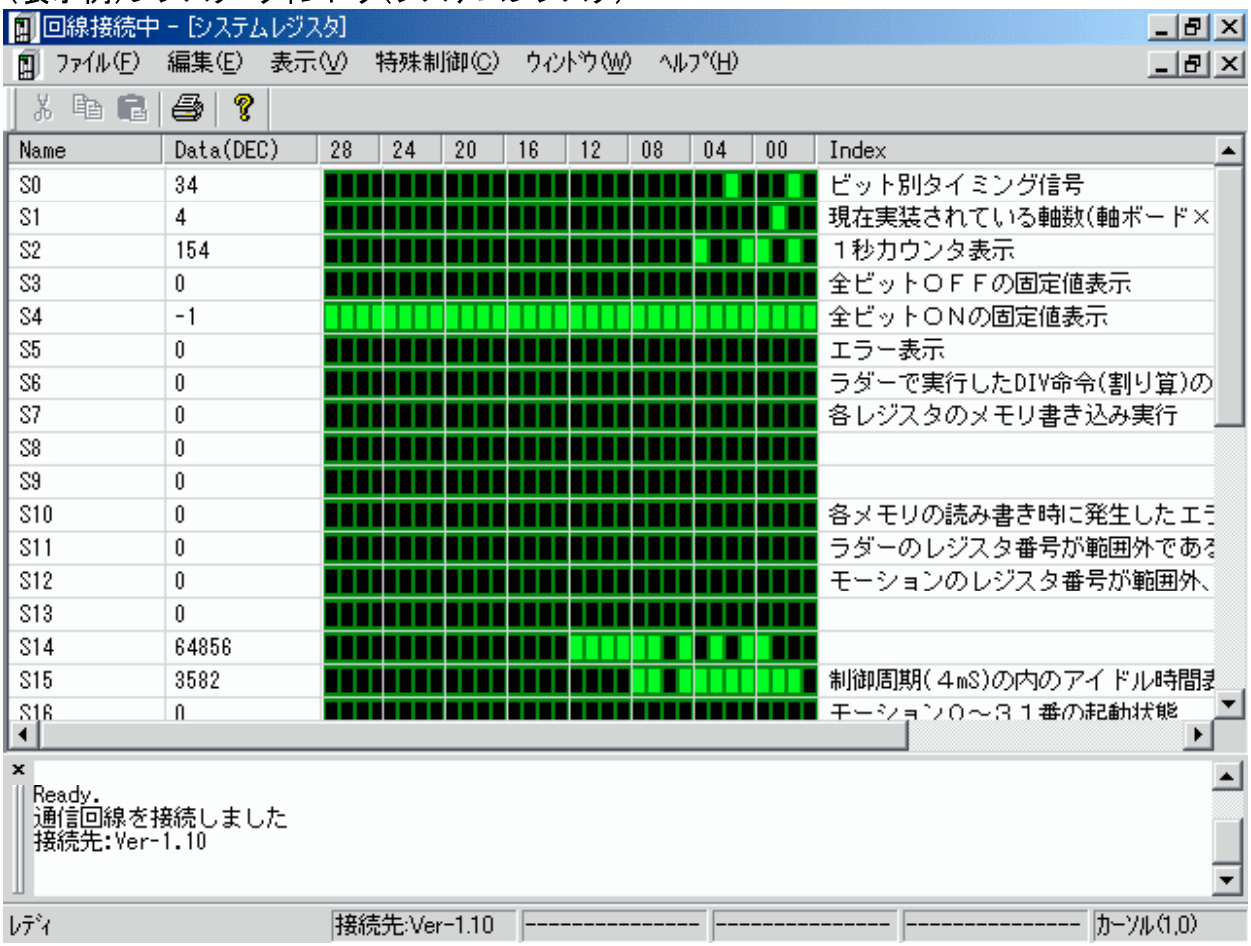
レディ 接続先:Ver-1.10 カーソル(1,0)

# 表示(V)・拡張形式

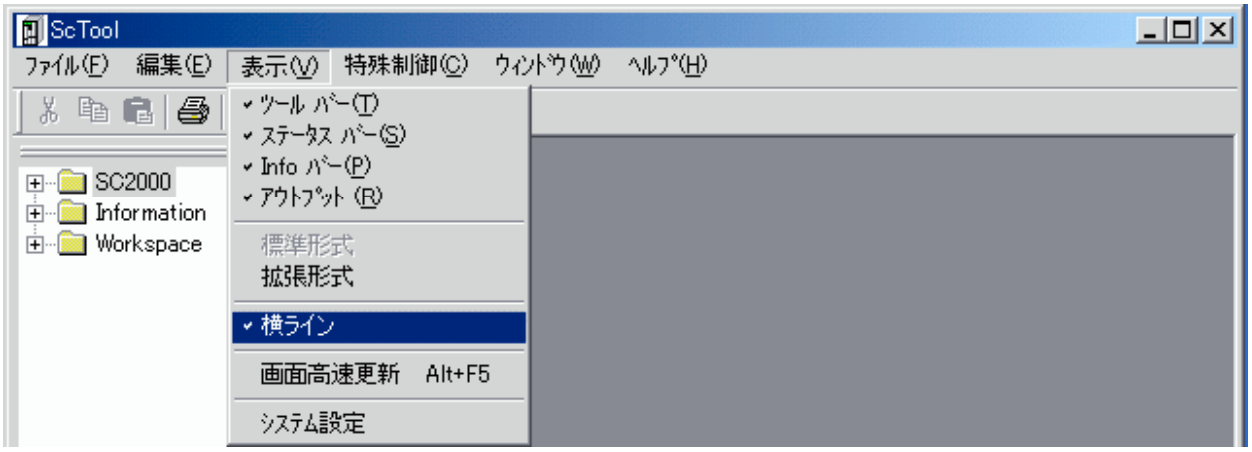



レジスタ表示時に、表示形式を拡張形式（ビット表示 ）で表示します。
ビット表示部は、マウスでクリックする事により、操作が可能です。
（ビット表示部でマウスをクリックした場合、該当する位置に有るビットは反転 します）

（表示例）レジスタ・ウィンドウ（システムレジスタ）

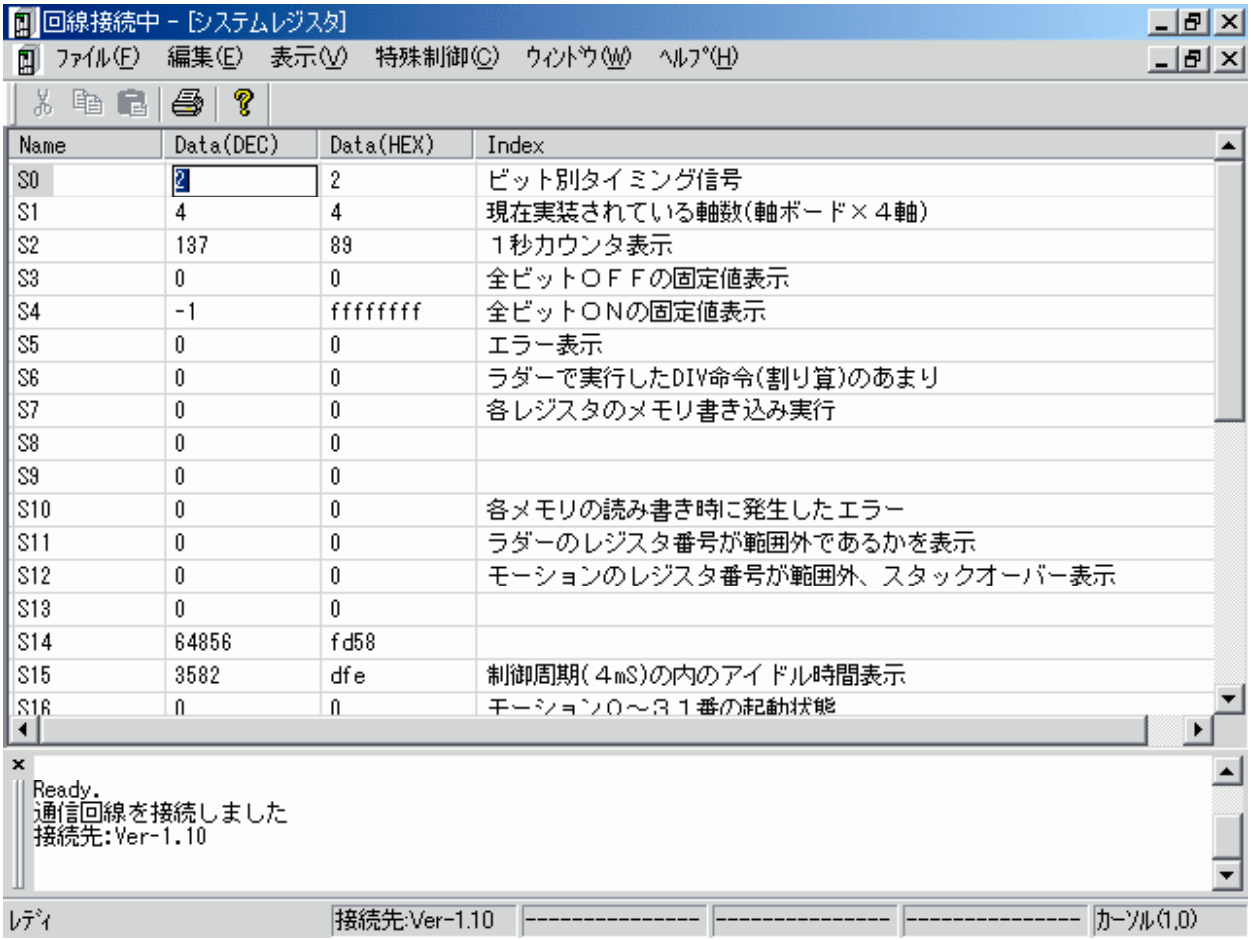



# 表示(V)・横ライン

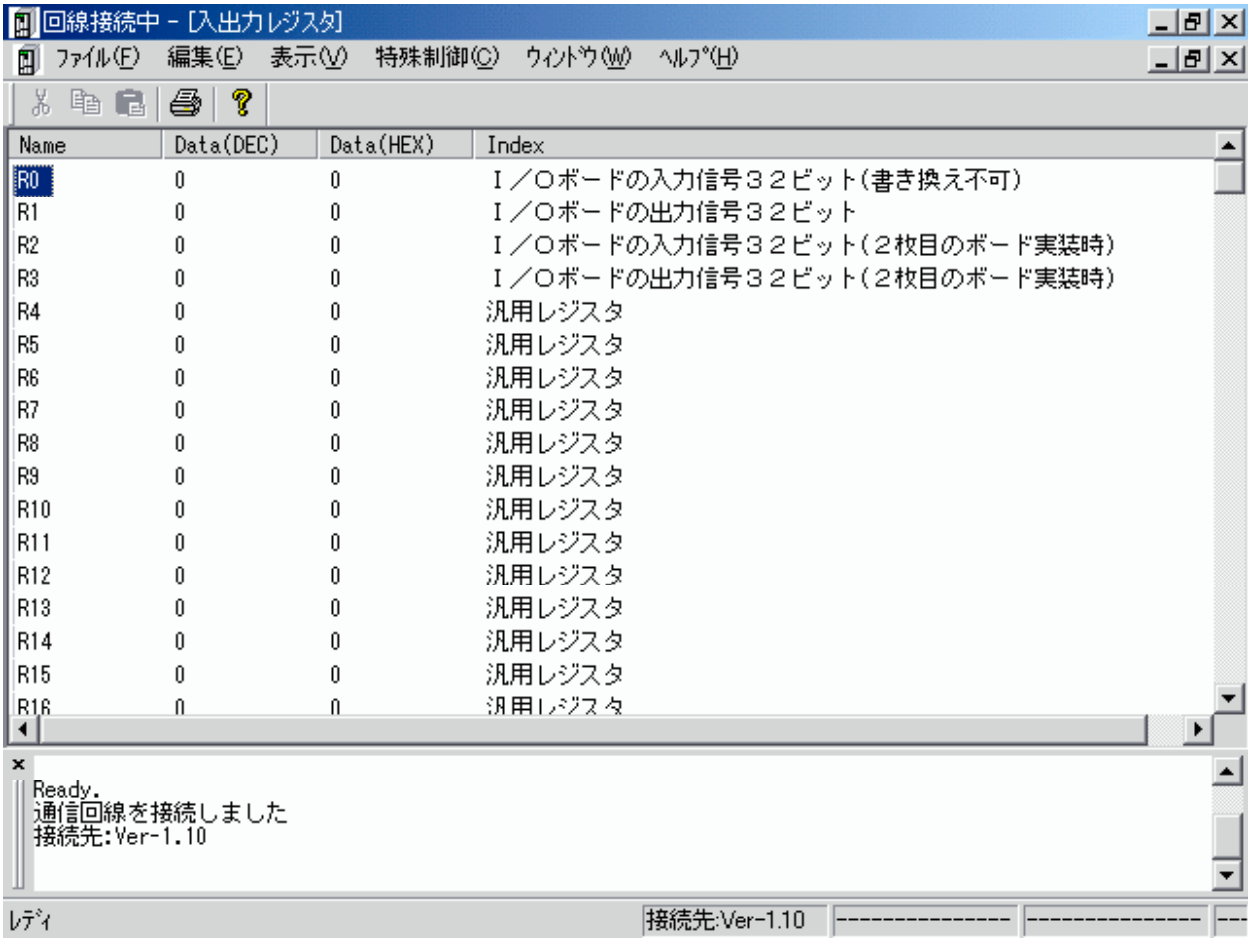



ウィンドウの項目間にライン（縦・横）を表示します
2度選択すると元に戻ります。

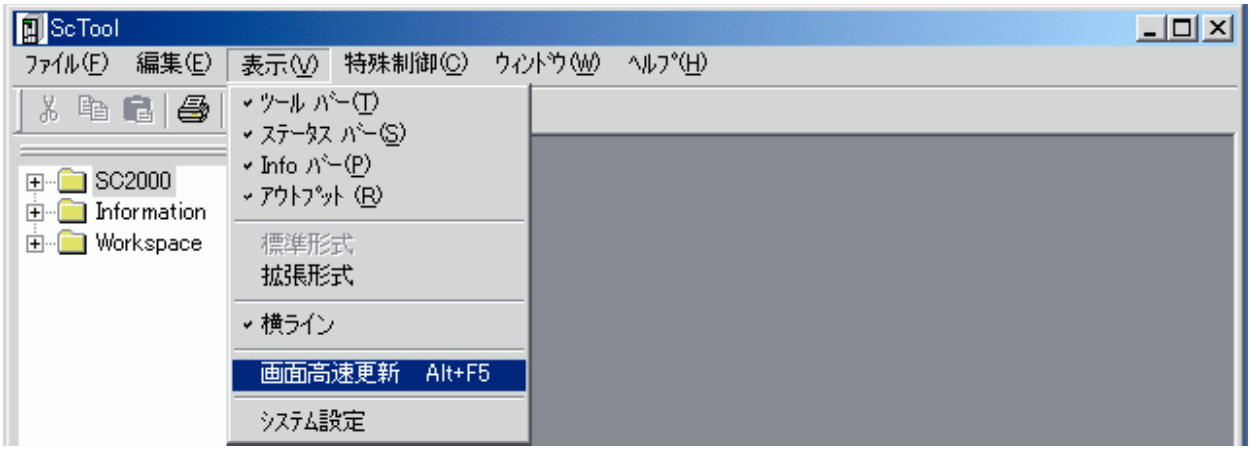
（表示例）レジスタ・ウィンドウ（システムレジスタ・横ライン表示有り）




（表示例）レジスタ・ウィンドウ（横ライン表示無し）




# 表示(V)・画面高速更新



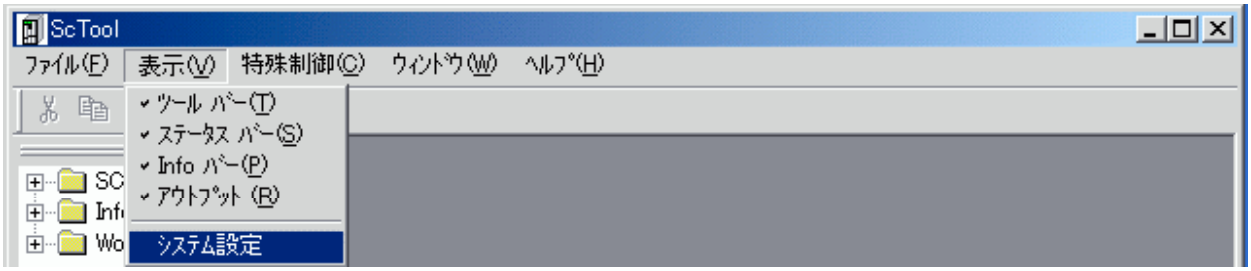

表示が自動更新されるモードの時、更新速度を変更します
メニュー左側のチェック状態（選択状態）で、更新速度が変わります(下記)
但しこの更新速度は、御使用のPC・SC2xxxにより多少変化する場合があります。

初期状態は「チェック無し（1秒に1回更新）」です

| チェック状態 | 画面更新間隔 |
| --- | --- |

| | |
|---|---|
| 無し | 1秒に1回(1000ms周期) |
| 有り | 1秒に10回(100ms周期) |

# 表示(V)・システム設定



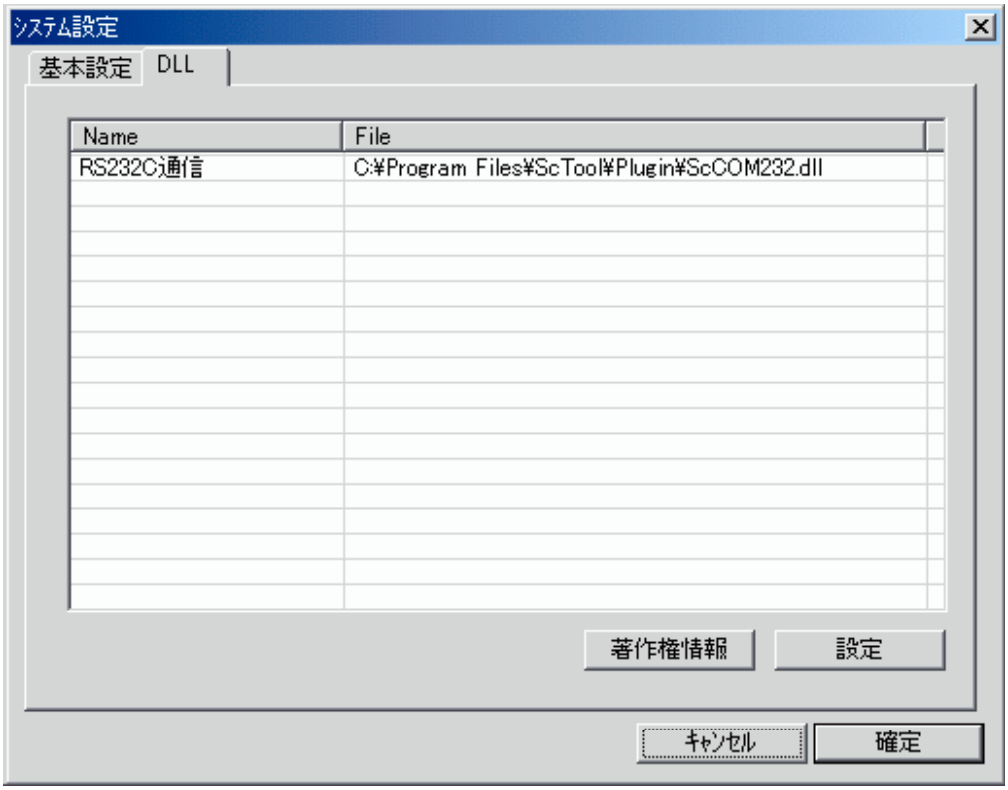

本ソフトの動作設定を行います
設定後、「確定」を押す事により確定されます


「基本設定タブ」

| 項目名 | 内容 |
|---|---|
| パラメータ表示カラム数 | パラメータ表示時の横項目数を設定します<br>(デフォルト:1列) |
| 使用通信回線 | SC2xxxと接続する通信回線種別(RS232C等)を設定します。<br>通信ポートの設定は、「通信設定」メニューか、別タブ画面(DLL)で設定する必要があります |

| | |
|---|---|
| | （デフォルト：RS232C） |
| オフセット領域 | アナログボード搭載時、アナログボードのオフセット値が格納されるレジスタ領域があります。<br>その領域に対して、「ファイル読み込み」「リストア」類で本来の値が書き換わる事を防ぐ為の設定です。<br>保護を無効にする為には、「無制限」に設定する必要があります<br>（デフォルト：保護） |
| レジスタ初期表示 | レジスタウィンドウを開いた場合に、初期表示形態を通常／拡張に設定します。<br>（デフォルト：通常） |
| 起動時ウィンドウ | 起動時にトップレベルのウィンドウを最大表示する為の設定です。<br>（デフォルト：最大表示） |
| 起動時復元 | 起動時に任意のウィンドウ状態を復元する為の設定です。<br>対象となる設定は、「ウィンドウ」メニューで保存が可能な設定１～設定４となります。<br>起動時に自動でウィンドウ設定を戻さない場合には、「復元しない」にチェックを入れる必要があります<br>(デフォルト：復元しない) |
| 文書整形（インデント） | 文書（モーション・ラダー）整形時にスペース・タブのどちらを使用するか設定します。<br>（デフォルト：スペース使用） |
| タブサイズ | 文書（モーション・ラダー）入力・整形時に使用されるタブサイズを設定します。（0を設定すると4になります）<br>（デフォルト：0） |
| 情報ウィンドウ補助 | Infoバーに表示される「レジスタ／プログラムへのショートカット」をダブルクリックした時の挙動を設定します。<br>「ウィンドウが有る場合に、そのウィンドウに移動する」又は「常にウィンドウを開く」設定が行えます。<br>（デフォルト：ウィンドウが有れば利用） |
| ウィンドウ復元時エラー | ウィンドウの復元時にエラーが発生した場合、その後の挙動を決定します。<br>（デフォルト：継続） |
| ワークスペース設定 | 「バックアップ／リストア」等を行う時にデータが格納されるフォルダを指定します。<br>（「標準」を選択した状態では、プログラムをインストールしたディレクトリに作成されます。）<br>「参照」ボタンを押す事により、任意のフォルダを指定する事が可能です。<br>（デフォルト：標準） |
| 表示用フォント名・ポイント | 各ウィンドウの表示に使用するフォント及びフォントサイズを指定します。<br>「参照」ボタンを押す事により、フォントダイアログから選択する事が可能です |
| 印刷フォント名・ポイント | 印刷に使用されるフォント及びフォントサイズを指定します。<br>「参照」ボタンを押す事により、フォントダイアログから選択する事が可能です。 |
| 印刷マージン | 印刷時の上下左右マージン（余白）をミリ単位で設定します。<br>但し、御使用のプリンタで使用出来ないマージンサイズを指定した場合は、プリンタの規定値が優先されます。 |
| | |

| | |
|---|---|
| 文書色分け | 文書(モーション・ラダー)編集時に、予約語に関して色をつけるかどうか設定します。<br>(デフォルト:使う(チェック無し)) |
| 数値編集色 | レジスタ編集時、数値入力時に表示される文字色／背景色を設定します。<br>「参照」ボタンを押すことにより、カラーダイアログから選択する事が可能です。 |
| プログラム編集色 | 文書(モーション・ラダー)編集時、特定の言葉に関して文字色／背景色を設定します。<br>「参照」ボタンを押すことにより、カラーダイアログから選択する事が可能です。 |




