TRiLOGI

Version4.1

目次

第3章　TBASIC ステートメント及び関数オペレータ

第1章　TRiLOGI Ver3.x からの移行

本マニュアルは新機種Mシリーズ専用のTRiLOGI Ver4.0リファレンスマニュアルに相当します。新たに追加された諸機能及び専用コマンドを取りまとめています。H シリーズや E シリーズにつきましては従来のTRiLOGI Ver3.xのリファレンスマニュアルをご参照ください。

TRiLOGI Ver4.0は従来のVer3.xラダーロジック言語の開発環境を拡大して、新たにTBASIC言語を追加・併用することで、複雑な数値演算のプログラミングも、豊富な組み込み関数や柔軟な TBASIC 言語を使用することで、簡単に目的のプログラムを開発することができます。

TRiLOGI Ver4.0のTBASICコマンドはラダーロジック言語をより柔軟に拡張できるように、カスタム関数 CusFn という概念を導入しました。これは最大 128 個までのカスタム関数をTRiLOGI Ver4.0の専用カスタム関数エディターに登録し、このカスタム関数をラダープログラムの実行コイルとして起動すると、専用カスタム関数エディターに TBASIC 言語で記述したプログラムを実行します。またカスタム関数から他のカスタム関数も呼び出してサブルーチンとして実行させることもできます。

次の 2～4 章で TRiLOGI Ver3.x から Ver4.0 へ移行に際し、新たに追加された TBASIC コマンドやカスタム関数CusFnについて取りまとめます。この章は主にTRiLOGI Ver3.xに追加・変更された操作メニューコマンドについてご説明します。

1. File Menu ファイル メニュー

ファイルメニューに新たに追加されたRun UtilitiesコマンドはTRiLOGIを終了することなく、他の実行形式の DOS プログラムを起動することができます。ハイライトバーをポップアップさてた EXE ファイルに合わせて Enter キーを押すと、目的の DOS プログラムを別タスクで起動することができます。

2. Edit Menu エディット メニュー

① エディットメニューには 2 つの Edit Custom Function メニューと Browse All CusFn メニューが新たに追加されました。カスタム関数 CusFn 用ウィンドウに TBASIC でプログラムを記述します。(カスタム関数 CusFn の詳細は第 2 章をご参照ください。)

Browse All CusFn <Ctrl-F7>コマンドは読み出し専用でカスタム関数 CusFn ウィンドウをオープンできます。目的のカスタム関数 CusFn をオープンするには、作成登録したカスタム関数の関数番号 1～128 を入力するか、オープンしているカスタム関数 CusFn ウィンドウで<↑ ↓>キーを押すと、順次ウィンドウを切り替えることができます。またシュミレーション機能を実行中に Browse All CusFn コマンドのショートカットキー<Ctrl-F7>を押すと、直ちにカスタム関数を参照することができます。

② TRiLOGI Ver4.0 は高速タイマ機能をサポートしています。1～128 の高速タイマは TBASIC の HSTIMER コマンドを使用して構成できます。高速タイマとして定義したタイマは 0.01～99.99sec の 0.01sec ステップでセット値 SV がカウントダウンします。また高速タイマとして定義しないタイマは Ver3.x と同様に通常のタイマ機能として動作し 0.1～999.9sec の 0.1sec ステップでカウントダウンします。

③ TRiLOGI Ver4.0 はすべての Input 入力・Output 出力・Relay リレー・Timer タイマ・Counter カウンタの現在値 Present Value は 16 ビットの Integer 整数値として格納されます。入力の場合は I/O 表の左端に I/O 割付番号が表示され、右端の CH bit 番号が格納 16 ビットの Integer 整数値の I/O ワード番号に相当します。

Input #1 入力番号 1 の場合は INPUT[1] = 0
Input #11 入力番号 11 の場合は INPUT[1] = A

3. Controller Menu コントローラ メニュー

コントローラメニューには次のコマンドが新たに追加されました。

① Set PLC's Clock/Calendar 時刻/日付の設定

PLC 内臓のリアルタイムクロックに年月日及び時刻を設定します。コマンドを実行し<Del><Backspase>キーでデータを修正し<Enter>キーでデータ確定します。年月日及び時刻データを設定完了後は PLC の RTC.Err フラグがクリアーされます。

② 1: Host Timer/Ctr SV -> PLC タイマ・カウンタ値を PC から PLC へ転送
2: PLC's Tim/Ctr SV -> Host タイマ・カウンタ値を PLC から PC へ読み出し

TRiLOGI Ver4.0 はタイマ・カウンタの設定値 Set value (SV)をショートカットキーの<F5><F6>を押して I/O 表を呼び出し、この設定値 Set value (SV)を変更しできますが、変更した値を 1: Host Timer/Ctr SV -> PLC コマンドで PC から PLC へ転送できます。また PLC 側の値も 2: PLC's Tim/Ctr SV ->Host コマンドで PLC から PC へ読み出しをすることができます。