「DLLタブ」

| 項目名 | 内容 |
|---|---|
| 著作権情報 | 選択された拡張機能の著作権情報を表示します。 |
| 設定 | 選択された拡張機能(プラグイン)の設定画面を表示します。<br>通信回線種別に関する機能の場合、通信に使用するポートの設定等が有ります。<br>他の拡張機能に関しても設定項目が有る場合があります。<br>詳細については、各拡張機能のマニュアルを参照願います |

# 特殊制御(C)メニュー

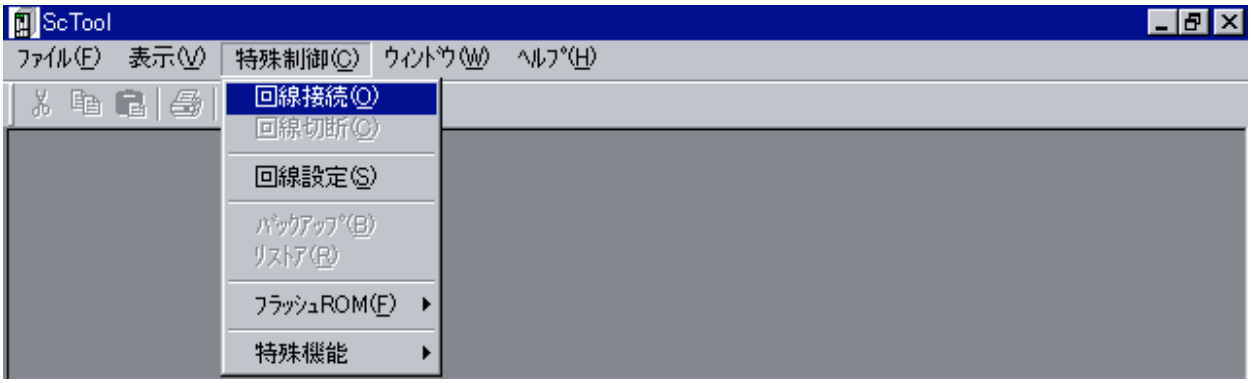
レジスタ／プログラム類ウィンドウ未選択時


レジスタ／プログラム類ウィンドウ選択時


このメニューには、下記7点の項目があります
（項目名をクリックすると、詳細説明に移動します）

回線接続
回線切断
回線設定
バックアップ
リストア
アラームリセット
フラッシュROM
特殊機能

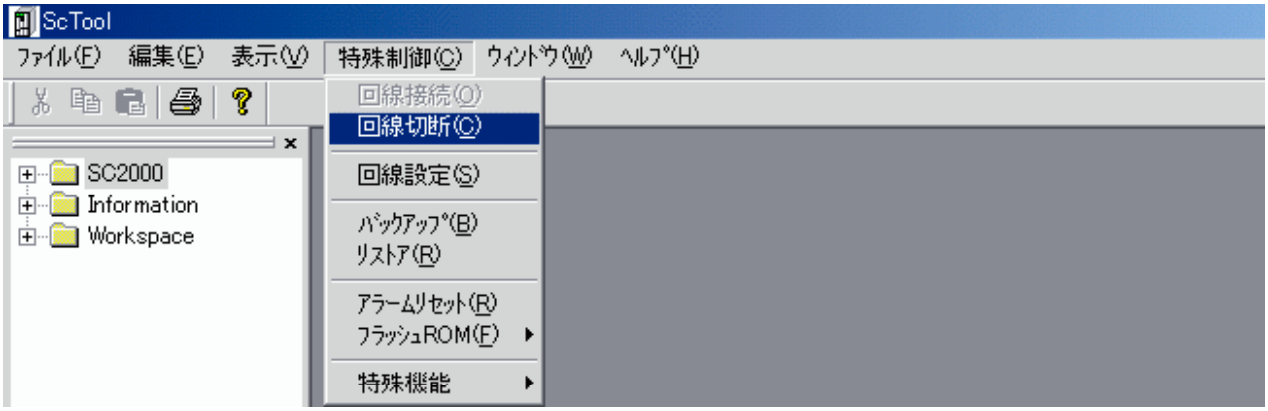
# 特殊制御(C)・回線接続(O)



通信回線を使用して、SC2xxxと接続します。
（通常、SC2xxxと接続が必要となった場合には、自動的に接続されます）
使用する回線の設定は、「システム設定」の接続先と、「回線設定」の設定が使用されますので、事前に合わせる必要が有ります。

---

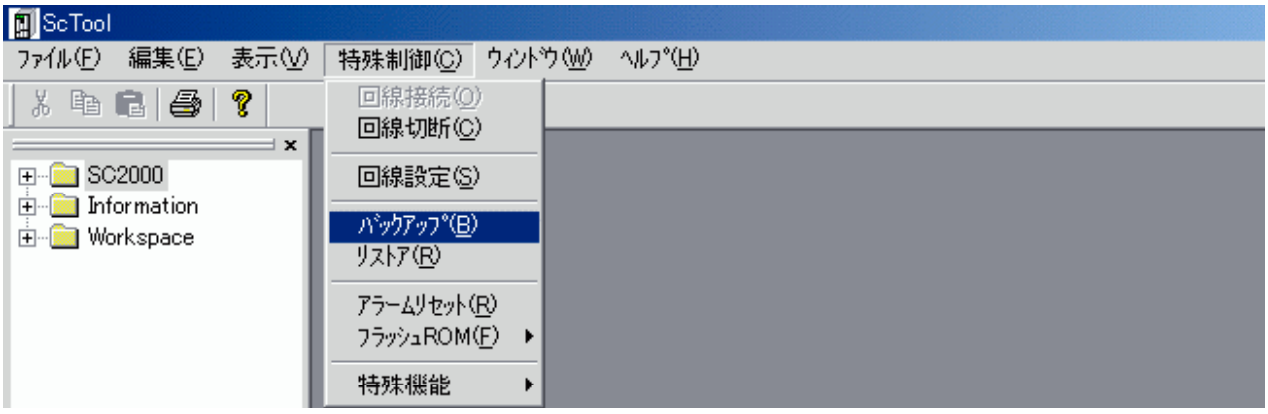
# 特殊制御(C)・回線切断(C)



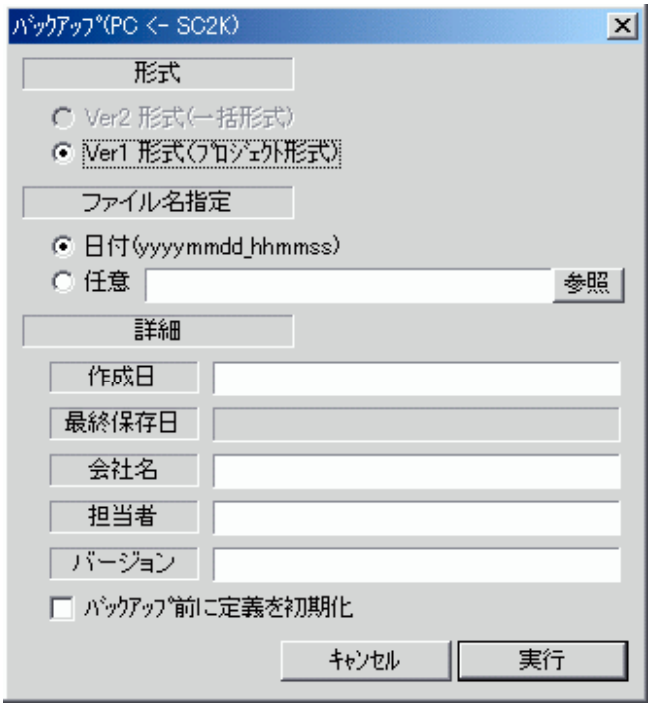

SC2xxxとの「通信回線での接続」を切断します
通信回線を明示的に切断します。以降、回線を接続（自動接続含む）するまで、SC2xxxとの通信は行えません。

---

# 特殊制御(C)・バックアップ(B)




SC2xxxから全データを取得し、ファイルとして保存します
起動すると下記ウィンドウが表示されます




形式

ver1形式

データ種類別のファイル（複数ファイル）

ver2形式

全てのデータを包括する単一ファイル（現在使用不可）

ファイル名指定

日付(yymmdd_hhmmss)

バックアップ時点の時間を元にしたファイルネームで保存します
同名のフォルダの中に作成されます。（推奨）

任意

ユーザーで任意のファイル名を付けます

詳細

作成日

最初にデータが作成された年月日を入力します（空欄可）

最終保存日

最後に保存した年月日が入力されます（自動）

会社名

作成した会社名を入力します（空欄可）

担当者

作成した担当者を入力します（空欄可）

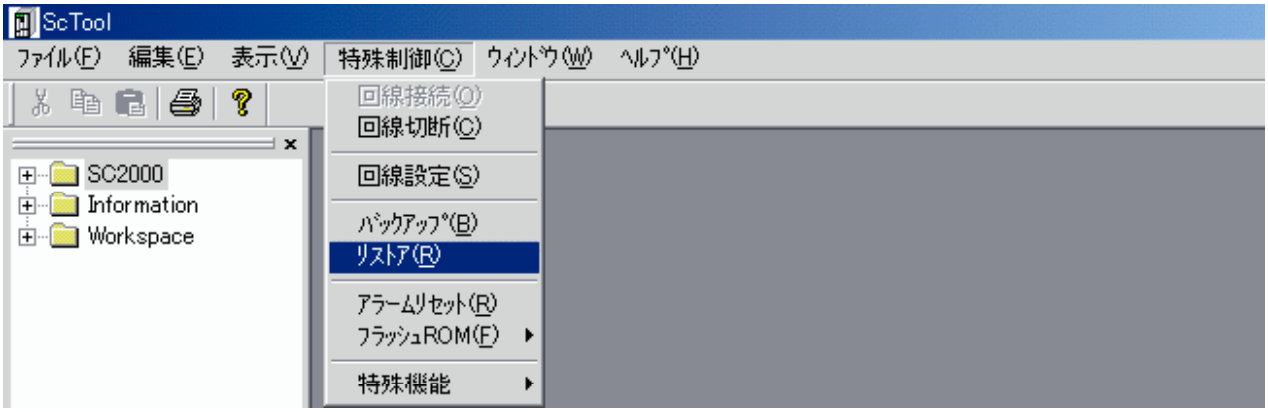
バージョン

バージョン番号を入力します（空欄可）

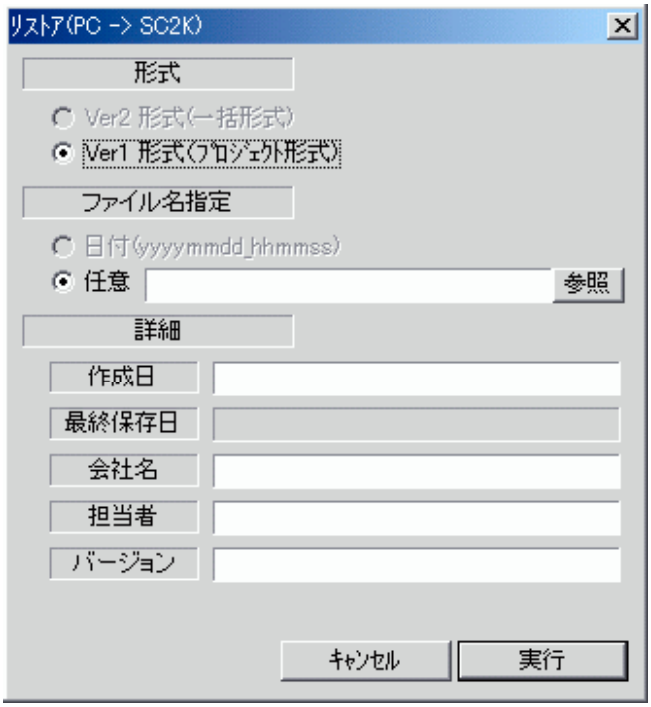
---

# 特殊制御(C)・リストア(R)

---




起動すると下記ウィンドウが表示されます
「参照」を押して、バックアップしたファイルがあるディレクトリの「*.PSC」ファイルを指定して下さい




形式

ver1形式

データ種類別のファイル（複数ファイル）

ver2形式

全てのデータを包括する単一ファイル（現在使用不可）

ファイル名指定

バックアップされたデータの中に「.PSC」という拡張子のファイルがありますので、それを指定して下さい

詳細

作成日

最初にデータが作成された年月日が表示されます

最終保存日

最後に保存した年月日が表示されます

会社名

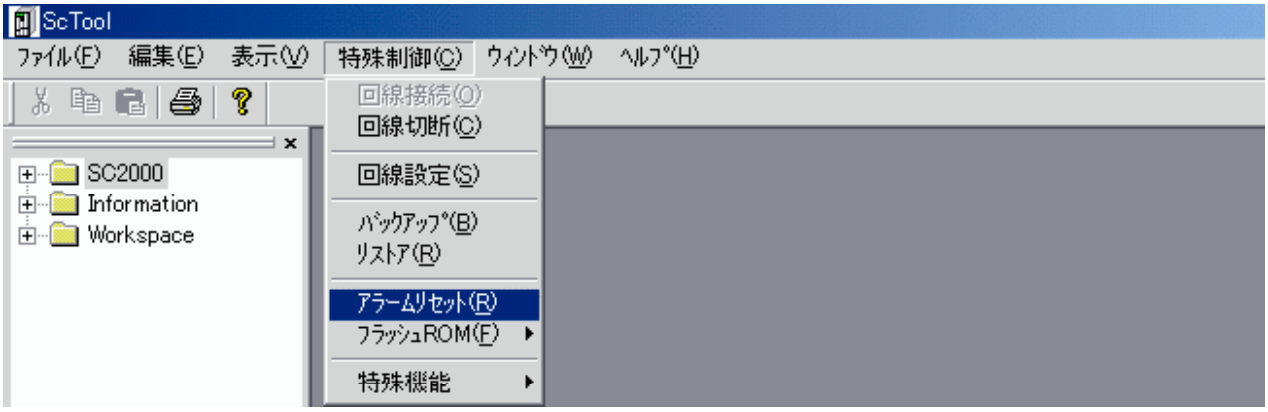
作成した会社名が表示されます

担当者
　　作成した担当者が表示されます
バージョン
　　バージョン番号が表示されます

---

# 特殊制御(C)・アラームリセット(R)

---



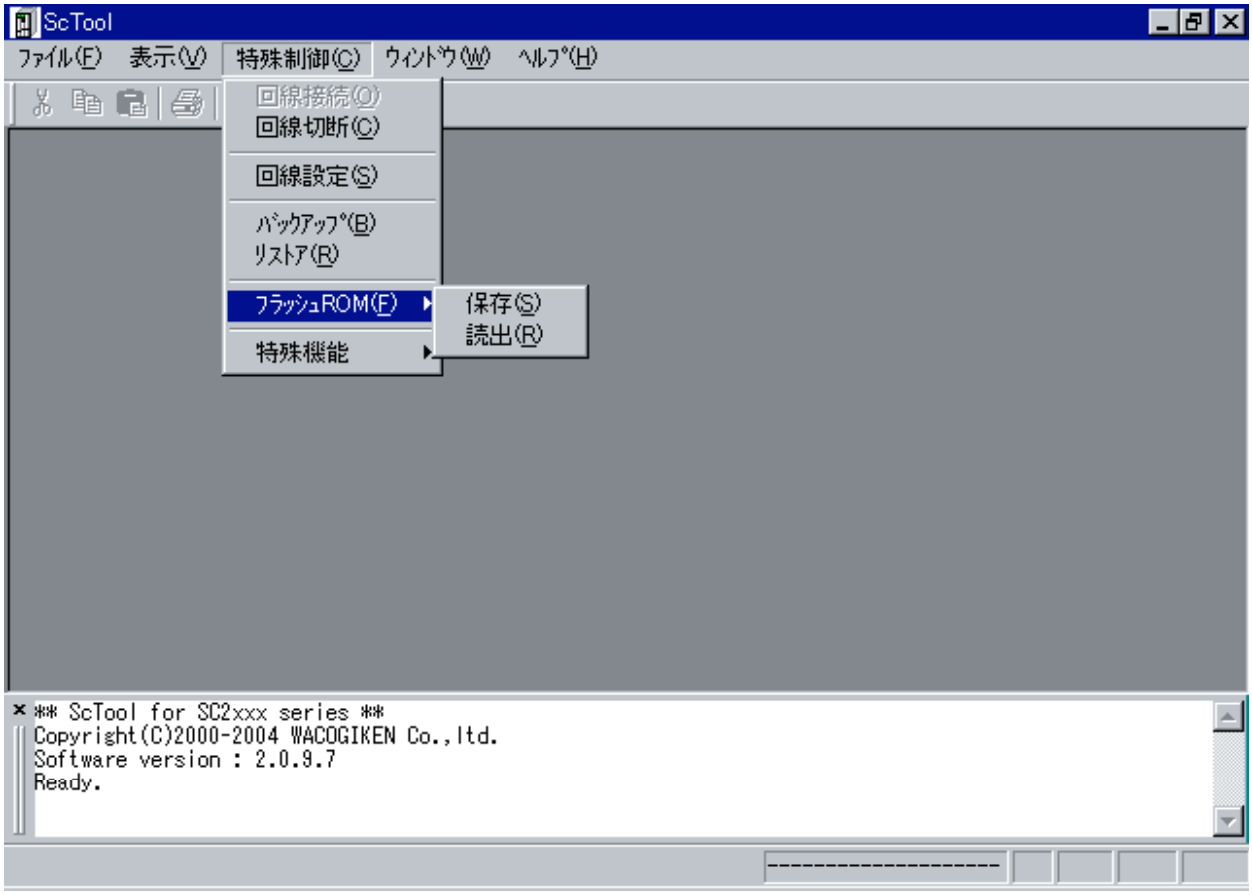

---

SC2xxxで発生しているアラームを解除します
このメニューは、通信回線が接続されている状態のみ有効となります

---

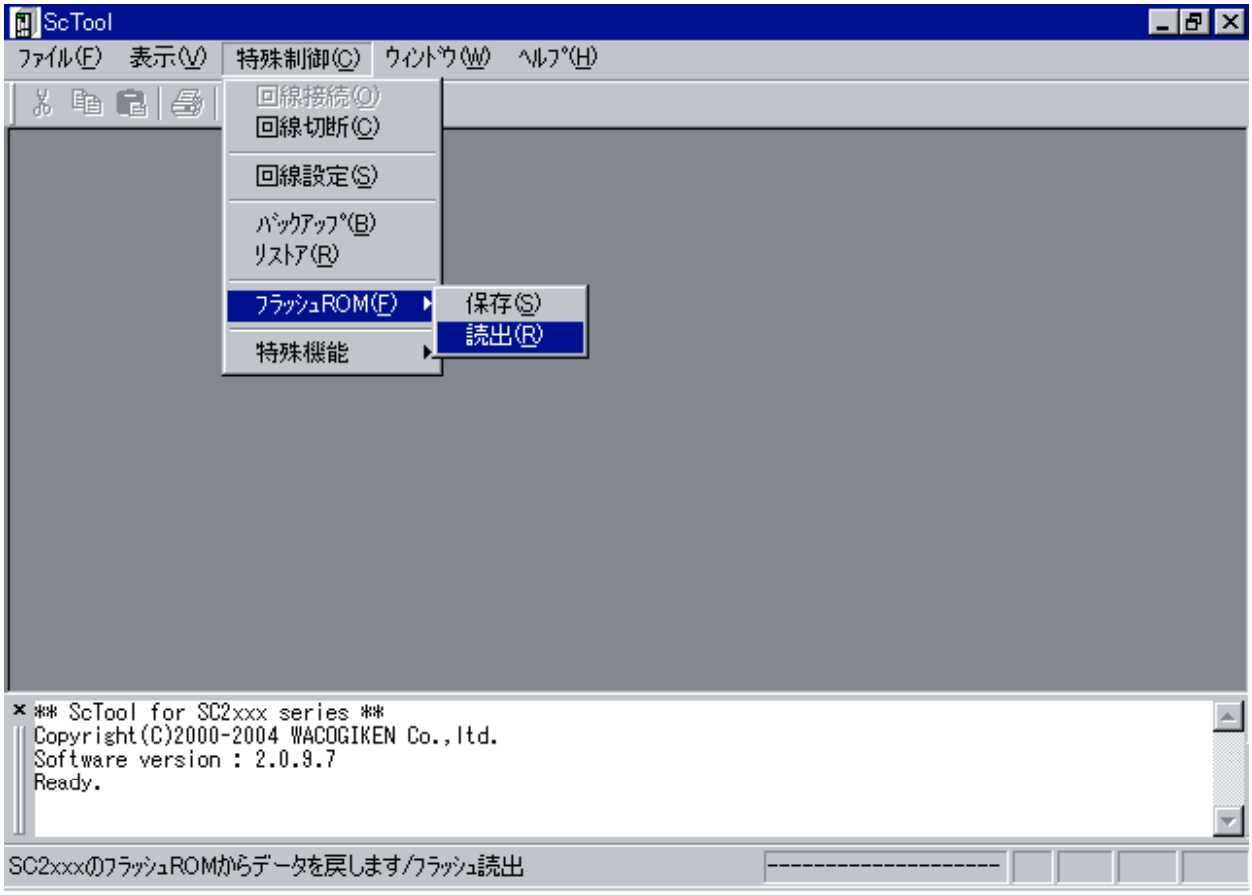
# 特殊制御(C)・フラッシュROM(F)

---

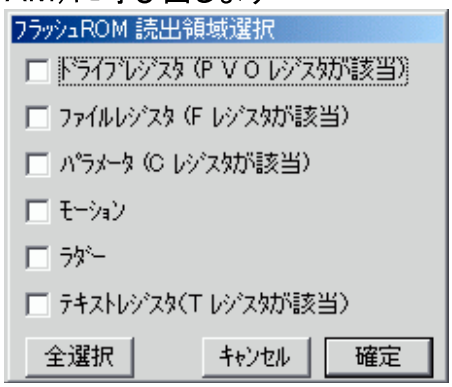



SC2xxxに搭載されているフラッシュROMを制御します
保存
読出

# 特殊制御(C)・フラッシュROM(F)・読出(R)




SC2xxxの不揮発性記憶領域（フラッシュROM）に格納されているデータを、揮発性記憶領域（RAM）に呼び出します




ドライブレジスタ

3種類のレジスタが対応します

P（位置）レジスタ
V（速度）レジスタ
O（オーダ）レジスタ

ファイルレジスタ

1種類のレジスタが対応します

F（ファイル）レジスタ

パラメータ

1種類のレジスタが対応します

C（パラメータ）レジスタ

モーション

１種類のプログラムが対応します

モーションプログラム

ラダー

３種類のプログラムが対応します

ラダープログラム：ラダーのプログラム
ラダーの定義：ラダー内にある定義文(#define)
ラダーのコメント：ラダー内にあるコメント(//で始まる行)

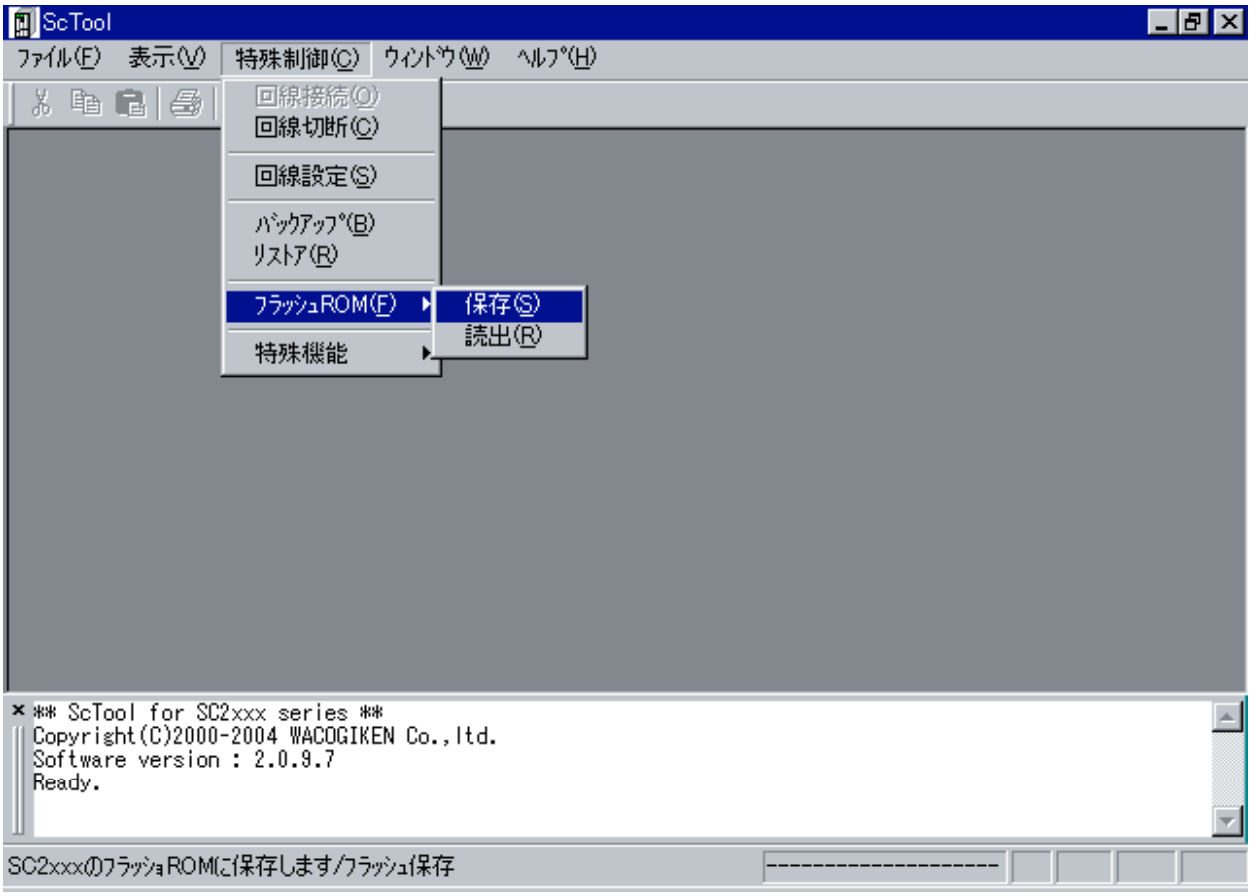
テキスト

１種類のレジスタが対応します

T（テキスト）レジスタ

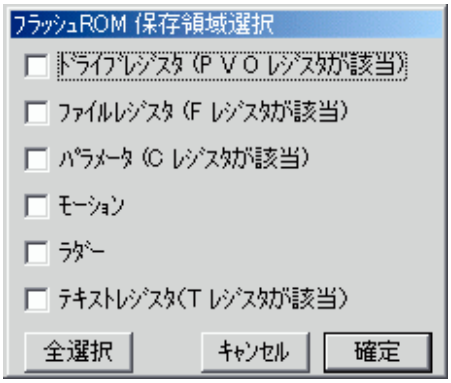
---

# 特殊制御(C)・フラッシュROM(F)・保存(S)




---

SC2xxxの揮発性記憶領域（RAM）に格納されているデータを、不揮発性記憶領域（フラッシュROM）に保存します

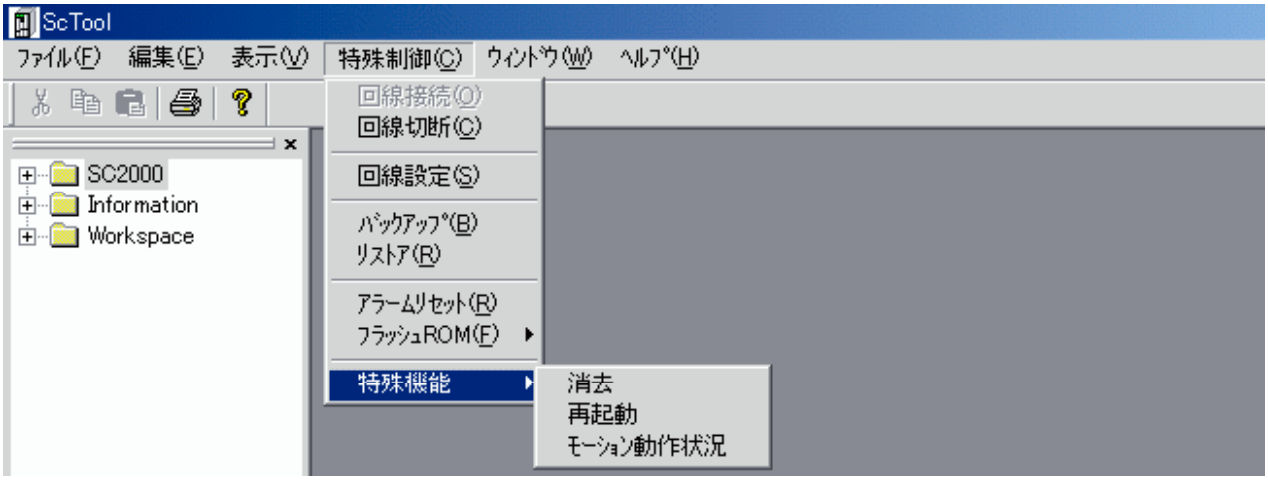



ドライブレジスタ
　3種類のレジスタが対応します
　　P(位置)レジスタ
　　V(速度)レジスタ
　　O(オーダ)レジスタ
ファイルレジスタ
　1種類のレジスタが対応します
　　F(ファイル)レジスタ
パラメータ
　1種類のレジスタが対応します
　　C(パラメータ)レジスタ
モーション
　1種類のプログラムが対応します
　　モーションプログラム
ラダー
　3種類のプログラムが対応します
　　ラダープログラム:ラダーのプログラム
　　ラダーの定義:ラダー内にある定義文(#define)
　　ラダーのコメント:ラダー内にあるコメント(//で始まる行)
テキスト
　1種類のレジスタが対応します
　　T(テキスト)レジスタ

---

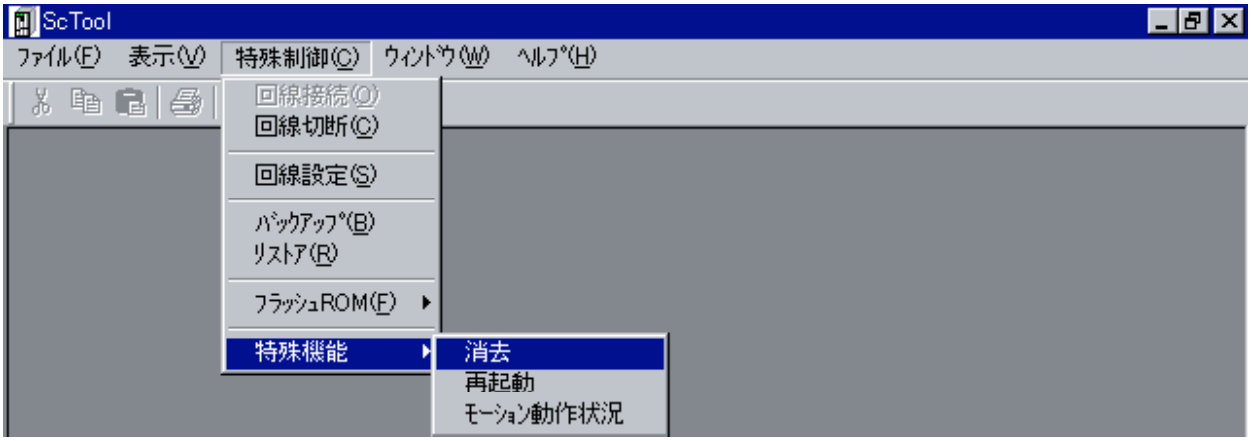
# 特殊制御(C)・特殊機能

---

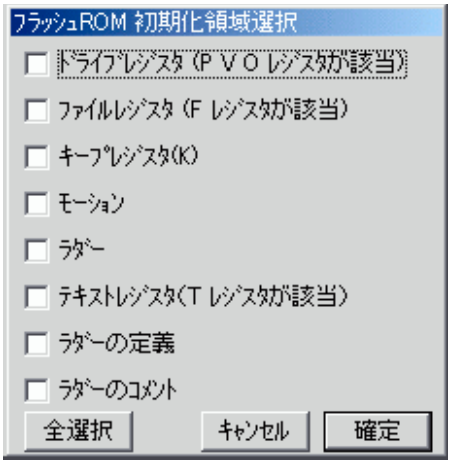



SC2xxxの特殊機能を使用します。
（SC2xxxのファームウェアバージョンにより、使用出来る機能は異なります）
消去
再起動
モーション動作状況

# 特殊制御(C)・特殊機能・消去




SC2xxxの不揮発性記憶領域に格納されているデータの内、指定する領域を初期化します。
メニューを選択すると、下記ウィンドウが表示されます



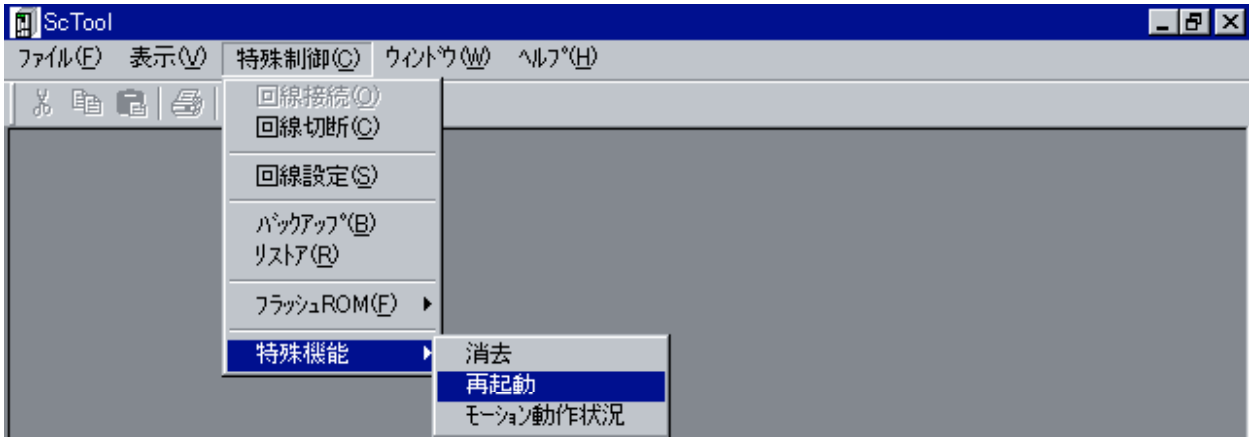
ドライブレジスタ
　３種類のレジスタが対応します
　　Ｐ（位置）レジスタ
　　Ｖ（速度）レジスタ
　　Ｏ（オーダ）レジスタ
ファイルレジスタ
　１種類のレジスタが対応します
　　Ｆ（ファイル）レジスタ
パラメータ
　１種類のレジスタが対応します
　　Ｃ（パラメータ）レジスタ
モーション
　１種類のプログラムが対応します
　　モーションプログラム
ラダー
　３種類のプログラムが対応します
　　ラダーのプログラム
　　ラダー内にある定義文(#define)
　　ラダー内にあるコメント(//で始まる行)
テキスト
　１種類のレジスタが対応します
　　Ｔ（テキスト）レジスタ
ラダーの定義
　１種類のプログラムが対応します
　　ラダー内にある定義文(#define)
　　(ファームウェアのバージョンによって使えない場合があります)
ラダーのコメント
　１種類のプログラムが対応します
　　ラダー内にあるコメント(//で始まる行)
　　(ファームウェアのバージョンによって使えない場合があります)

チェックを決定後、「確定」を押す事により、初期化を行います。

---

# 特殊制御(C)・特殊機能・再起動

---



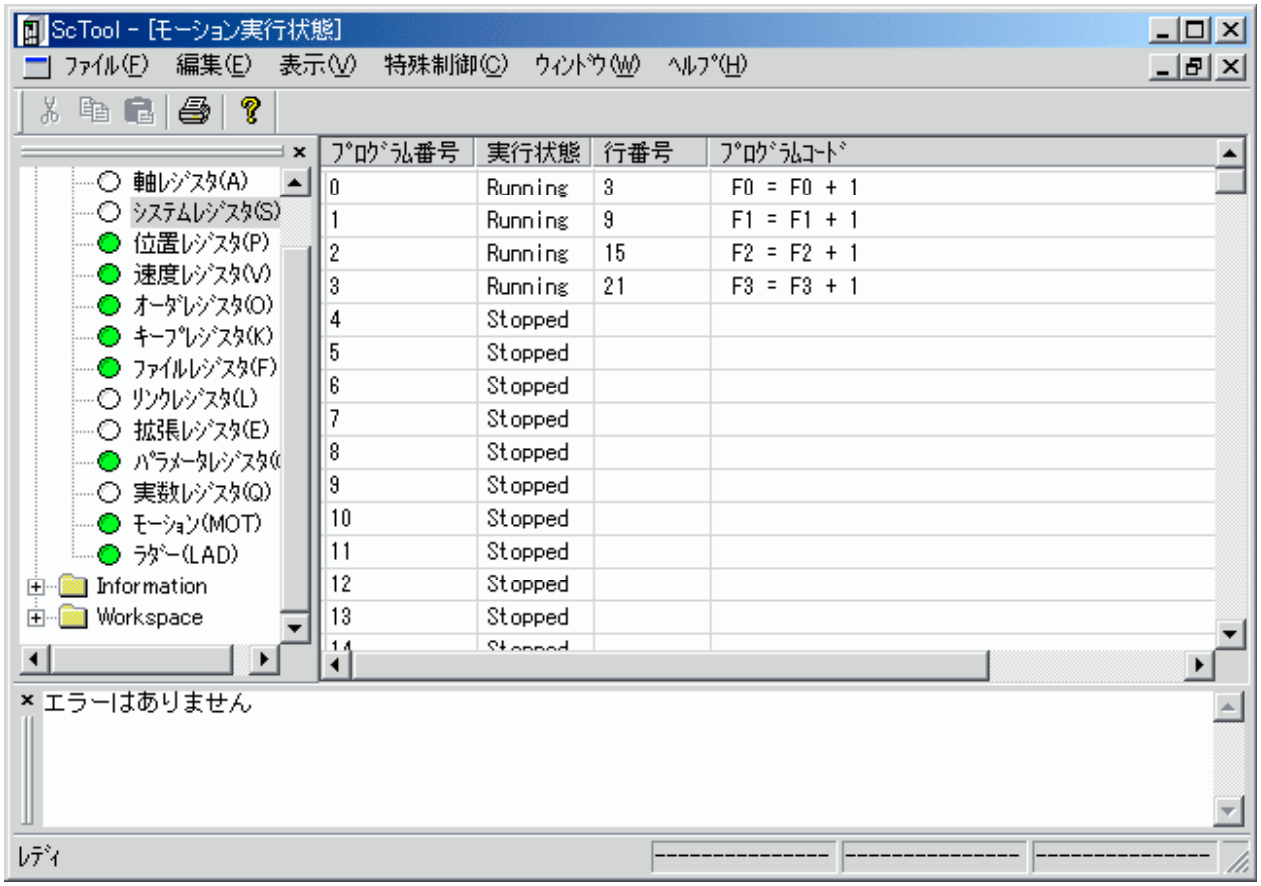

現在接続しているSC2xxxを再起動（電源再投入と同じ）を行います。

実行すると下記の状態になります

変更されているレジスタは、フラッシュROMに記録されている値に戻ります
変更した事でついた「変更済み表示」は、「未変更」時の状態に戻ります
ネットワークボード搭載時、IP設定が「DHCPサーバからの取得」の場合は再取得されます