③ パスワード保護スキームの変更

TRiLOGI Ver3.x ではパスワードで保護したプログラムを、PLC にプログラムを上書きして再転送する場合、前回設定したパスワードは消去されました。このため再度 PLC にプログラムを上書きして再転送する際に、その都度、パスワードを設定する必要がありました。しかし Ver4.0 は一度パスワードを設定すると、プログラムを上書きして再転送する場合も、パスワードを再設定する必要がなくなりました。設定したパスワードの消去は Target PLC Access コマンドを実行し、新たに追加された Delete Password & Clear Program コマンドを実行すると、設定したパスワードを消去することができます。このコマンドはパスワード及びプログラムも消去されますのでコマンド実行後は、再度プログラムを転送する必要があります。

4. Simulate menu シュミレート メニュー

TRiLOGI Ver4.0 のユーザ作成プログラムメモリーは、従来のプログラムステップからワード(16 ビット/2 バイト)で表現されます。タイマ・カウンタ・特殊機能コイル及び TBASIC コマンドのオペランド数やオペレータ数（16bit/32bit）を含むすべてのラダーロジック要素は正確にメモリー上に 1 ワード単位で格納されます。ラダーロジックやカスタム関数 CusFn で作成したプログラムの全ワード数は Only Compile コマンドを実行してプログラムの全ワード数を確認できます。

もし、カスタム関数 CusFn に定義した TBASIC プログラムがラダーロジックに使用されていない場合、TRiLOGI Ver4.0 はこのカスタム関数 CusFn をコンパイルせず、またユーザ開放メモリーエリアにも割り当てません。

Silent Operation サイレント操作

一部の TBASIC 命令（ PRINT,INPUT,ADC 等）はシュミレーションの実行中、命令が実行されると繰り返しウィンドウ画面に表示されます。この画面表示がわずらわしい場合は、Silent Operation コマンド、Suppress All Messages コマンド、Allow Only ADC input コマンドを選択することで表示させないことができます。このコマンドを選択した内容は初期設定ファイルに保存されませんのでご注意ください。

5. Print menu プリント メニュー

① 新しい追加コマンドの Custom Function コマンドは TBASIC で作成したカスタム関数の関数番号 CusFn #1～CusFn #128 を印刷します。（カスタム関数 CusFn で未定義の関数番号は印刷されません。）

② 最初にプリントメニューのどれかの項目を選択実行すると、デフォルトで印刷結果は DOS File のテキストファイル形式でディスクに出力されます。これは既存のエディターで出力されたファイルを開くことができ、目的に応じて内容を編集することが容易になります。また Windows95（3.1）にもとづいたワードプロセッサーを使用する場合は、すべてのファイル内容を選択するか、表示用のグラフィック文字に適合させるため MS Linedraw を選択しフォントを変更する必要があります。

また DOS File のテキストファイル形式でディスクに出力させる場合、適当なファイル名を付ける必要があります。一度ファイル名を付けてディスクに出力させると、次の印刷で前回と同じファイル名でディスクに出力されます。

③ TRiLOGI Ver3.x に設定されていた CH : Bit logic コマンドは削除されました。

④ Printer Setup コマンドでプリンター固有のスタートコード及びエンドコードを設定登録できます。(必要時選択、また詳細コードはプリンター側に依存します。)

6. Special Bits Menu 特殊ビット メニュー

Special Bits menu 特殊ビットメニューの Clock Pulse 組み込みパルスに 0.05s・0.5s のクロックパルス周期が追加されました。

① RTC.Err Flag リアルタイムクロックエラーフラグ

すべての M シリーズは RTC リアルタイムクロックを搭載しています。リアルタイムクロックは年月日及び時刻を格納し計測します。しかしオプションのバッテリーバックアップ付き RTC ユニットを搭載していない場合、PLC の電源が切れると RTC リアルタイムクロックの格納データもリセットされ、再度 PLC に電源が投入されると RTC リアルタイムクロックは工場出荷設定値のデータから計測開始します。Special Bits Menu 特殊ビットメニューに RTC.Err Flag リアルタイムクロックエラーフラグが追加され、PLC に電源断が認められた場合、この RTC.Err Flag リアルタイムクロックエラーフラグが ON になります。また M シリーズの LED インジケータ搭載モデルはリアルタイムクロックエラーLED が点灯します。

RTC.Err Flag リアルタイムクロックエラーフラグのリセット OFF は Controller Menu コントローラメニューの Set PLC's Clock/Calendar コマンドを実行し年月日及び時刻を設定すると RTC.Err Flag リアルタイムクロックエラーフラグをリセットできます。<Del><Backspase>キーでデータを修正し<Enter>キーでデータ確定します。年月日及び時刻データを設定完了後は PLC の RTC.Err フラグがクリアーされます。