# 特殊制御(C)・特殊機能・モーション動作状況


SC2xxxで動作中のモーション状態を取得します。
メニューを選択すると、モーションを読み込んだ後、下記ウィンドウ（表示内容は一例）が表示されます
（表示迄に時間が掛かる場合があります）




各個所の説明

プログラム番号

モーションのプログラム番号に該当します

実行状態

モーションのプログラム状態を表示します。

「Running」. . . 実行中
「Stopped」. . . 停止中

行番号

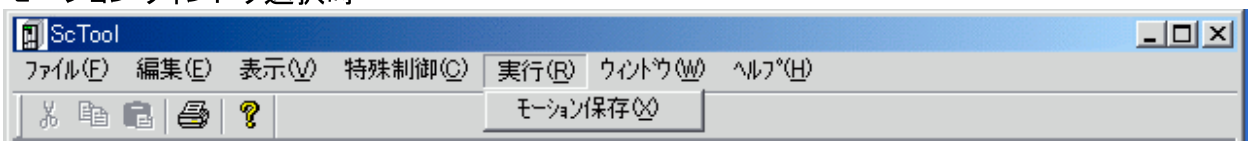
モーションの実行中行番号を表示します

「プログラムコード」

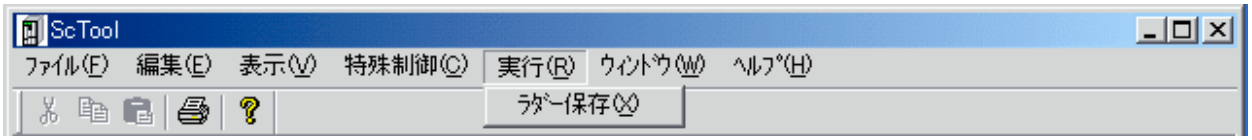
実行中のプログラムコードを表示します

# 実行(R)メニュー

モーションウィンドウ選択時




ラダーウィンドウ選択時




このメニューには、下記2点の項目があります

[モーション保存](#)
[ラダー保存](#)

# 実行(X)・モーション保存

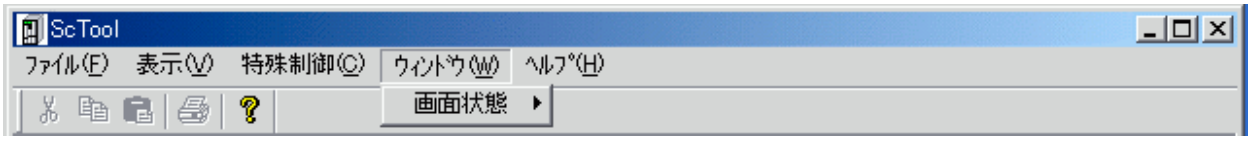

編集中のモーションを、SC2xxxに転送します。
実行時にモーションのエラーチェックも行われ、結果は「アウトプットウィンドウ」(リザルトウィンドウ)に表示されます。

# 実行(X)・ラダー保存(X)

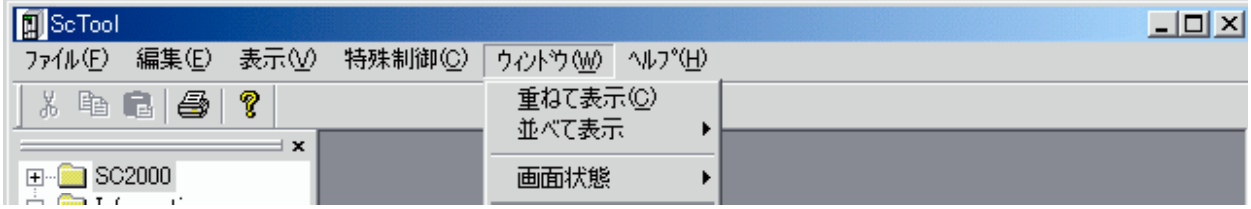

編集中のラダーを、SC2xxxに転送します。
実行時にラダーのエラーチェックも行われ、結果は「アウトプットウィンドウ」(リザルトウィンドウ)に表示されます。

# ウィンドウ(W)メニュー

レジスタ／プログラム類ウィンドウ未選択時




レジスタ／プログラム類ウィンドウ選択時



このメニューには、下記３点の項目があります
（項目名をクリックすると、詳細説明に移動します）

重ねて表示
並べて表示
画面状態

# ウィンドウ(W)・重ねて表示(C)


開いているウィンドウを重ねて表示します。

# ウィンドウ(W)・並べて表示

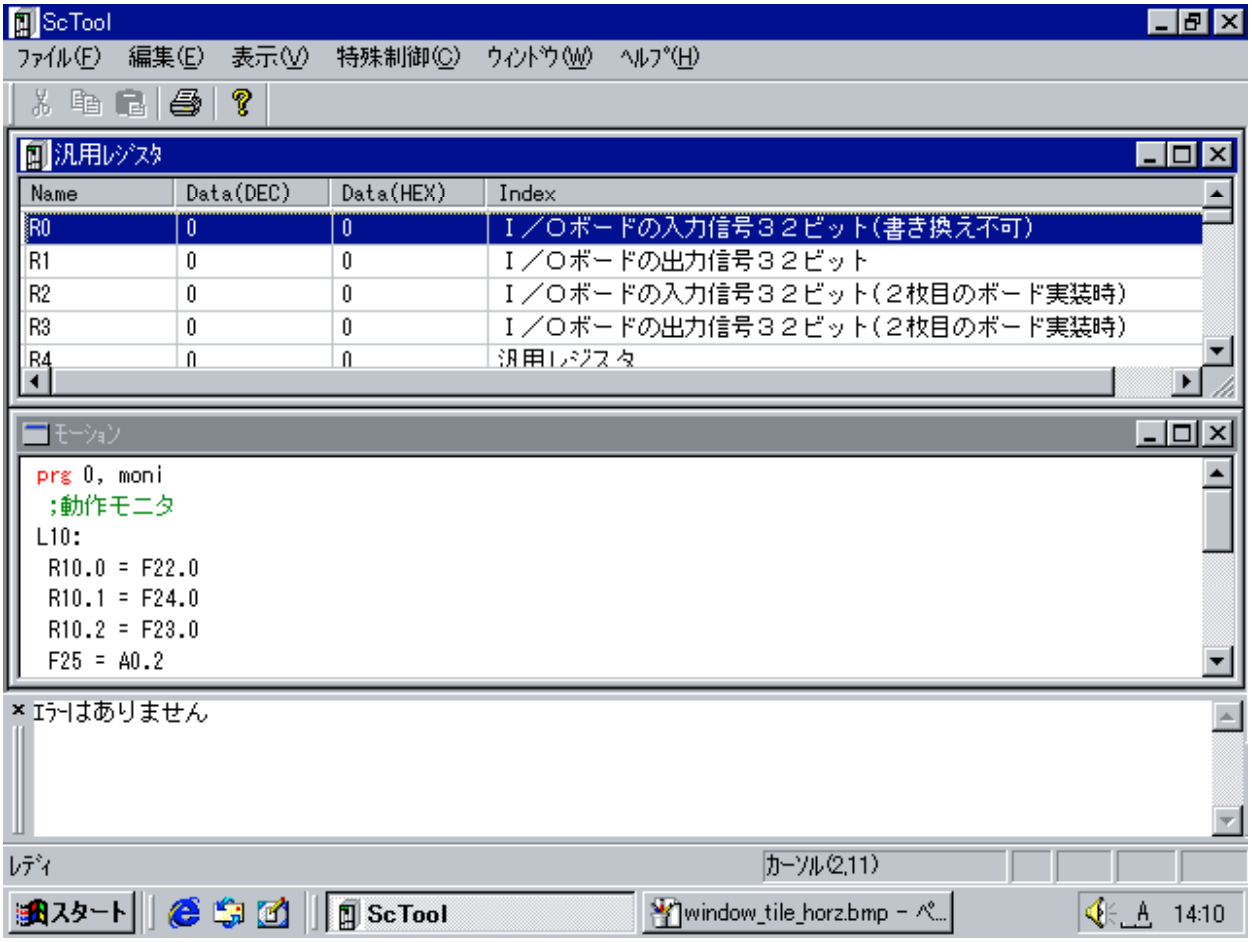

このメニューには、下記２点の項目があります

縦方向
横方向

# ウィンドウ(W)・並べて表示・縦方向


開いているウィンドウを縦方向に並べます

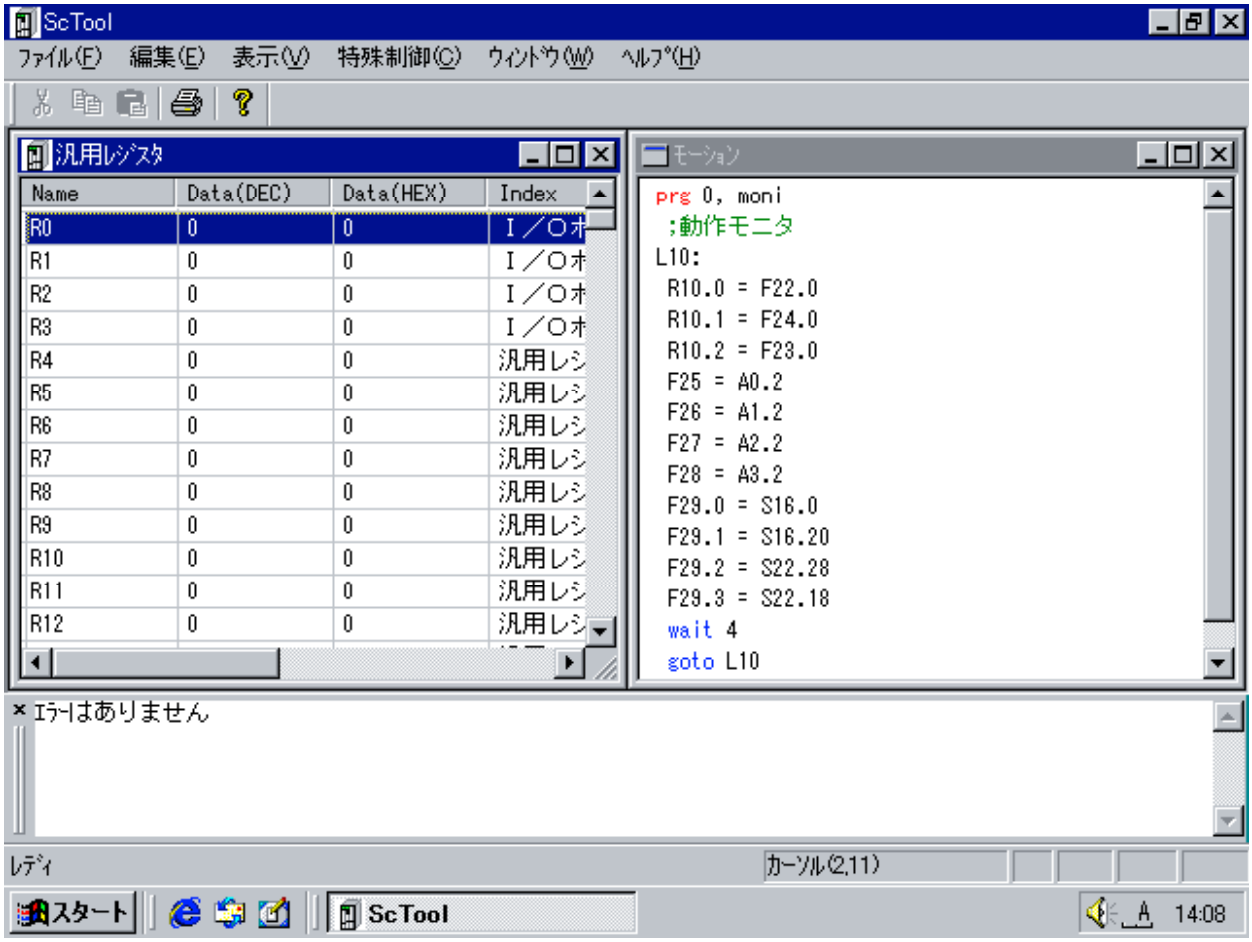
（実行例）



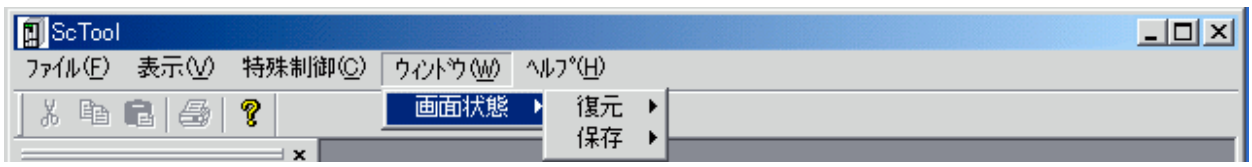

# ウィンドウ(W)・並べて表示・横方向


開いているウィンドウを横方向に並べます

（実行例）




# ウィンドウ(W)・画面状態




このメニューには、下記２点の項目があります

復元
保存

# ウィンドウ(W)・画面状態・復元


このメニューには、下記４点の項目があります

設定１
　以前に「設定１」に保存されたウィンドウ位置設定を復元します
設定２
　以前に「設定２」に保存されたウィンドウ位置設定を復元します
設定３
　以前に「設定３」に保存されたウィンドウ位置設定を復元します
設定４
　以前に「設定４」に保存されたウィンドウ位置設定を復元します

「復元についての詳細」
　復元されるのは、下記の設定のみです。

　ウィンドウのモード（新規作成で選択したモード）
　ウィンドウ位置

　（復元されるウィンドウが、「通信」を必要とする場合には、自動で接続処理を行います。
　但し、接続処理が失敗した場合は、そのウィンドウは復元されません。
　又、「モーション」「ラダー」の復元がある場合に、ウィンドウを復元する途中で、編集するデータに関しての問い合わせが表示されます。
　（次に進むには、その問い合わせに対して、答えを選択する必要があります）

# ウィンドウ(W)・画面状態・保存


このメニューには、下記４点の項目があります

設定１
　「設定１」に現在のウィンドウ位置設定を保存します
設定２
　「設定２」に現在のウィンドウ位置設定を保存します
設定３

「設定3」に現在のウィンドウ位置設定を保存します

設定4

「設定4」に現在のウィンドウ位置設定を保存します

「保存についての詳細」

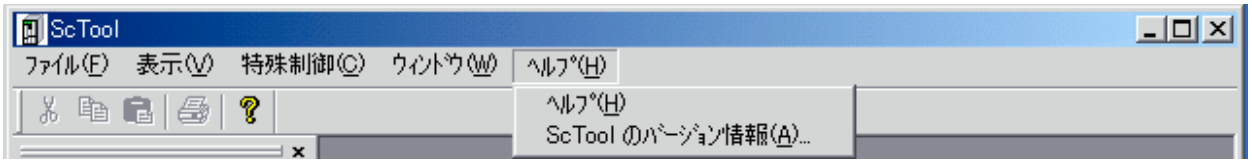
保存されるのは、下記の設定のみです。

ウィンドウのモード（新規作成で選択したモード）
ウィンドウ位置
最小化・最大化状態

注意！ ウィンドウ内のデータが保存されるわけではありません

---

# ヘルプ(H)メニュー

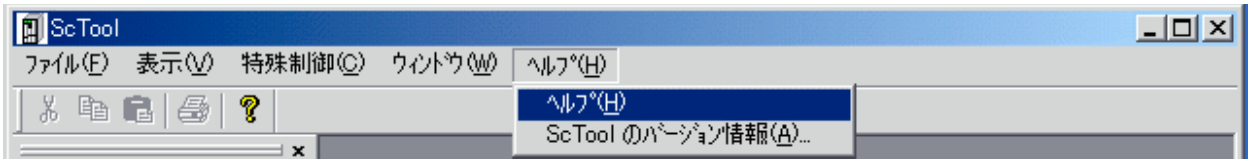



このメニューには、下記2点の項目があります

ヘルプ
ScToolのバージョン表示／バージョンアップ

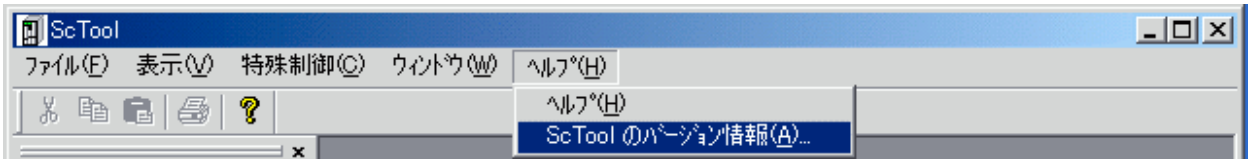
---

# ヘルプ(H)・ヘルプ(H)

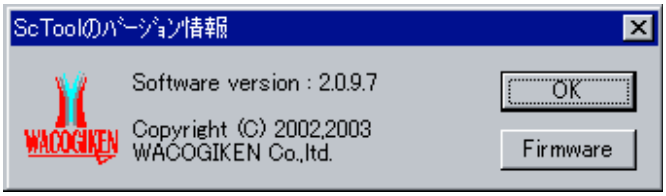



オンラインヘルプを表示します

# ヘルプ(H)・ScToolのバージョン情報(A)




本ツールのバージョン番号を表示します

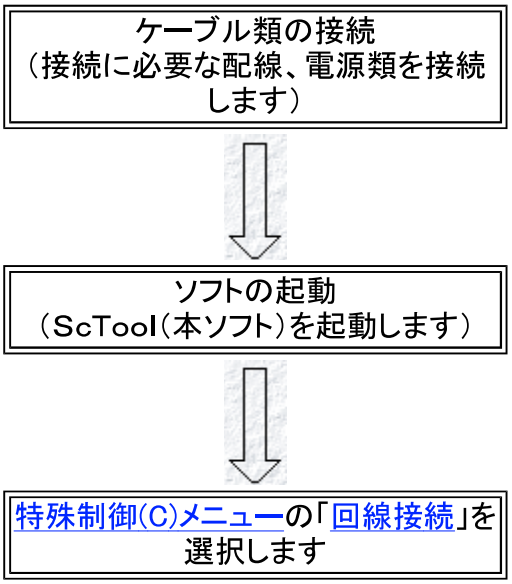



「OK」を押すとバージョン表示を閉じます
「Firmware」を押すとSC2xxx搭載のファームウェアアップデートが行えます

---

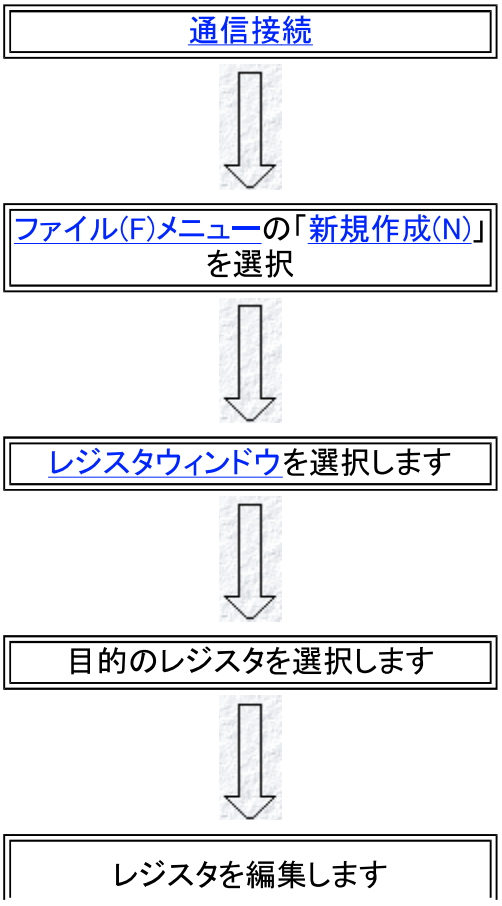
# 目的別：通信接続まで

初期状態から通信接続迄の手順




# 目的別：レジスタ編集

初期状態からレジスタ編集終了迄の手順



（基本操作方法はこちら）

編集が終了したらウィンドウを閉じます

必要に応じて、フラッシュROMに保存します
（保存方法はこちら）

---

# 目的別：パラメータ編集

---

初期状態からパラメータ編集終了迄の手順

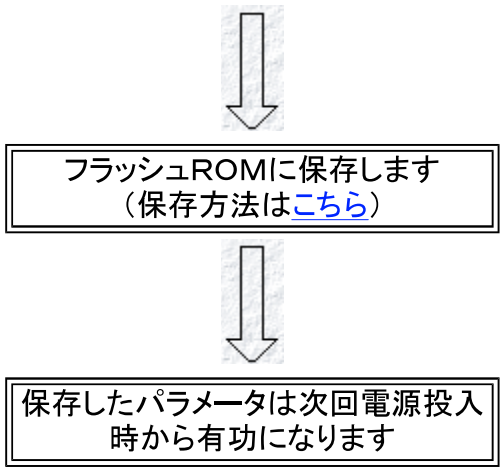
通信接続

ファイル(F)メニューの「新規作成(N)」を選択

パラメータウィンドウを選択します

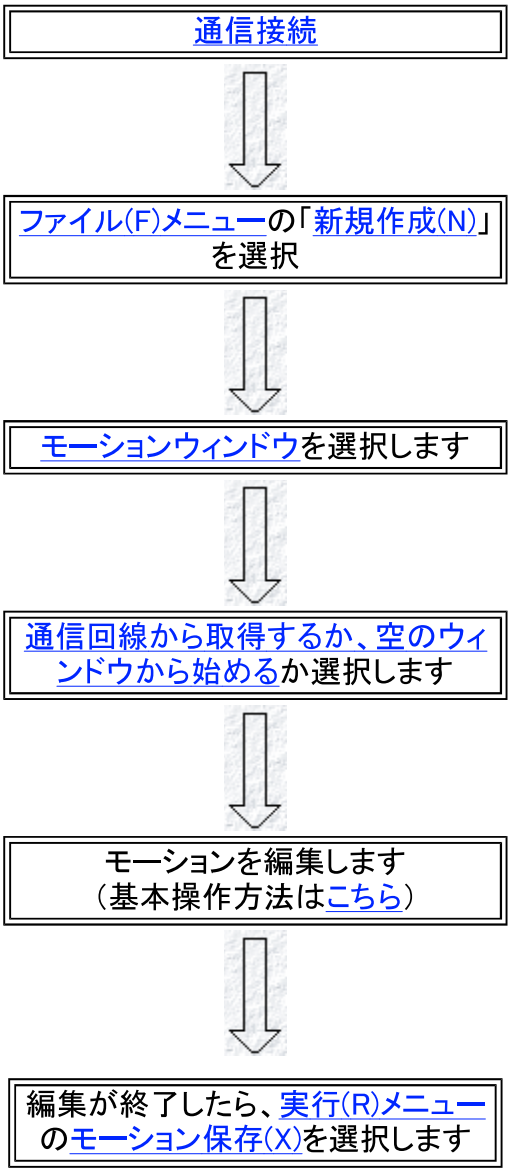
パラメータを編集します
（基本操作方法はこちら）

編集が終了したらウィンドウを閉じます




# 目的別：モーション編集

初期状態からモーション編集終了迄の手順



リザルトウィンドウにエラーの有無が表示されます。
エラーがある場合には、再度編集します

必要に応じて、フラッシュROMに保存します
（保存方法はこちら）

ウィンドウを閉じて、終了します

# 目的別：ラダー編集

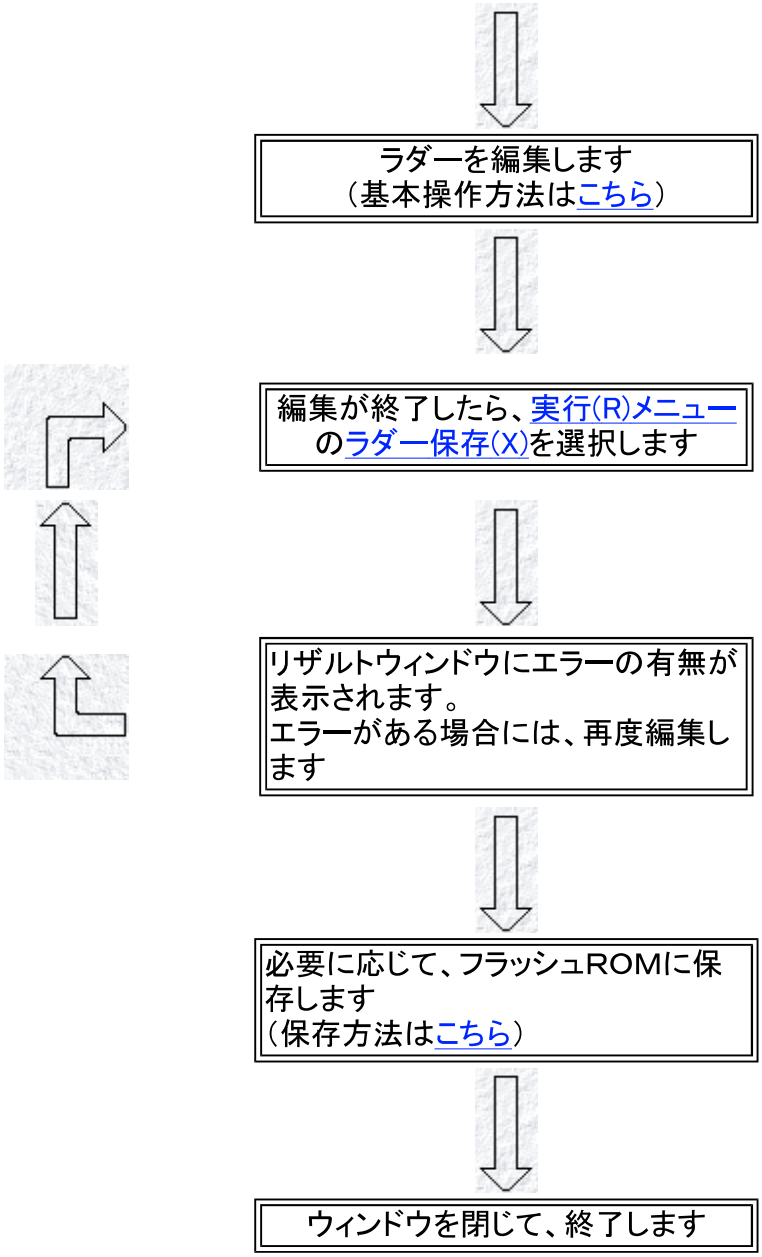
初期状態からラダー編集終了迄の手順

通信接続

ファイル(F)メニューの「新規作成(N)」を選択

ラダーウィンドウを選択します

通信回線から取得するか、空のウィンドウから始めるか選択します


ラダーを編集します
（基本操作方法はこちら）
編集が終了したら、実行(R)メニューのラダー保存(X)を選択します
リザルトウィンドウにエラーの有無が表示されます。
エラーがある場合には、再度編集します
必要に応じて、フラッシュROMに保存します
（保存方法はこちら）
ウィンドウを閉じて、終了します

# リファレンス選択

モーションで使用できる関数・コマンド一覧
アルファベット順一覧
機能別一覧
ラダーで使用できる関数・コマンド一覧
アルファベット順一覧
機能別一覧

# リファレンス選択

モーションで使用できる関数・コマンド一覧
アルファベット順一覧
機能別一覧

# リファレンス一覧（モーション）

モーションで使用できる関数・コマンド一覧（アルファベット順）

| アルファベット順 | | | |
|---|---|---|---|
| 代入文 | コメント文 | abs | acos |
| asin | atan | atan2 | ceil |
| close | cos | else | elseif |
| endif | float | floor | getcom |
| getl | gosub | goto | if |
| int | movc | movci | movh |
| movm | movmi | movo | movs |
| movsi | movt | movti | open |
| padj | PI | PI2 | pow |
| prg | print | prio | pset |
| putcom | putl | reset | return |
| scan | seplh | sin | sqrt |
| stklh | stop | sub | tan |
| wait | | | |

# リファレンス一覧（機能別一覧）

モーションで使用できる関数・コマンド一覧（機能別）

| プログラム制御 | | | |
|---|---|---|---|
| prg | sub | if | elseif |
| else | endif | goto | gosub |
| return | wait | stop | prio |
| **駆動/軸関係** | | | |
| movh | movs | movsi | movm |
| movmi | movc | movci | movo |
| movt | movti | pset | padj |
| **エラー/その他** | | | |
| reset | 代入文 | コメント文 | |
| **関数** | | | |
| sin | cos | tan | asin |
| cos | tan | tan2 | sqrt |
| pow | ceil | floor | float |
| int | abs | PI | PI2 |
| printf | scan | seplh | stklh |
| **通信関係** | | | |
| open | close | getl | putl |
| getcom | putcom | | |

# 代入文

書式

<reg>=<expr>

説明

右辺の計算式の結果を左辺のレジスタに代入します。
左辺、右辺ともワード指定、ビット指定が可能で、左辺がビット指定、右辺がワード指定の場合右辺の最下位ビットが代入されます

# コメント文

書式

;（セミコロン）<remarks>

説明

;（セミコロン）の後に任意の文字列を書くことが出来ます。

この行は実行時には何もしません。

---

# #define

---

書式

#define <label> <arg>

説明

<arg>を別名<label>で定義します。
定義した行以降 で、別名が使用可能となります。
SC2xxxから読み出す時に「逆変換」されている為、逆変換すると困る様な定義は御止めください

---

# #include

---

書式

#include <filename>

説明

<filename>のファイルをその位置に展開します。
（この命令は、現在動作いたしません）

---

# abs()

---

書式

abs(<expr>)

説明

<expr>の絶対値を計算します。<expr>が整数の場合結果は整数、実数の場合実数を返します。

---

# asin(),acos(),atan(),atan2()

---

書式

asin(<expr>)
acos(<expr>)
atan(<expr>)
atan2(<expr:y>,<expr:x>)

説明

<expr>のasin,acos,atanの計算を行います。角度の表現はradです。
atan2は<expr:y>,<expr:x>の2つの因数から計算され、結果は±πで得られます。

結果は実数を返します。

---

# asin(),acos(),atan(),atan2()

---

書式

asin(<expr>)
acos(<expr>)
atan(<expr>)
atan2(<expr:y>,<expr:x>)

説明

<expr>のasin,acos,atanの計算を行います。角度の表現はradです。
atan2は<expr:y>,<expr:x>の２つの因数から計算され、結果は±πで得られます。
結果は実数を返します。

---

# asin(),acos(),atan(),atan2()

---

書式

asin(<expr>)
acos(<expr>)
atan(<expr>)
atan2(<expr:y>,<expr:x>)

説明

<expr>のasin,acos,atanの計算を行います。角度の表現はradです。
atan2は<expr:y>,<expr:x>の２つの因数から計算され、結果は±πで得られます。
結果は実数を返します。

---

# asin(),acos(),atan(),atan2()

---

書式

asin(<expr>)
acos(<expr>)
atan(<expr>)
atan2(<expr:y>,<expr:x>)

説明

<expr>のasin,acos,atanの計算を行います。角度の表現はradです。
atan2は<expr:y>,<expr:x>の２つの因数から計算され、結果は±πで得られます。
結果は実数を返します。

---

# ceil(),floor()

書式

ceil(<expr>)
floor(<expr>)

説明

<expr>の小数点以下の、ceilは切り捨て、floorは切り上げを行います。
結果は実数を返します。

---

# close

---

書式

close <FileNo>

説明

open命令で開いたファイルを閉じます。
<FileNo>には、open命令を使用したときの返り値(変数RES)の値を指定します。

---

# sin(),cos(),tan()

---

書式

sin(<expr>)
cos(<expr>)
tan(<expr>)

説明

<expr>のsin,cos,tanの計算を行います。角度の表現はradです。
結果は実数を返します。

---

# if,elseif,else,endif

---

書式

if <expr>
elseif <expr>
else
endif

説明

ifから始まりendifで終了する条件命令です。
間にelseif,elseをはさむ事によって条件から1つを選択する事が出来ます。

例

if <expr1>
・・・・・ <expr1>が真なら実行します
elseif <expr2>
・・・・・ <expr2>が真なら実行します

elseif <expr3>
・・・・・ <expr3>が真なら実行します
else
・・・・・ <expr1><expr2><expr3>いずれも真で無いなら実行します
endif

---

# if,elseif,else,endif

---

書式

if <expr>
elseif <expr>
else
endif

説明

ifから始まりendifで終了する条件命令です。
間にelseif,elseをはさむ事によって条件から1つを選択する事が出来ます。

例

if <expr1>
・・・・・ <expr1>が真なら実行します
elseif <expr2>
・・・・・ <expr2>が真なら実行します
elseif <expr3>
・・・・・ <expr3>が真なら実行します
else
・・・・・ <expr1><expr2><expr3>いずれも真で無いなら実行します
endif

---

# if,elseif,else,endif

---

書式

if <expr>
elseif <expr>
else
endif

説明

ifから始まりendifで終了する条件命令です。
間にelseif,elseをはさむ事によって条件から1つを選択する事が出来ます。

例

if <expr1>
・・・・・ <expr1>が真なら実行します
elseif <expr2>
・・・・・ <expr2>が真なら実行します
elseif <expr3>
・・・・・ <expr3>が真なら実行します
else

・・・・・ <expr1><expr2><expr3>いずれも真で無いなら実行します
endif

---

# float(),int()

---

書式

float(<expr>)
int(<expr>)

説明

floatは<expr>を実数に型変換します。
intは<expr>は整数に型変換します。

---

# ceil(),floor()

---

書式

ceil(<expr>)
floor(<expr>)

説明

<expr>の小数点以下の、ceilは切り捨て、floorは切り上げを行います。
結果は実数を返します。

---

# getcom

---

書式

getcom <port>,<timeoutMS>

説明

<port>で指定された通信ポートから1バイト取得します。
<timeoutMS>で指定される時間（単位：ms）受信するデータが無かった場合、エラーで戻ります。
変数RESに受信したデータが返ります。（0未満でエラー）

---

# getl,putl

---

書式

getl <FileNo>,<Text_reg>
putl <FileNo>,<Text_reg>

説明

getlはopen命令で開いたファイルから1行取得して、<Text_reg>で指定されるテキストレジス

タに格納します。
putlはopen命令で開いたファイルに、<Text_reg>で指定されるテキストレジスタの内容を格納します。
<FileNo>には、open命令を使用したときの返り値(変数RES)の値を指定します。

---

# gosub

---

書式

gosub <label>,<arg1>,<arg2>,<arg3>...

説明

prgまたはsubのラベルに分岐します。
分岐先でreturn命令を実行すると呼び出したgosub命令の次の行に戻ります。
スタックが一杯になるとs12.0がON し、エラーを知らせます。
arg1(1～)が指定されている場合、分岐先でY16～(実数はZ16～)で使用する事が可能になります。

例

```
prg 0
gosub test, 1 , 2 , 3.0 , 4 , 5.2
stop

sub test
(この時点で、下記変数が自動的に設定されています)
|
Y16 = 1
Y17 = 2
Z18 = 3.0 (実数レジスタに入るので注意)< /FONT >
Y19 = 4
Z20 = 5.2 (実数レジスタに入るので注意)< /FONT >
|
return
```

---

# goto

---

書式

goto <label>

説明

現在のプログラム中へのラベルへ分岐します。
ラベルを探す範囲はprg又はsubで始まる1つのプログラム内で、他のプログラム中に同一のラベルがあっても参照されません。

---

# if,elseif,else,endif

---

書式

if <expr>
elseif <expr>
else
endif

説明

ifから始まりendifで終了する条件命令です。
間にelseif,elseをはさむ事によって条件から1つを選択する事が出来ます。

例

if <expr1>
・・・・・ <expr1>が真なら実行します
elseif <expr2>
・・・・・ <expr2>が真なら実行します
elseif <expr3>
・・・・・ <expr3>が真なら実行します
else
・・・・・ <expr1><expr2><expr3>いずれも真で無いなら実行します
endif

---

# float(),int()

---

書式

float(<expr>)
int(<expr>)

説明

floatは<expr>を実数に型変換します。
intは<expr>は整数に型変換します。

---

# movc

---

書式

movc <expr:ax>

説明

movcは絶対値、movciは相対値円弧補完運転の起動を行います。
円弧補完運転が終了するとプログラムは次のステップへ進みます。
命令の先駆けて位置(A□.17)、速度(A□.18)、駆動軸指定(A□.22)、円弧通過点の位置(A□.26)、円弧補完を行う軸(A□.27)の指定を行います。
実行開始する為にmovc,movci命令はA□.16.6をONにしますが,既に命令実行中の場合、実行終了を待ってから行います。
A□.16.6がONのままだと命令は無視されますので注意が必要です。
実行終了後、A□.16の全ビットはOFFされます。
詳細はSC2000取扱説明書の【4.1.5円弧運転】を参照して下さい。

---

# movc

書式

movc <expr:ax>

説明

movcは絶対値、movciは相対値円弧補完運転の起動を行います。
円弧補完運転が終了するとプログラムは次のステップへ進みます。
命令の先駆けて位置(A□.17)、速度(A□.18)、駆動軸指定(A□.22)、円弧通過点の位置(A□.26)、円弧補完を行う軸(A□.27)の指定を行います。
実行開始する為にmovc,movci命令はA□.16.6をONにしますが,既に命令実行中の場合、実行終了を待ってから行います。
A□.16.6がONのままだと命令は無視されますので注意が必要です。
実行終了後、A□.16の全ビットはOFFされます。
詳細はSC2000取扱説明書の【4.1.5円弧運転】を参照して下さい。

# movh

書式

movh <expr:ax>

説明

パラメータに指定された原点復帰方法で<expr:ax>で指定する1軸を原点復帰し、動作終了後次のステップへ進みます。
命令に先駆けて速度(A□.18)の指定を行います。
実行開始する為にmovh命令はA□.16.7をONにしますが,既に命令実行中の場合、実行終了を待ってから行います。
A□.16.7がONのままだと命令は無視されますので注意が必要です。
実行終了後、A□.16の全ビットはOFFされます。
詳細はSC2000取扱説明書の【4.1.2】原点復帰を参照して下さい。

# movm,movmi

書式

movm <expr:ax>
movmi <expr:ax>

説明

movmは絶対値、movmiは相対値多軸運転の起動を行います。
多軸運転が終了するとプログラムは次のステップへ進みます。
命令の先駆けて位置(A□.17)、速度(A□.18)の指定を行います。
実行開始する為にmovm,movmi命令はA□.16.3をONにしますが,既に命令実行中の場合、実行終了を待ってから行います。
A□.16.3がONのままだと命令は無視されますので注意が必要です。
実行終了後、A□.16の全ビットはOFFされます。
詳細はSC2000取扱説明書の【4.1.4多軸運転】を参照して下さい。

# movm,movmi

書式

movm <expr:ax>
movmi <expr:ax>

説明

movmは絶対値、movmiは相対値多軸運転の起動を行います。
多軸運転が終了するとプログラムは次のステップへ進みます。
命令の先駆けて位置(A□.17)、速度(A□.18)の指定を行います。
実行開始する為にmovm,movmi命令はA□.16.3をONにしますが,既に命令実行中の場合、実行終了を待ってから行います。
A□.16.3がONのままだと命令は無視されますので注意が必要です。
実行終了後、A□.16の全ビットはOFFされます。
詳細はSC2000取扱説明書の【4.1.4多軸運転】を参照して下さい。

# movo

書式

movo <expr:ax>

説明

オーダー運転の起動を行います。
オーダー運転が終了するとプログラムは次のステップへ進みます。
命令の先駆けてステップ番号(A□.19)の指定を行います。
実行開始する為にA□.16.4をONにしますが,既に命令実行中の場合、実行終了を待ってから行います。
A□.16.4がONのままだと命令は無視されますので注意が必要です。
実行終了後、A□.16の全ビットはOFFされます。
詳細はSC2000取扱説明書の【4.1.4多軸運転】を参照して下さい。

# movs,movsi

書式

movs <expr:ax>
movsi <expr:ax>

説明

movsは絶対値、movsiは相対値で<expr:ax>で指定される1軸の単軸運転を行います。
単軸運転が終了するとプログラムは次のステップへ進みます。
命令の先駆けて位置(A□.17)、速度(A□.18)の指定を行います。
実行開始する為にmovs,movsi命令はA□.16.2をONにしますが,既に命令実行中の場合、実行終了を待ってから行います。
A□.16.2がONのままだと命令は無視されますので注意が必要です。

実行終了後、A□.16の全ビットはOFFされます。
詳細はSC2000取扱説明書の【4.1.3単軸運転】を参照して下さい。

---

# movs,movsi

---

書式

movs <expr:ax>
movsi <expr:ax>

説明

movsは絶対値、movsiは相対値で<expr:ax>で指定される1軸の単軸運転を行います。
単軸運転が終了するとプログラムは次のステップへ進みます。
命令の先駆けて位置(A□.17)、速度(A□.18)の指定を行います。
実行開始する為にmovs,movsi命令はA□.16.2をONにしますが,既に命令実行中の場合、実行終了を待ってから行います。
A□.16.2がONのままだと命令は無視されますので注意が必要です。
実行終了後、A□.16の全ビットはOFFされます。
詳細はSC2000取扱説明書の【4.1.3単軸運転】を参照して下さい。

---

# movt,movti

---

書式

movc <expr:ax>

説明

movtは絶対値、movtiは相対値単位時間運転の起動を行います。
単位時間運転が終了するとプログラムは次のステップへ進みます。
命令の先駆けて位置(A□.17)、時間指定(A□.20)の指定を行います。
実行開始する為にA□.16.5をONにしますが,既に命令実行中の場合、実行終了を待ってから行います。
A□.16.5がONのままだと命令は無視されますので注意が必要です。
実行終了後、A□.16の全ビットはOFFされます。
詳細はSC2000取扱説明書【4.1.7単位時間運転】を参照して下さい。