② MX-RTC バッテリーバックアップ付きユニット搭載（オプション）

オプションの MX-RTC を搭載している場合は、自動的に年月日及び時刻を計測しますので、RTC.Err Flag リアルタイムクロックエラーフラグはセットされません。

7. ラダー図のコメント

TRiLOGI Ver3.x の Put Comments コマンドは英数字 1 行 70 キャラクターの 3 行ですが、Ver4.0 は英数字 1 行 70 キャラクターの 4 行まで拡大されました。

8. タイマ カウンタ設定値 Set Value（SV）変更

TRiLOGI Ver3.x で内部タイマ・カウンタの Set Value（SV）をプログラム実行中に変更する方法は、Edit Present Value 現在値変更ウィンドウで Set Value（SV）を設定・変更しましたが、Ver4.0 ではプログラム実行中に 2 つの TBASIC コマンド SetTimerSV と SetCtrSV を使っていつでも内部タイマ・カウンタの Set Value（SV）の設定値をプログラムで変更きできるように拡張されました。

たとえばシュミレーション画面やオンラインモニタ画面で、従来通り内部タイマ・カウンタの Set Value（SV）変更するために Edit Present Value 現在値変更ウィンドウをオープンしたとします。ここで TBASIC コマンドを使ってプログラムで Set Value（SV）を変更する様なステートメントが記述されていれば、直ちに Set Value（SV）は変更され、Edit Present Value 現在値変更ウィンドウ内のデータも変更され反映されます。

内部タイマ・カウンタの Set Value（SV）変更する Edit Present Value 現在値変更ウィンドウや TBASIC コマンドは、PLC の EEPROM の I/O 表プリセットデータを書き換えてることはできません。この様な場合は前節でご説明しました Controller Menu コントローラメニューの Host Timer/Ctr SV -> PLC コマンドや PLC's Tim/Ctr SV ->Host コマンドをご使用ください。

第2章　TBASIC エディタ & シュミレータ

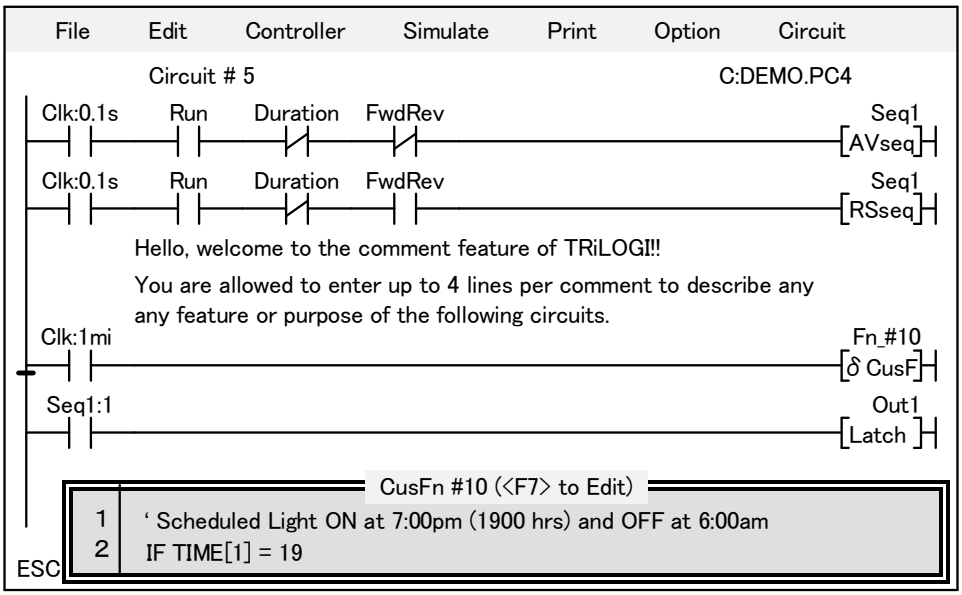
1. カスタム関数 CusFn

TRiLOGI Ver4.0 のカスタム関数 CusFn は関数番号 1～128 個まで作成し登録可能です。カスタム関数 CusFn はカスタム関数エディターに BASIC 言語を応用した TBASIC コマンド及びステートメントを使ってプログラムを記述します。

カスタム関数 CusFn 用のカスタム関数エディター画面を開く方法は、次の 2 通りの方法で簡単にエディターをオープンできます。

① Edit メニューの Custom Function コマンドを選択し実行するか、ショートカット<F7>キーを押すと、Enter Function # (1-128) ウィンドウボックスに、作成・登録する関数番号を入力するとカスタム関数エディターをオープンできます。