---

# movt,movti

---

書式

movc <expr:ax>

説明

movtは絶対値、movtiは相対値単位時間運転の起動を行います。
単位時間運転が終了するとプログラムは次のステップへ進みます。
命令の先駆けて位置(A□.17)、時間指定(A□.20)の指定を行います。
実行開始する為にA□.16.5をONにしますが,既に命令実行中の場合、実行終了を待ってから行います。

A□.16.5がONのままだと命令は無視されますので注意が必要です。
実行終了後、A□.16の全ビットはOFFされます。
詳細はSC2000取扱説明書【4.1.7単位時間運転】を参照して下さい。

---

# open

---

書式

open "ファイルネーム",<text_mode>
open <text_arg>,<text_mode>

説明

MMC(メモリーカード)に対してアクセスする為に開きます
最初の書式では、アクセスするファイル名を直接指定出来ます。
2番目の書式では、アクセスするファイル名をTレジスタで指定する事が出来ます。

<text_mode>には、"R","W","D"の何れかを指定 する必要が有ります。
"R"を指定した場合、読み込み用モードとしてファイルを開きます。
"W"を指定した場合、書き込み用モードとしてファイルを開きます。
(既存の同名ファイルがある場合、開く前に削除されます)
"D"を指定した場合、MMCのディレクトリー覧を取得するモードとして開きます。
この時、ファイルネームには、一覧を取得する「絶対パス」を指定する事が可能 です。
(空文字列を指定した場合は、ルートディレクトリが対象になります)

変数RESに開いたファイル番号が返ります。
エラーの場合は、変数RESに0未満の値が返ります。

例

ファイル名("test.txt")を読み込みモードで開いて、T0～レジスタに格納する場合

```
sub readtest
Y0 = 0                  ;ループ用
open "test.txt","R"     ;test.txtを読み込みモードで開きます
Y1 = RES                ;Y1にファイル番号を格納します
loop:                   ;ループの開始位置
getl Y1,T(Y0)           ;ファイルから1行読み込みます
if(RES < 0)             ;読めなかったかどうか確認します
goto loopend            ;読めなかったらループから抜けます(終了)
endif                   ;
Y0 = Y0 + 1             ;次の格納場所に変数を設定します
goto loop               ;ループの開始位置に戻る
loopend:                ;ループの終了位置
close Y1                ;開いたファイルを閉じます
return
```

ファイル名("test2.txt")を書き込みモードで開いて、T0～レジスタからの値を保存する場合

```
sub writetest
Y0 = 0
open "test2.txt","W"     ;test2.txtを書き込みモードで開きます
Y1 = RES                 ;Y1にファイル番号を格納します
loop:
putl Y1,T(Y0)            ;ファイルへ1行書き込みます
if(T(Y0).0 = 0)          ;書き込んだデータが空かどうか確認します
goto loopend             ;空なら終了と判断します
endif
Y0 = Y0 + 1              ;次の格納場所に変数を設定します
goto loop
loopend:
close Y1                 ;開いたファイルを閉じます
return
```

ルートディレクトリの一覧をT0～レジスタに保存します

```
sub dirtest
Y0 = 0
open "","D"              ;ルートディレクトリを一覧モードで開きます
Y1 = RES                 ;Y1にファイル番号を格納します
loop:
getl Y1,T(Y0)            ;一覧から1行読み込みます
if(RES < 0)              ;読めなかったかどうか確認します
goto loopend             ;読めなかったら終了と判断します
endif
Y0 = Y0 + 1              ;次の格納場所に変数を設定します
goto loop
loopend:
close Y1                 ;開いたファイルを閉じます
return
```

---

# padj

---

書式

padj <expr:ax>

説明

padjは<expr>で指定される1軸の現在値調整を行います。
実行開始する為にpadj命令はA□.16.14をONにしますが,既に命令実行中の場合、実行終了を待ってから行います。
A□.16.14がONのままだと命令は無視されますので注意が必要です。
実行終了後、A□.16の全ビットはOFFされます。

---

# PI

書式
　PI
説明
　πの値を実数で返します。

# PI2

書式
　PI
説明
　πの2乗を実数で返します。

# pow()

書式
　pow(<expr:x>,<expr:y>)
説明
　<expr:x>の<expr:y>乗を計算します。結果は実数を返します。

# prg

書式
　prg <no>,<label>
説明
　モーションプログラムの先頭に記述し、プログラム番号<no>とプログラム名<label>から構成されます。
　<no>は0から255の範囲で指定します。
　<label>はgosubから参照する場合に必要ですが、省略も可能です。
　実行中にprg命令を見つけた場合、スタックが空でないなら呼び出したgosub命令の次に戻り、スタックが空ならプログラムは終了します。

# prgno

書式

prgno

説明

現在のプログラム番号(sub等の場合は分岐元プログラム番号)を戻します。

---

# print

---

書式

print <Text_reg>,<Format>,<arg1>,....

説明

<arg>で指定されたレジスタ(又は数値)を指定の書式に変換した物を<Text_reg>で指定されるテキストレジスタに格納します。
<Format>と<arg>は、型および個数が一致しなくてはいけません。
もしもこれらが一致しないと、おかしな結果が得られることがあります。

---

# prio

---

書式

prio <arg>

説明

<arg>で指定される行数(実際はこの値に+10された物)を元にモーションの一括動作行数 を指定します。
行数を上げると動作速度は上がりますが、他への影響が出る場合もあるので使用は慎重に行って下さい。
(通常、使用する必要は有りません)

---

# pset

---

書式

pset <expr:ax>

説明

psetは<expr>で指定される1軸の現在値設定を行います。
命令の先駆けて位置(A□.17)の指定を行います。
実行開始する為にpset命令はA□.16.12をONにしますが,既に命令実行中の場合、実行終了を待ってから行います。
A□.16.12がONのままだと命令は無視されますので注意が必要です。
実行終了後、A□.16の全ビットはOFFされます。

---

# putcom

書式

putcom <port>,<arg>

説明

<port>で指定された通信ポートへ<arg>で指定される1バイトのデータを送信します。
エラーが発生した場合、変数RESの値が0以外になります。

# getl,putl

書式

getl <FileNo>,<Text_reg>
putl <FileNo>,<Text_reg>

説明

getlはopen命令で開いたファイルから1行取得して、<Text_reg>で指定されるテキストレジスタに格納します。
putlはopen命令で開いたファイルに、<Text_reg>で指定されるテキストレジスタの内容を格納します。
<FileNo>には、open命令を使用したときの返り値(変数RES)の値を指定します。

# reset

書式

reset

説明

エラー状態から復帰する為に必要な処理を行い、エラーを解除します。

# return

書式

return

説明

呼び出し元のgosub命令の次に戻ります。
return命令実行時にスタックが空の場合プログラムは終了します。
この時エラーにはなりません。

# float(),int()

書式

float(<expr>)
int(<expr>)

説明

floatは<expr>を実数に型変換します。
intは<expr>は整数に型変換します。

---

# scan

---

書式

scan <Text_reg>,<Format>,<arg1>,....

説明

<Text_reg>で指定されたテキストレジスタを指定の書式で変換した物を<arg>で指定されるレジスタに格納します。
<Format>と<arg>は、型および個数が一致しなくてはいけません。
もしもこれらが一致しないと、おかしな結果が得られることがあります。

---

# seplh

---

書式

seplh <val32>,<low16>,<high16>

説明

<val32>の値（32bitレジスタを指定）を16Bits x 2に分割して、<low>と<high>（両方とも16Bitレジスタを指定）に格納します。

例

F10 = $12345678
seplh(F10,E100,E101)

E100に$5678が入ります。
E101に$1234が入ります。

---

# sin(),cos(),tan()

---

書式

sin(<expr>)
cos(<expr>)
tan(<expr>)

説明

<expr>のsin,cos,tanの計算を行います。角度の表現はradです。
結果は実数を返します。

---

# sqrt()

書式

sqrt(<expr>)

説明

<expr>の平方根を計算します。結果は実数を返します。

# stklh()

書式

stklh(<low16>,<high16>)

説明

<low>と<high>（両方とも16Bitレジスタを指定）を下位、上位として結合した値を32Bitで返します。

例

```
E100 = $5678
E101 = $1234
F0 = stklh(E100,E101)

F0に$12345678が入ります。
```

# stop

書式

stop

説明

プログラムを停止します

# sub

書式

sub <label>

説明

サブルーチンの先頭に記述します。
<label>はgosub命令から参照されます。
サブルーチン中のreturn命令を実行すると、呼ばれたgosub命令の直後に復帰します。
実行中にsub命令を見つけた場合、スタックが空で無いなら呼び出したgosub命令の次に戻り、スタックが空ならプログラムは終了します。

# sin(),cos(),tan()

書式

sin(<expr>)
cos(<expr>)
tan(<expr>)

説明

<expr>のsin,cos,tanの計算を行います。角度の表現はradです。
結果は実数を返します。

# wait

書式

wait <expr>

説明

<expr>の値(1ms単位)プログラムを停止します。
実際の実行時間は4msの処理周期で判定される為、4msの整数倍となります。
またモーションの実行中のプログラム数が多い場合、実行時間が影響を受ける場合もあります。

# リファレンス選択

ラダーで使用できる関数・コマンド一覧
アルファベット順一覧
機能別一覧

# リファレンス一覧

ラダーで使用できる関数・コマンド一覧（アルファベット順）

| アルファベット順 | | | |
|---|---|---|---|
| ADD | ADDI | ADDIr | ADDr |
| AND | ANDN | ANDS | CPL |
| DIV | DIVI | DIVIr | DIVr |
| DIX | EQ | EQI | FALL |
| GE | GEI | GT | GTI |
| IXR | LD | LDA | LDBCD |
| | | | |

| | | | |
|---|---|---|---|
| LDBCDr | LDN | LDS | LDSA |
| LE | LEI | LT | LTI |
| MC | MCR | MERGE | MERGEr |
| MOV | MOVB | MOVBL | MOVBLr |
| MOVBr | MOVBW | MOVBWr | MOVI |
| MOVIr | MOVr | MUL | MULI |
| MULIr | MULr | NE | NEG |
| NEGr | NEI | NOP | OR |
| ORN | ORS | OUT | OUTBCD |
| OUTBCDr | RAISE | RES | RESr |
| SET | SETr | SFT | SFTr |
| SIX | SUB | SUBI | SUBIr |
| SUBr | TMR | TMRI | TMRIr |
| TMRr | TMRRST | TMRRSTr | XOR |
| XORN | XTRACT | XTRACTr | |

# リファレンス一覧（機能別一覧）

ラダーで使用できる関数・コマンド一覧（機能別）

| 基本命令 | | | |
|---|---|---|---|
| LD | LDN | LDA | OUT |
| AND | ANDN | ANDS | OR |
| ORN | ORS | LDSA | LDS |
| XOR | XORN | CPL | RAISE |
| FALL | MC | MCR | TMR |
| TMRr | TMRI | TMRIr | TMRRST |
| TMRRSTr | EQ | NE | GE |
| GT | LE | LT | EQI |
| NEI | GEI | GTI | LEI |
| LTI | | | |

| 応用命令 | | | |
|---|---|---|---|
| SET | SETr | RES | RESr |
| MOV | MOVr | MOVI | MOVIr |
| ADD | ADDr | ADDI | ADDIr |
| SUB | SUBr | SUBI | SUBIr |
| MUL | MULr | MULI | MULIr |
| DIV | DIVr | DIVI | DIVIr |
| | | | |

| | | | |
|---|---|---|---|
| NEG | NEGr | SFT | SFTr |
| LDBCD | LDBCDr | OUTBCD | OUTBCDr |
| XTRACT | XTRACTr | MERGE | MERGEr |
| MOVB | MOVBr | MOVBW | MOVBWr |
| MOVBL | MOVBLr | SIX | DIX |
| IXR | NOP | | |

# ADD,ADDr,ADDI,ADDIr

書式

ADD <reg:src>,<reg:dist>
ADDr <reg:src>,<reg:dist>
ADDI <imm>,<reg:dist>
ADDIr <imm>,<reg:dist>

実行条件

ADD,ADDIはAccがONの時実行。
ADDr,ADDIrはAccがOFFからONになる時実行。

説明

ADD,ADDrは<reg:dist>と<reg:src>を加算し、<reg:dist>に格納します。<reg:src>の値は変化しません。
ADDI,ADDIrは<reg:dist>と<imm>を加算し、<reg:dist>に格納します。

# ADD,ADDr,ADDI,ADDIr

書式

ADD <reg:src>,<reg:dist>
ADDr <reg:src>,<reg:dist>
ADDI <imm>,<reg:dist>
ADDIr <imm>,<reg:dist>

実行条件

ADD,ADDIはAccがONの時実行。
ADDr,ADDIrはAccがOFFからONになる時実行。

説明

ADD,ADDrは<reg:dist>と<reg:src>を加算し、<reg:dist>に格納します。<reg:src>の値は変化しません。
ADDI,ADDIrは<reg:dist>と<imm>を加算し、<reg:dist>に格納します。

# ADD,ADDr,ADDI,ADDIr

書式

ADD <reg:src>,<reg:dist>
ADDr <reg:src>,<reg:dist>
ADDI <imm>,<reg:dist>
ADDIr <imm>,<reg:dist>

実行条件

ADD,ADDIはAccがONの時実行。
ADDr,ADDIrはAccがOFFからONになる時実行。

説明

ADD,ADDrは<reg:dist>と<reg:src>を加算し、<reg:dist>に格納します。<reg:src>の値は変化しません。
ADDI,ADDIrは<reg:dist>と<imm>を加算し、<reg:dist>に格納します。

---

# ADD,ADDr,ADDI,ADDIr

---

書式

ADD <reg:src>,<reg:dist>
ADDr <reg:src>,<reg:dist>
ADDI <imm>,<reg:dist>
ADDIr <imm>,<reg:dist>

実行条件

ADD,ADDIはAccがONの時実行。
ADDr,ADDIrはAccがOFFからONになる時実行。

説明

ADD,ADDrは<reg:dist>と<reg:src>を加算し、<reg:dist>に格納します。<reg:src>の値は変化しません。
ADDI,ADDIrは<reg:dist>と<imm>を加算し、<reg:dist>に格納します。

---

# LDA

---

書式

AND <bit>
ANDN <bit>

実行条件

常時

説明

ANDはAccと<bit>の積をとり、Accに格納します。
ANDNはAccと<bit>の反転との積をとり、Accに格納します。

---

# LDA

---

書式

AND <bit>
ANDN <bit>

実行条件

常時

説明

ANDはAccと<bit>の積をとり、Accに格納します。
ANDNはAccと<bit>の反転との積をとり、Accに格納します。

---

# ANDS

---

書式

ANDS

実行条件

常時

説明

Accの状態とスタックの状態の論理積を取りAccに格納します。
実行後スタックは1段ポップされます。

例

LD R5.2 (回路1)
OR R6.8
LD S16.2 (回路2)
OR R2.3
ANDS (結合)

---

# CPL

---

書式

CPL

実行条件

常時

説明

Accの状態を反転させます。OFFならONに、ONならOFFになります。

---

# DIV,DIVr,DIVI,DIVIr

---

書式

DIV <reg:src>,<reg:dist>
DIVr <reg:src>,<reg:dist>
DIVI <imm>,<reg:dist>

DIVIr <imm>,<reg:dist>

実行条件

DIV,DIVIはAccがONの時実行。
DIVr,DIVIrはAccがOFFからONになる時実行。

説明

DIV,DIVrは<reg:dist>を<reg:src>で除算し、<reg:dist>に格納します。<reg:src>の値は変化しません。
DIVI,DIVIrは<reg:dist>を<imm>で除算し、<reg:dist>に格納します。
除算の余りはS6レジスタに格納されます。

---

# DIV,DIVr,DIVI,DIVIr

---

書式

DIV <reg:src>,<reg:dist>
DIVr <reg:src>,<reg:dist>
DIVI <imm>,<reg:dist>
DIVIr <imm>,<reg:dist>

実行条件

DIV,DIVIはAccがONの時実行。
DIVr,DIVIrはAccがOFFからONになる時実行。

説明

DIV,DIVrは<reg:dist>を<reg:src>で除算し、<reg:dist>に格納します。<reg:src>の値は変化しません。
DIVI,DIVIrは<reg:dist>を<imm>で除算し、<reg:dist>に格納します。
除算の余りはS6レジスタに格納されます。

---

# DIV,DIVr,DIVI,DIVIr

---

書式

DIV <reg:src>,<reg:dist>
DIVr <reg:src>,<reg:dist>
DIVI <imm>,<reg:dist>
DIVIr <imm>,<reg:dist>

実行条件

DIV,DIVIはAccがONの時実行。
DIVr,DIVIrはAccがOFFからONになる時実行。

説明

DIV,DIVrは<reg:dist>を<reg:src>で除算し、<reg:dist>に格納します。<reg:src>の値は変化しません。
DIVI,DIVIrは<reg:dist>を<imm>で除算し、<reg:dist>に格納します。
除算の余りはS6レジスタに格納されます。

---

# DIV,DIVr,DIVI,DIVIr

書式

DIV <reg:src>,<reg:dist>
DIVr <reg:src>,<reg:dist>
DIVI <imm>,<reg:dist>
DIVIr <imm>,<reg:dist>

実行条件

DIV,DIVIはAccがONの時実行。
DIVr,DIVIrはAccがOFFからONになる時実行。

説明

DIV,DIVrは<reg:dist>を<reg:src>で除算し、<reg:dist>に格納します。<reg:src>の値は変化しません。
DIVI,DIVIrは<reg:dist>を<imm>で除算し、<reg:dist>に格納します。
除算の余りはS6レジスタに格納されます。

---

# SIX,DIX

書式

SIX <reg>
DIX <reg>

実行条件

常時

説明

SIXは<reg>の値をソースインデックスに設定します。
DIXは<reg>の値をディスティネーションインデックスに設定します。

---

# EQ,NE,GE,GT,LE,LT,EQI,NEI,GEI,GTI,LEI,LTI

書式

EQ <reg1>,<reg2>
NE <reg1>,<reg2>
GE <reg1>,<reg2>
GT <reg1>,<reg2>
LE <reg1>,<reg2>
LT <reg1>,<reg2>
EQI <imm>,<reg2>
NEI <imm>,<reg2>
GEI <imm>,<reg2>
GTI <imm>,<reg2>
LEI <imm>,<reg2>
LTI <imm>,<reg2>

実行条件

常時

説明

比較を行い結果をAccに格納します。
実行時現在のAccの状態をスタックに格納してから行います。
比較前のAccと演算する場合はANDS,ORS命令を使用します。
比較の結果AccがONする条件は以下の通りです。

| 命令 | 結果が真になる条件 |
|---|---|
| EQ<br>EQI | <reg1> ＝ <reg2><br><imm> ＝ <reg2> |
| NE<br>NEI | <reg1> ＜＞ <reg2><br><imm> ＜＞ <reg2> |
| GE<br>GEI | <reg1> ＜＝ <reg2><br><imm> ＜＝ <reg2> |
| GT<br>GTI | <reg1> ＜ <reg2><br><imm> ＜ <reg2> |
| LE<br>LEI | <reg1> ＞＝ <reg2><br><imm> ＞＝ <reg2> |
| LT<br>LTI | <reg1> ＞ <reg2><br><imm> ＞ <reg2> |

例

LD R5.2
EQ R10,F25

（動作説明）ANDS R5.2がONかつ、R10とF25が等しい時AccがON

---

# EQ,NE,GE,GT,LE,LT,EQI,NEI,GEI,GTI,LEI,LTI

---

書式

EQ <reg1>,<reg2>
NE <reg1>,<reg2>
GE <reg1>,<reg2>
GT <reg1>,<reg2>
LE <reg1>,<reg2>
LT <reg1>,<reg2>
EQI <imm>,<reg2>
NEI <imm>,<reg2>
GEI <imm>,<reg2>
GTI <imm>,<reg2>
LEI <imm>,<reg2>
LTI <imm>,<reg2>

実行条件

常時

説明

比較を行い結果をAccに格納します。
実行時現在のAccの状態をスタックに格納してから行います。
比較前のAccと演算する場合はANDS,ORS命令を使用します。
比較の結果AccがONする条件は以下の通りです。

| 命令 | 結果が真になる条件 |
|---|---|
| EQ<br>EQI | <reg1> ＝ <reg2><br><imm> ＝ <reg2> |
| NE<br>NEI | <reg1> ＜＞ <reg2><br><imm> ＜＞ <reg2> |
| GE<br>GEI | <reg1> ＜＝ <reg2><br><imm> ＜＝ <reg2> |
| GT<br>GTI | <reg1> ＜ <reg2><br><imm> ＜ <reg2> |
| LE<br>LEI | <reg1> ＞＝ <reg2><br><imm> ＞＝ <reg2> |
| LT<br>LTI | <reg1> ＞ <reg2><br><imm> ＞ <reg2> |

例

LD R5.2
EQ R10,F25

（動作説明）ANDS R5.2がONかつ、R10とF25が等しい時AccがON

---

# FALL

---

書式

FALL

実行条件

常時

説明

AccのONからOFFになった1周期だけAccをONにします。
立ち下がりエッジを検出します。

---

# EQ,NE,GE,GT,LE,LT,EQI,NEI,GEI,GTI,LEI,LTI

---

書式

EQ <reg1>,<reg2>
NE <reg1>,<reg2>
GE <reg1>,<reg2>
GT <reg1>,<reg2>
LE <reg1>,<reg2>
LT <reg1>,<reg2>

EQI <imm>,<reg2>
NEI <imm>,<reg2>
GEI <imm>,<reg2>
GTI <imm>,<reg2>
LEI <imm>,<reg2>
LTI <imm>,<reg2>

実行条件

常時

説明

比較を行い結果をAccに格納します。
実行時現在のAccの状態をスタックに格納してから行います。
比較前のAccと演算する場合はANDS,ORS命令を使用します。
比較の結果AccがONする条件は以下の通りです。

| 命令 | 結果が真になる条件 |
|---|---|
| EQ<br>EQI | <reg1> ＝ <reg2><br><imm> ＝ <reg2> |
| NE<br>NEI | <reg1> ＜＞ <reg2><br><imm> ＜＞ <reg2> |
| GE<br>GEI | <reg1> ＜＝ <reg2><br><imm> ＜＝ <reg2> |
| GT<br>GTI | <reg1> ＜ <reg2><br><imm> ＜ <reg2> |
| LE<br>LEI | <reg1> ＞＝ <reg2><br><imm> ＞＝ <reg2> |
| LT<br>LTI | <reg1> ＞ <reg2><br><imm> ＞ <reg2> |

例

LD R5.2
EQ R10,F25

（動作説明）ANDS R5.2がONかつ、R10とF25が等しい時AccがON

---

# EQ,NE,GE,GT,LE,LT,EQI,NEI,GEI,GTI,LEI,LTI

---

書式

EQ <reg1>,<reg2>
NE <reg1>,<reg2>
GE <reg1>,<reg2>
GT <reg1>,<reg2>
LE <reg1>,<reg2>
LT <reg1>,<reg2>
EQI <imm>,<reg2>
NEI <imm>,<reg2>
GEI <imm>,<reg2>
GTI <imm>,<reg2>

LEI <imm>,<reg2>
LTI <imm>,<reg2>

実行条件

常時

説明

比較を行い結果をAccに格納します。
実行時現在のAccの状態をスタックに格納してから行います。
比較前のAccと演算する場合はANDS,ORS命令を使用します。
比較の結果AccがONする条件は以下の通りです。

| 命令 | 結果が真になる条件 |
|---|---|
| EQ<br>EQI | <reg1> ＝ <reg2><br><imm> ＝ <reg2> |
| NE<br>NEI | <reg1> ＜＞ <reg2><br><imm> ＜＞ <reg2> |
| GE<br>GEI | <reg1> ＜＝ <reg2><br><imm> ＜＝ <reg2> |
| GT<br>GTI | <reg1> ＜ <reg2><br><imm> ＜ <reg2> |
| LE<br>LEI | <reg1> ＞＝ <reg2><br><imm> ＞＝ <reg2> |
| LT<br>LTI | <reg1> ＞ <reg2><br><imm> ＞ <reg2> |

例

LD R5.2
EQ R10,F25

（動作説明）ANDS R5.2がONかつ、R10とF25が等しい時AccがON

---

# EQ,NE,GE,GT,LE,LT,EQI,NEI,GEI,GTI,LEI,LTI

---

書式

EQ <reg1>,<reg2>
NE <reg1>,<reg2>
GE <reg1>,<reg2>
GT <reg1>,<reg2>
LE <reg1>,<reg2>
LT <reg1>,<reg2>
EQI <imm>,<reg2>
NEI <imm>,<reg2>
GEI <imm>,<reg2>
GTI <imm>,<reg2>
LEI <imm>,<reg2>
LTI <imm>,<reg2>

実行条件

常時

説明

比較を行い結果をAccに格納します。
実行時現在のAccの状態をスタックに格納してから行います。
比較前のAccと演算する場合はANDS,ORS命令を使用します。
比較の結果AccがONする条件は以下の通りです。

| 命令 | 結果が真になる条件 |
|---|---|
| EQ<br>EQI | <reg1> ＝ <reg2><br><imm> ＝ <reg2> |
| NE<br>NEI | <reg1> ＜＞ <reg2><br><imm> ＜＞ <reg2> |
| GE<br>GEI | <reg1> ＜＝ <reg2><br><imm> ＜＝ <reg2> |
| GT<br>GTI | <reg1> ＜ <reg2><br><imm> ＜ <reg2> |
| LE<br>LEI | <reg1> ＞＝ <reg2><br><imm> ＞＝ <reg2> |
| LT<br>LTI | <reg1> ＞ <reg2><br><imm> ＞ <reg2> |

例

LD R5.2
EQ R10,F25

（動作説明）ANDS R5.2がONかつ、R10とF25が等しい時AccがON

---

# EQ,NE,GE,GT,LE,LT,EQI,NEI,GEI,GTI,LEI,LTI

---

書式

EQ <reg1>,<reg2>
NE <reg1>,<reg2>
GE <reg1>,<reg2>
GT <reg1>,<reg2>
LE <reg1>,<reg2>
LT <reg1>,<reg2>
EQI <imm>,<reg2>
NEI <imm>,<reg2>
GEI <imm>,<reg2>
GTI <imm>,<reg2>
LEI <imm>,<reg2>
LTI <imm>,<reg2>

実行条件

常時

説明

比較を行い結果をAccに格納します。
実行時現在のAccの状態をスタックに格納してから行います。
比較前のAccと演算する場合はANDS,ORS命令を使用します。

比較の結果AccがONする条件は以下の通りです。

| 命令 | 結果が真になる条件 |
|---|---|
| EQ<br>EQI | <reg1> ＝ <reg2><br><imm> ＝ <reg2> |
| NE<br>NEI | <reg1> ＜＞ <reg2><br><imm> ＜＞ <reg2> |
| GE<br>GEI | <reg1> ＜＝ <reg2><br><imm> ＜＝ <reg2> |
| GT<br>GTI | <reg1> ＜ <reg2><br><imm> ＜ <reg2> |
| LE<br>LEI | <reg1> ＞＝ <reg2><br><imm> ＞＝ <reg2> |
| LT<br>LTI | <reg1> ＞ <reg2><br><imm> ＞ <reg2> |

例

LD R5.2
EQ R10,F25

（動作説明）ANDS R5.2がONかつ、R10とF25が等しい時AccがON

---

# IXR

---

書式

IXR

実行条件

常時

説明

ソースインデックス及びディスティネーションインデックスの値を0クリアします。

---

# LD,LDN

---

書式

LD <bit>
LDN <bit>

実行条件

常時

説明

LDは<bit>の状態をAccにロードします。
LDNは<bit>の反転状態をAccにロードします。
直前のAccの状態はスタックにプッシュされ後で使用することが出来ます。

---

# LDA

書式

LDA

実行条件

常時

説明

Accの状態をスタックにプッシュします。
この命令は現在のAccの状態を別の回路で使用するためにスタックに保持する為に使用します。

# LDBCD,LDBCDr

書式

LDBCD <reg:src>,<reg:dist>
LDBCDr <reg:src>,<reg:dist>

実行条件

LDBCDはAccがONの時実行。
LDBCDrはAccがOFFからONになる時実行。

説明

<reg:src>のBCDコードをバイナリーに変換して<reg:dist>に格納します。
<reg:src>の値は変化しません。

# LDBCD,LDBCDr

書式

LDBCD <reg:src>,<reg:dist>
LDBCDr <reg:src>,<reg:dist>

実行条件

LDBCDはAccがONの時実行。
LDBCDrはAccがOFFからONになる時実行。

説明

<reg:src>のBCDコードをバイナリーに変換して<reg:dist>に格納します。
<reg:src>の値は変化しません。

# LD,LDN

書式

LD <bit>
LDN <bit>

実行条件

常時

説明

LDは<bit>の状態をAccにロードします。
LDNは<bit>の反転状態をAccにロードします。
直前のAccの状態はスタックにプッシュされ後で使用することが出来ます。

---

# LDS

---

書式

LDS

実行条件

常時

説明

スタックの先頭をAccにロードします。
実行後スタックは1段ポップされます。

---

# LDSA

---

書式

LDSA

実行条件

常時

説明

スタックの先頭をAccにロードします。
実行後スタックはポップされず、以前のままの状態です。

---

# EQ,NE,GE,GT,LE,LT,EQI,NEI,GEI,GTI,LEI,LTI

---

書式

EQ <reg1>,<reg2>
NE <reg1>,<reg2>
GE <reg1>,<reg2>
GT <reg1>,<reg2>
LE <reg1>,<reg2>
LT <reg1>,<reg2>
EQI <imm>,<reg2>
NEI <imm>,<reg2>

GEI <imm>,<reg2>
GTI <imm>,<reg2>
LEI <imm>,<reg2>
LTI <imm>,<reg2>

実行条件

常時

説明

比較を行い結果をAccに格納します。
実行時現在のAccの状態をスタックに格納してから行います。
比較前のAccと演算する場合はANDS,ORS命令を使用します。
比較の結果AccがONする条件は以下の通りです。

| 命令 | 結果が真になる条件 |
|---|---|
| EQ<br>EQI | <reg1> ＝ <reg2><br><imm> ＝ <reg2> |
| NE<br>NEI | <reg1> ＜＞ <reg2><br><imm> ＜＞ <reg2> |
| GE<br>GEI | <reg1> ＜＝ <reg2><br><imm> ＜＝ <reg2> |
| GT<br>GTI | <reg1> ＜ <reg2><br><imm> ＜ <reg2> |
| LE<br>LEI | <reg1> ＞＝ <reg2><br><imm> ＞＝ <reg2> |
| LT<br>LTI | <reg1> ＞ <reg2><br><imm> ＞ <reg2> |

例

LD R5.2
EQ R10,F25

（動作説明）ANDS R5.2がONかつ、R10とF25が等しい時AccがON

---

# EQ,NE,GE,GT,LE,LT,EQI,NEI,GEI,GTI,LEI,LTI

---

書式

EQ <reg1>,<reg2>
NE <reg1>,<reg2>
GE <reg1>,<reg2>
GT <reg1>,<reg2>
LE <reg1>,<reg2>
LT <reg1>,<reg2>
EQI <imm>,<reg2>
NEI <imm>,<reg2>
GEI <imm>,<reg2>
GTI <imm>,<reg2>
LEI <imm>,<reg2>
LTI <imm>,<reg2>

実行条件

常時

説明

比較を行い結果をAccに格納します。
実行時現在のAccの状態をスタックに格納してから行います。
比較前のAccと演算する場合はANDS,ORS命令を使用します。
比較の結果AccがONする条件は以下の通りです。

| 命令 | 結果が真になる条件 |
|---|---|
| EQ<br>EQI | <reg1> ＝ <reg2><br><imm> ＝ <reg2> |
| NE<br>NEI | <reg1> ＜＞ <reg2><br><imm> ＜＞ <reg2> |
| GE<br>GEI | <reg1> ＜＝ <reg2><br><imm> ＜＝ <reg2> |
| GT<br>GTI | <reg1> ＜ <reg2><br><imm> ＜ <reg2> |
| LE<br>LEI | <reg1> ＞＝ <reg2><br><imm> ＞＝ <reg2> |
| LT<br>LTI | <reg1> ＞ <reg2><br><imm> ＞ <reg2> |

例

LD R5.2
EQ R10,F25

（動作説明）ANDS R5.2がONかつ、R10とF25が等しい時AccがON

---

# EQ,NE,GE,GT,LE,LT,EQI,NEI,GEI,GTI,LEI,LTI

---

書式

EQ <reg1>,<reg2>
NE <reg1>,<reg2>
GE <reg1>,<reg2>
GT <reg1>,<reg2>
LE <reg1>,<reg2>
LT <reg1>,<reg2>
EQI <imm>,<reg2>
NEI <imm>,<reg2>
GEI <imm>,<reg2>
GTI <imm>,<reg2>
LEI <imm>,<reg2>
LTI <imm>,<reg2>

実行条件

常時

説明

比較を行い結果をAccに格納します。
実行時現在のAccの状態をスタックに格納してから行います。
比較前のAccと演算する場合はANDS,ORS命令を使用します。
比較の結果AccがONする条件は以下の通りです。

| 命令 | 結果が真になる条件 |
|---|---|
| EQ<br>EQI | <reg1> ＝ <reg2><br><imm> ＝ <reg2> |
| NE<br>NEI | <reg1> ＜＞ <reg2><br><imm> ＜＞ <reg2> |
| GE<br>GEI | <reg1> ＜＝ <reg2><br><imm> ＜＝ <reg2> |
| GT<br>GTI | <reg1> ＜ <reg2><br><imm> ＜ <reg2> |
| LE<br>LEI | <reg1> ＞＝ <reg2><br><imm> ＞＝ <reg2> |
| LT<br>LTI | <reg1> ＞ <reg2><br><imm> ＞ <reg2> |

例

LD R5.2
EQ R10,F25

（動作説明）ANDS R5.2がONかつ、R10とF25が等しい時AccがON

---

# EQ,NE,GE,GT,LE,LT,EQI,NEI,GEI,GTI,LEI,LTI

---

書式

EQ <reg1>,<reg2>
NE <reg1>,<reg2>
GE <reg1>,<reg2>
GT <reg1>,<reg2>
LE <reg1>,<reg2>
LT <reg1>,<reg2>
EQI <imm>,<reg2>
NEI <imm>,<reg2>
GEI <imm>,<reg2>
GTI <imm>,<reg2>
LEI <imm>,<reg2>
LTI <imm>,<reg2>

実行条件

常時

説明

比較を行い結果をAccに格納します。
実行時現在のAccの状態をスタックに格納してから行います。
比較前のAccと演算する場合はANDS,ORS命令を使用します。
比較の結果AccがONする条件は以下の通りです。

| 命令 | 結果が真になる条件 |
|---|---|
| EQ<br>EQI | <reg1> ＝ <reg2><br><imm> ＝ <reg2> |
| NE<br>NEI | <reg1> ＜＞ <reg2><br><imm> ＜＞ <reg2> |
| GE<br>GEI | <reg1> ＜＝ <reg2><br><imm> ＜＝ <reg2> |
| GT<br>GTI | <reg1> ＜ <reg2><br><imm> ＜ <reg2> |
| LE<br>LEI | <reg1> ＞＝ <reg2><br><imm> ＞＝ <reg2> |
| LT<br>LTI | <reg1> ＞ <reg2><br><imm> ＞ <reg2> |

例

LD R5.2
EQ R10,F25

（動作説明）ANDS R5.2がONかつ、R10とF25が等しい時AccがON

---

# MC,MCR

---

書式

MC
MCR

実行条件

常時

説明

MCとMCR命令に囲まれたプログラムの実行を制御することが出来ます。
制御する命令は LD,LDN,EQ,EQI,NE,NEI,GE,GEI,LE,LEI でこれらの命令実行後のAccをOFFとします。
MCとMCR命令のペアは入れ子で使用することが可能です。
つまりMCとMCR命令間のプログラムにさらにMCとMCR命令を記述する事が出来、複雑な回路を整理することが可能です。
入れ子は31回まで可能です。
MCR命令はAccの状態にかかわらず常に実行を行います。

---

# MC,MCR

---

書式

MC
MCR

実行条件

常時

説明

MCとMCR命令に囲まれたプログラムの実行を制御することが出来ます。
制御する命令は LD,LDN,EQ,EQI,NE,NEI,GE,GEI,LE,LEI でこれらの命令実行後のAccをOFFとします。
MCとMCR命令のペアは入れ子で使用することが可能です。
つまりMCとMCR命令間のプログラムにさらにMCとMCR命令を記述する事が出来、複雑な回路を整理することが可能です。
入れ子は31回まで可能です。
MCR命令はAccの状態にかかわらず常に実行を行います。

---

# MERGE,MERGEr

---

書式

MERGE <reg:src>,<imm:nbit>,<reg:dist>,<imm:bpos>
MERGEr <reg:src>,<imm:nbit>,<reg:dist>,<imm:bpos>

実行条件

MERGEはAccがONの時実行。
MERGErはAccがOFFからONになる時実行。

説明

<reg:src>の0ビット目から<imm:nbit>の示すビット数だけ<reg:dist>の<imm:bpos>の示すビット番号にコピーします。
<reg:src>は変化しません。
<imm:nbit>から<reg:dist>ビットだけが変化します。

例

MERGE F0,4,F1,6

| 転送元 | | 転送先 |
|---|---|---|
| F1.0 – F1.5 | → | F1.0 – F1.5 |
| F0.0 | → | F1.6 |
| F0.1 | → | F1.7 |
| F0.2 | → | F1.8 |
| F0.3 | → | F1.9 |
| F1.10 – F1.31 | → | F1.10 – F1.31 |

---

# MERGE,MERGEr

---

書式

MERGE <reg:src>,<imm:nbit>,<reg:dist>,<imm:bpos>
MERGEr <reg:src>,<imm:nbit>,<reg:dist>,<imm:bpos>

実行条件

MERGEはAccがONの時実行。

MERGErはAccがOFFからONになる時実行。

説明

<reg:src>の0ビット目から<imm:nbit>の示すビット数だけ<reg:dist>の<imm:bpos>の示すビット番号にコピーします。
<reg:src>は変化しません。
<imm:nbit>から<reg:dist>ビットだけが変化します。

例

MERGE F0,4,F1,6

| 転送元 | | 転送先 |
|---|---|---|
| F1.0 – F1.5 | → | F1.0 – F1.5 |
| F0.0 | → | F1.6 |
| F0.1 | → | F1.7 |
| F0.2 | → | F1.8 |
| F0.3 | → | F1.9 |
| F1.10 – F1.31 | → | F1.10 – F1.31 |

---

# MOV,MOVr,MOVI,MOVIr

---

書式

MOV <reg:src>,<reg:dist>
MOVr <reg:src>,<reg:dist>
MOVI <imm>,<reg:dist>
MOVIr <imm>,<reg:dist>

実行条件

MOV,MOVIはAccがONの時実行。
MOVr,MOVIrはAccがOFFからONになる時実行。

説明

MOV,MOVrは<reg:src>を<reg:dist>にコピーします。
MOVI,MOVIrは<imm>を<reg:dist>にコピーします。

---

# MOVB,MOVBr

---

書式

MOVB <reg:src>,<reg:dist>,<imm:n>
MOVBr <reg:src>,<reg:dist>,<imm:n>

実行条件

MOVBはAccがONの時実行。
MOVBrはAccがOFFからONになる時実行。

説明

<reg:src>から<reg:dist>へ<imm:n>の示すワード数だけコピーします。

ビット長の異なるレジスタを指定した場合、下位ビットのみコピーされます。
<reg:src>の値は変化しません。
<imm:n>の範囲は1から63です。
SIX,DIX命令によるインデックス修飾が行えます。

例

MOVB F0,F10,4

| 転送元 | | 転送先 |
|---|---|---|
| F0.0 - F0.31 | → | F10.0 - F10.31 |
| F1.0 - F1.31 | → | F11.0 - F11.31 |
| F2.0 - F2.31 | → | F12.0 - F12.31 |
| F3.0 - F3.31 | → | F13.0 - F13.31 |

# MOVBL,MOVBLr

書式

MOVBL <reg:src32>,<reg:dist16>,<imm:n>
MOVBLr <reg:src32>,<reg:dist16>,<imm:n>

実行条件

MOVBLはAccがONの時実行。
MOVBLrはAccがOFFからONになる時実行。

説明

<reg:src32>から<reg:dist16>へ<imm:n>の示すワード数だけコピーします。
この時32ビットレジスタ1個を16ビットレジスタ2個にコピーします。
コピーするワード数は32ビットレジスタの数で指定します。
<reg:src16>の値は変化しません。
<imm:n>の範囲は1から63です。
SIX,DIX命令によるインデックス修飾が行えます。

例

MOVBL F0,E10,3

| 転送元 | | 転送先 |
|---|---|---|
| F10.0 - F10.15 | → | E0.0 - E0.15 |
| F10.16 - F10.31 | → | E1.0 - E1.15 |
| F11.0 - F11.15 | → | E2.0 - E2.15 |
| F11.16 - F11.31 | → | E3.0 - E3.15 |
| F12.0 - F12.15 | → | E4.0 - E4.15 |
| F12.16 - F12.31 | → | E5.0 - E5.15 |

# MOVB,MOVBr

書式

MOVB <reg:src>,<reg:dist>,<imm:n>
MOVBr <reg:src>,<reg:dist>,<imm:n>

実行条件

MOVBはAccがONの時実行。
MOVBrはAccがOFFからONになる時実行。

説明

<reg:src>から<reg:dist>へ<imm:n>の示すワード数だけコピーします。
ビット長の異なるレジスタを指定した場合、下位ビットのみコピーされます。
<reg:src>の値は変化しません。
<imm:n>の範囲は1から63です。
SIX,DIX命令によるインデックス修飾が行えます。

例

MOVB F0,F10,4

| 転送元 | | 転送先 |
|---|---|---|
| F0.0 – F0.31 | → | F10.0 – F10.31 |
| F1.0 – F1.31 | → | F11.0 – F11.31 |
| F2.0 – F2.31 | → | F12.0 – F12.31 |
| F3.0 – F3.31 | → | F13.0 – F13.31 |

---

# MOVBW,MOVBWr

---

書式

MOVBW <reg:src16>,<reg:dist32>,<imm:n>
MOVBWr <reg:src16>,<reg:dist32>,<imm:n>

実行条件

MOVBWはAccがONの時実行。
MOVBWrはAccがOFFからONになる時実行。

説明

<reg:src16>から<reg:dist32>へ<imm:n>の示すワード数だけコピーします。
この時16ビットレジスタ2個を32ビットレジスタ1個にコピーします。
コピーするワード数は32ビットレジスタの数で指定します。
<reg:src16>の値は変化しません。<imm:n>の範囲は1から63です。
SIX,DIX命令によるインデックス修飾が行えます。

例

MOVBW E0,F10,3

| 転送元 | | 転送先 |
|---|---|---|
| E0.0 – E0.15 | → | F10.0 – F10.15 |
| E1.0 – E1.15 | → | F10.16 – F16.31 |
| E2.0 – E2.0 | → | F11.0 – F11.15 |
| E3.0 – E3.15 | → | F11.16 – F11.31 |
| | | |

| | | |
|---|---|---|
| E4.0 – E4.15 | → | F12.0 – F12.15 |
| E5.0 – E5.15 | → | F12.16 – F12.31 |

# MOVBW,MOVBWr

書式

MOVBW <reg:src16>,<reg:dist32>,<imm:n>
MOVBWr <reg:src16>,<reg:dist32>,<imm:n>

実行条件

MOVBWはAccがONの時実行。
MOVBWrはAccがOFFからONになる時実行。

説明

<reg:src16>から<reg:dist32>へ<imm:n>の示すワード数だけコピーします。
この時16ビットレジスタ2個を32ビットレジスタ1個にコピーします。
コピーするワード数は32ビットレジスタの数で指定します。
<reg:src16>の値は変化しません。<imm:n>の範囲は1から63です。
SIX,DIX命令によるインデックス修飾が行えます。

例

MOVBW E0,F10,3

| 転送元 | | 転送先 |
|---|---|---|
| E0.0 – E0.15 | → | F10.0 – F10.15 |
| E1.0 – E1.15 | → | F10.16 – F16.31 |
| E2.0 – E2.0 | → | F11.0 – F11.15 |
| E3.0 – E3.15 | → | F11.16 – F11.31 |
| E4.0 – E4.15 | → | F12.0 – F12.15 |
| E5.0 – E5.15 | → | F12.16 – F12.31 |

# MOV,MOVr,MOVI,MOVIr

書式

MOV <reg:src>,<reg:dist>
MOVr <reg:src>,<reg:dist>
MOVI <imm>,<reg:dist>
MOVIr <imm>,<reg:dist>

実行条件

MOV,MOVIはAccがONの時実行。
MOVr,MOVIrはAccがOFFからONになる時実行。

説明

MOV,MOVrは<reg:src>を<reg:dist>にコピーします。
MOVI,MOVIrは<imm>を<reg:dist>にコピーします。

# MOV,MOVr,MOVI,MOVIr

書式

MOV <reg:src>,<reg:dist>
MOVr <reg:src>,<reg:dist>
MOVI <imm>,<reg:dist>
MOVIr <imm>,<reg:dist>

実行条件

MOV,MOVIはAccがONの時実行。
MOVr,MOVIrはAccがOFFからONになる時実行。

説明

MOV,MOVrは<reg:src>を<reg:dist>にコピーします。
MOVI,MOVIrは<imm>を<reg:dist>にコピーします。

# MOV,MOVr,MOVI,MOVIr

書式

MOV <reg:src>,<reg:dist>
MOVr <reg:src>,<reg:dist>
MOVI <imm>,<reg:dist>
MOVIr <imm>,<reg:dist>

実行条件

MOV,MOVIはAccがONの時実行。
MOVr,MOVIrはAccがOFFからONになる時実行。

説明

MOV,MOVrは<reg:src>を<reg:dist>にコピーします。
MOVI,MOVIrは<imm>を<reg:dist>にコピーします。

# MUL,MULr,MULI,MULIr

書式

MUL <reg:src>,<reg:dist>
MULr <reg:src>,<reg:dist>
MULI <imm>,<reg:dist>
MULIr <imm>,<reg:dist>

実行条件

MUL,MULIはAccがONの時実行。
MULr,MULIrはAccがOFFからONになる時実行。

説明

MUL,MULrは<reg:dist>と<reg:src>を乗算し、<reg:dist>に格納します。<reg:src>の値は変化

しません。
MULI,MULIrは<reg:dist>と<imm>を乗算し、<reg:dist>に格納します。

---

# EQ,NE,GE,GT,LE,LT,EQI,NEI,GEI,GTI,LEI,LTI

---

書式

EQ <reg1>,<reg2>
NE <reg1>,<reg2>
GE <reg1>,<reg2>
GT <reg1>,<reg2>
LE <reg1>,<reg2>
LT <reg1>,<reg2>
EQI <imm>,<reg2>
NEI <imm>,<reg2>
GEI <imm>,<reg2>
GTI <imm>,<reg2>
LEI <imm>,<reg2>
LTI <imm>,<reg2>

実行条件

常時

説明

比較を行い結果をAccに格納します。
実行時現在のAccの状態をスタックに格納してから行います。
比較前のAccと演算する場合はANDS,ORS命令を使用します。
比較の結果AccがONする条件は以下の通りです。

| 命令 | 結果が真になる条件 |
|---|---|
| EQ<br>EQI | <reg1> ＝ <reg2><br><imm> ＝ <reg2> |
| NE<br>NEI | <reg1> ＜＞ <reg2><br><imm> ＜＞ <reg2> |
| GE<br>GEI | <reg1> ＜＝ <reg2><br><imm> ＜＝ <reg2> |
| GT<br>GTI | <reg1> ＜ <reg2><br><imm> ＜ <reg2> |
| LE<br>LEI | <reg1> ＞＝ <reg2><br><imm> ＞＝ <reg2> |
| LT<br>LTI | <reg1> ＞ <reg2><br><imm> ＞ <reg2> |

例

LD R5.2
EQ R10,F25

（動作説明）ANDS R5.2がONかつ、R10とF25が等しい時AccがON

---

# NEG,NEGr

書式

NEG <reg:src>,<reg:dist>
NEGr <reg:src>,<reg:dist>

実行条件

NEGはAccがONの時実行。
NEGrはAccがOFFからONになる時実行。

説明

<reg:src>を符号反転して<reg:dist>に格納します。
<reg:src>の値は変化しません。

# NEG,NEGr

書式

NEG <reg:src>,<reg:dist>
NEGr <reg:src>,<reg:dist>

実行条件

NEGはAccがONの時実行。
NEGrはAccがOFFからONになる時実行。

説明

<reg:src>を符号反転して<reg:dist>に格納します。
<reg:src>の値は変化しません。

# EQ,NE,GE,GT,LE,LT,EQI,NEI,GEI,GTI,LEI,LTI

書式

EQ <reg1>,<reg2>
NE <reg1>,<reg2>
GE <reg1>,<reg2>
GT <reg1>,<reg2>
LE <reg1>,<reg2>
LT <reg1>,<reg2>
EQI <imm>,<reg2>
NEI <imm>,<reg2>
GEI <imm>,<reg2>
GTI <imm>,<reg2>
LEI <imm>,<reg2>
LTI <imm>,<reg2>

実行条件

常時

説明

比較を行い結果をAccに格納します。
実行時現在のAccの状態をスタックに格納してから行います。
比較前のAccと演算する場合はANDS,ORS命令を使用します。
比較の結果AccがONする条件は以下の通りです。

| 命令 | 結果が真になる条件 |
|---|---|
| EQ<br>EQI | <reg1> ＝ <reg2><br><imm> ＝ <reg2> |
| NE<br>NEI | <reg1> ＜＞ <reg2><br><imm> ＜＞ <reg2> |
| GE<br>GEI | <reg1> ＜＝ <reg2><br><imm> ＜＝ <reg2> |
| GT<br>GTI | <reg1> ＜ <reg2><br><imm> ＜ <reg2> |
| LE<br>LEI | <reg1> ＞＝ <reg2><br><imm> ＞＝ <reg2> |
| LT<br>LTI | <reg1> ＞ <reg2><br><imm> ＞ <reg2> |

例

LD R5.2
EQ R10,F25

（動作説明）ANDS R5.2がONかつ、R10とF25が等しい時AccがON

---

# NOP

---

書式

NOP

実行条件

常時

説明

何もせずに次に進みます。

---

# OR,ORN

---

書式

OR <bit>
ORN <bit>

実行条件

常時

説明

ORはAccと<bit>の論理和を取りAccに格納します。

ORNはAccと<bit>の反転との論理和を取りAccに格納します。

---

# OR,ORN

---

書式

OR <bit>
ORN <bit>

実行条件

常時

説明

ORはAccと<bit>の論理和を取りAccに格納します。
ORNはAccと<bit>の反転との論理和を取りAccに格納します。

---

# ORS

---

書式

ORS

実行条件

常時

説明

Accの状態とスタックの状態の論理和をとりAccに格納します。
実行後スタックは1段ポップされます。

---

# LDA

---

書式

OUT <bit>

実行条件

常時

説明

Accの状態を<bit>に出力します。

---

# OUTBCD,OUTBCDr

---

書式

OUTBCD <reg:src>,<reg:dist>
OUTBCDr <reg:src>,<reg:dist>

実行条件

OUTBCDはAccがONの時実行。
OUTBCDrはAccがOFFからONになる時実行。

説明

<reg:src>のバイナリーをBCDに変換して<reg:dist>に格納します。
<reg:src>の値は変化しません。

---

# OUTBCD,OUTBCDr

---

書式

OUTBCD <reg:src>,<reg:dist>
OUTBCDr <reg:src>,<reg:dist>

実行条件

OUTBCDはAccがONの時実行。
OUTBCDrはAccがOFFからONになる時実行。

説明

<reg:src>のバイナリーをBCDに変換して<reg:dist>に格納します。
<reg:src>の値は変化しません。

---

# RAISE

---

書式

RAISE

実行条件

常時

説明

AccのOFFからONになった1周期だけAccをONにします。
立ち上がりエッジを検出します。

---

# RES,RESr

---

書式

RES <bit>
RESr <bit>

実行条件

RESはAccがONの時実行。
RESrはAccがOFFからONになる時実行。

説明

<bit>をOFFします。
OUT命令とは異なり実行時だけ出力を行いますので、RES命令でOFFさせたビットは他の回

路でセットすることが出来ます。

---

# RES,RESr

---

書式

RES <bit>
RESr <bit>

実行条件

RESはAccがONの時実行。
RESrはAccがOFFからONになる時実行。

説明

<bit>をOFFします。
OUT命令とは異なり実行時だけ出力を行いますので、RES命令でOFFさせたビットは他の回路でセットすることが出来ます。

---

# SET,SETr

---

書式

SET <bit>
SETr <bit>

実行条件

SETはAccがONの時実行。
SETrはAccがOFFからONになる時実行。

説明

<bit>をONします。
OUT命令とは異なり実行時だけ出力を行いますので、SET命令でONさせたビットは他の回路でリセットすることが出来ます。

---

# SET,SETr

---

書式

SET <bit>
SETr <bit>

実行条件

SETはAccがONの時実行。
SETrはAccがOFFからONになる時実行。

説明

<bit>をONします。
OUT命令とは異なり実行時だけ出力を行いますので、SET命令でONさせたビットは他の回路でリセットすることが出来ます。

---

# SFT,SFTr

書式

SFT <reg:src>,<reg:dist>,<imm:nbit>
SFTr <reg:src>,<reg:dist>,<imm:nbit>

実行条件

SFTはAccがONの時実行。
SFTrはAccがOFFからONになる時実行。

説明

<reg:src>を<imm:nbit>ビットシフトして<reg:dist>に格納します。
<imm:nbit>が正の場合左シフト、負の場合右シフトを行います。
<reg:src>の値は変化しません。

# SFT,SFTr

書式

SFT <reg:src>,<reg:dist>,<imm:nbit>
SFTr <reg:src>,<reg:dist>,<imm:nbit>

実行条件

SFTはAccがONの時実行。
SFTrはAccがOFFからONになる時実行。

説明

<reg:src>を<imm:nbit>ビットシフトして<reg:dist>に格納します。
<imm:nbit>が正の場合左シフト、負の場合右シフトを行います。
<reg:src>の値は変化しません。

# SIX,DIX

書式

SIX <reg>
DIX <reg>

実行条件

常時

説明

SIXは<reg>の値をソースインデックスに設定します。
DIXは<reg>の値をディスティネーションインデックスに設定します。

# SUB,SUBr,SUBI,SUBIr

書式

SUB <reg:src>,<reg:dist>
SUBr <reg:src>,<reg:dist>
SUBI <imm>,<reg:dist>
SUBIr <imm>,<reg:dist>

実行条件

SUB,SUBIはAccがONの時実行。
SUBr,SUBIrはAccがOFFからONになる時実行。

説明

SUB,SUBrは<reg:dist>から<reg:src>を減算し、<reg:dist>に格納します。<reg:src>の値は変化しません。
SUBI,SUBIrは<reg:dist>から<imm>を減算し、<reg:dist>に格納します。

---

# SUB,SUBr,SUBI,SUBIr

---

書式

SUB <reg:src>,<reg:dist>
SUBr <reg:src>,<reg:dist>
SUBI <imm>,<reg:dist>
SUBIr <imm>,<reg:dist>

実行条件

SUB,SUBIはAccがONの時実行。
SUBr,SUBIrはAccがOFFからONになる時実行。

説明

SUB,SUBrは<reg:dist>から<reg:src>を減算し、<reg:dist>に格納します。<reg:src>の値は変化しません。
SUBI,SUBIrは<reg:dist>から<imm>を減算し、<reg:dist>に格納します。

---

# SUB,SUBr,SUBI,SUBIr

---

書式

SUB <reg:src>,<reg:dist>
SUBr <reg:src>,<reg:dist>
SUBI <imm>,<reg:dist>
SUBIr <imm>,<reg:dist>

実行条件

SUB,SUBIはAccがONの時実行。
SUBr,SUBIrはAccがOFFからONになる時実行。

説明

SUB,SUBrは<reg:dist>から<reg:src>を減算し、<reg:dist>に格納します。<reg:src>の値は変化しません。
SUBI,SUBIrは<reg:dist>から<imm>を減算し、<reg:dist>に格納します。

---

# SUB,SUBr,SUBI,SUBIr

書式

SUB <reg:src>,<reg:dist>
SUBr <reg:src>,<reg:dist>
SUBI <imm>,<reg:dist>
SUBIr <imm>,<reg:dist>

実行条件

SUB,SUBIはAccがONの時実行。
SUBr,SUBIrはAccがOFFからONになる時実行。

説明

SUB,SUBrは<reg:dist>から<reg:src>を減算し、<reg:dist>に格納します。<reg:src>の値は変化しません。
SUBI,SUBIrは<reg:dist>から<imm>を減算し、<reg:dist>に格納します。

# TMR,TMRr,TMRI,TMRIr

書式

TMR <reg>
TMRr <reg>
TMRI <imm>
TMRIr <imm>

実行条件

TMR,TMRIはAccがONの時実行。
TMRr,TMRIrはAccがOFFからONになる時実行。

説明

TMR,TMRrは<reg>の時間(1ms単位)だけAccをONにします。
TMRI,TMRIrは<imm>の時間(1ms単位)だけAccをONにします。
TMR,TMRIは入力がONの時出力もON、ONからOFFになるとそこから設定時間後に出力をOFFします。
途中で出力をOFFするには、TMRRST命令を使用します、

# TMR,TMRr,TMRI,TMRIr

書式

TMR <reg>
TMRr <reg>
TMRI <imm>
TMRIr <imm>

実行条件

TMR,TMRIはAccがONの時実行。
TMRr,TMRIrはAccがOFFからONになる時実行。

説明

TMR,TMRrは<reg>の時間(1ms単位)だけAccをONにします。
TMRI,TMRIrは<imm>の時間(1ms単位)だけAccをONにします。
TMR,TMRIは入力がONの時出力もON、ONからOFFになるとそこから設定時間後に出力をOFFします。
途中で出力をOFFするには、TMRRST命令を使用します、

---

# TMR,TMRr,TMRI,TMRIr

---

書式

TMR <reg>
TMRr <reg>
TMRI <imm>
TMRIr <imm>

実行条件

TMR,TMRIはAccがONの時実行。
TMRr,TMRIrはAccがOFFからONになる時実行。

説明

TMR,TMRrは<reg>の時間(1ms単位)だけAccをONにします。
TMRI,TMRIrは<imm>の時間(1ms単位)だけAccをONにします。
TMR,TMRIは入力がONの時出力もON、ONからOFFになるとそこから設定時間後に出力をOFFします。
途中で出力をOFFするには、TMRRST命令を使用します、

---

# TMR,TMRr,TMRI,TMRIr

---

書式

TMR <reg>
TMRr <reg>
TMRI <imm>
TMRIr <imm>

実行条件

TMR,TMRIはAccがONの時実行。
TMRr,TMRIrはAccがOFFからONになる時実行。

説明

TMR,TMRrは<reg>の時間(1ms単位)だけAccをONにします。
TMRI,TMRIrは<imm>の時間(1ms単位)だけAccをONにします。
TMR,TMRIは入力がONの時出力もON、ONからOFFになるとそこから設定時間後に出力をOFFします。
途中で出力をOFFするには、TMRRST命令を使用します、

---

# TMRRST,TMRRSTr

書式

TMRRST
TMRRSTr

実行条件

TMRRSTはAccがONの時実行。
TMRRSTrはAccがOFFからONになる時実行。

説明

この命令直後のTMR,TMRr,TMRI,TMRIr命令の出力をOFFします。

直後のタイマー命令1つに対して有効です。

# TMRRST,TMRRSTr

書式

TMRRST
TMRRSTr

実行条件

TMRRSTはAccがONの時実行。
TMRRSTrはAccがOFFからONになる時実行。

説明

この命令直後のTMR,TMRr,TMRI,TMRIr命令の出力をOFFします。
直後のタイマー命令1つに対して有効です。

# XOR,XORN

書式

XOR <bit>
XORN <bit>

実行条件

常時

説明

XORはAccと<bit>との排他的論理和を取りAccに格納します。
XORNはAccと<bit>の反転との排他的論理和を取りAccに格納します。

# XOR,XORN

書式

XOR <bit>
XORN <bit>

実行条件

常時

説明

XORはAccと<bit>との排他的論理和を取りAccに格納します。
XORNはAccと<bit>の反転との排他的論理和を取りAccに格納します。

---

# XTRACT,XTRACTr

---

書式

XTRACT <reg:src>,<imm:bpos>,<reg:dist>,<imm:nbit>
XTRACTr <reg:src>,<imm:bpos>,<reg:dist>,<imm:nbit>

実行条件

XTRACTはAccがONの時実行。
XTRACTrはAccがOFFからONになる時実行。

説明

<reg:src>の<imm:bpos>の示すビット位置から<imm:nbit>の示すビット数だけ<reg:dist>の0ビット目にコピーします。<reg:src>は変化しません。<reg:dist>は下位ビットから<imm:nbit>だけ変化します。

例

XTRACT F0,4,F1,6

| 転送元 | | 転送先 |
|---|---|---|
| F0.4 | → | F1.0 |
| F0.5 | → | F1.1 |
| F0.6 | → | F1.2 |
| F0.7 | → | F1.3 |
| F1.4 – F1.31 | → | F1.4 – F1.31 |

---

# XTRACT,XTRACTr

---

書式

XTRACT <reg:src>,<imm:bpos>,<reg:dist>,<imm:nbit>
XTRACTr <reg:src>,<imm:bpos>,<reg:dist>,<imm:nbit>

実行条件

XTRACTはAccがONの時実行。
XTRACTrはAccがOFFからONになる時実行。

説明

<reg:src>の<imm:bpos>の示すビット位置から<imm:nbit>の示すビット数だけ<reg:dist>の0ビット目にコピーします。
<reg:src>は変化しません。
<reg:dist>は下位ビットから<imm:nbit>だけ変化します。

例

XTRACT F0,4,F1,6

| 転送元 | | 転送先 |
|---|---|---|
| F0.4 | → | F1.0 |
| F0.5 | → | F1.1 |
| F0.6 | → | F1.2 |
| F0.7 | → | F1.3 |
| F1.4 – F1.31 | → | F1.4 – F1.31 |


株式会社　ワコー技研

本社　〒２３０－００４５　横浜市鶴見区末広町１－１－５０
℡：０４５－５０２－４４４１　FAX：０４５－５０２－８６２４

大阪営業所　〒５７７－０８４３　大阪府東大阪市荒川３－２６－１０－１０１
℡：０６－６７２８－１１７２　FAX：０６－６７８２－１１７３

名古屋出張所　〒４８２－００１１　愛知県岩倉市昭和町２－６２－１－３０２
℡：０５８７－３８－４０３３　FAX：０５８７－３８－４０３３

当社HP　http://www.wacogiken.co.jp/