② すでにラダーロジック回路上の出力コイルに[CusFn]や[δ CusF]でカスタム関数が構成されている場合は、カソールをその回路番号に合わせると画面下に下図の様にカスタム関数の初めの 2 行が小さく表示されます。



上図の場合、画面下に表示されるカスタム関数の初めの 2 行は CusFn #10 関数番号 10 番を表します。ここで<F7>キーを押すとこの関数番号のカスタム関数エディターをオープンできます。

2. カスタム関数エディター

TRiLOGI Ver4.0 のカスタム関数エディター用ウィンドウボックスは 1 行 70 キャラクターの 60 行まで TBASIC を記述できます。ウィンドウボックスの左端には 1～60 行までの行番が付けられて、一度にウィンドウボックス表示できる行は 20 行までです。ウィンドウボックスに表示できない行は<PgUp><PgDn><↑><↓>キーを使って画面をスクロールすることができます。カスタム関数エディター用のウィンドウボックスでの文字操作は通常のテキストエディターの様に記述・編集ができます。

エディターの主な操作キーは次の通りです。

<Ctrl>+<Enter>	カソール位置のカレント行の前に 1 行挿入。
<Ctrl>+<Backspace>	カソール位置のカレント行を削除。
<←><↑><↓><→>	ウインドボックス内のカソールを移動。
<PgUp><PgDn>	前画面・次画面に 1 画面スクロール。
<Backspace>	カソールの左文字を削除。
<Del>	カソールの右文字を削除。
<Home>	カレント行の先頭位置にカソール移動。
<End>	カレント行の最終位置にカソール移動。
<Enter>	カレント行を改行。
<Ctrl>+<→>	次の先頭文字列に移動。
<Ctrl>+<←>	前の先頭文字列に移動。
<Ctrl>+<C>	選択行のコピー。
<Ctrl>+<P>	選択行の貼り付け。
<Ctrl>+<N>	次番号のカスタム関数表示。
<Ctrl>+<B>	前番号のカスタム関数表示。

通常のテキストエディターと異なり TRiLOGI Ver4.0 はウインドウボックス内でテキスト編集時に単に<Enter>キーを押して行を挿入して改行することはできません。この様な場合は<Ctrl>+<Enter>キーを押し操作してください。また入力されたテキストが 60 行で<Ctrl>+<Enter>でカソール位置のカレント行の前に 1 行挿入したり<Ctrl>+<P>で選択行の貼り付けを行うと、60 行からあふれた内容は消失してしまいますのでご注意ください。

カスタム関数エディターのウィンドウボックス 60 行に収まり切らない様なプログラムの場合は、作成したプログラムを簡略化して 60 行に収めるか、他のカスタム関数番号にサブルーチンを移し、このカスタム関数を CALL n ステートメントで呼び出す様なプログラム構造にして、ウィンドウボックス内には必ず 60 行に収まる様にしてください。

カスタム関数エディターのテキストのコピー

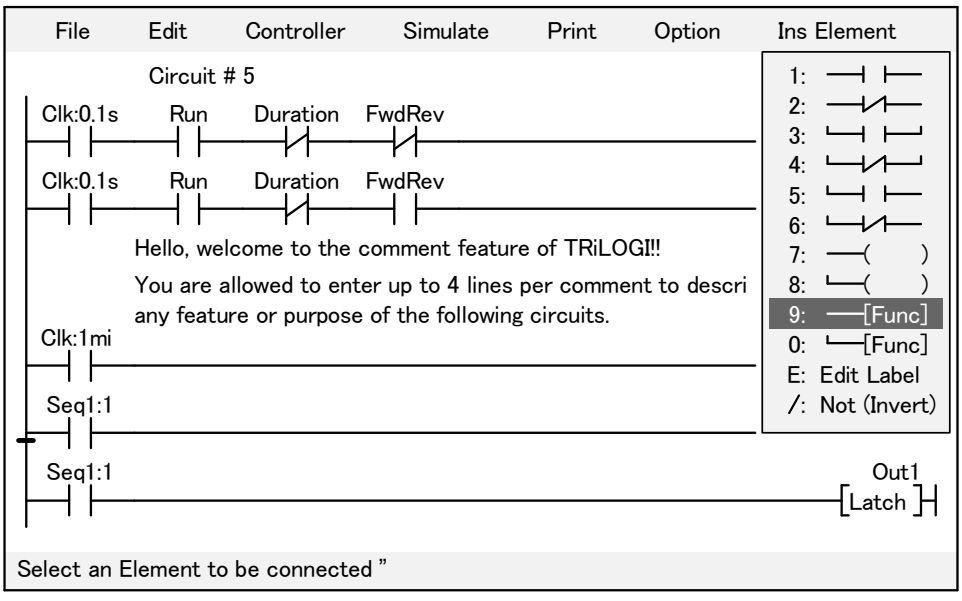
カスタム関数エディターは、現在編集中のウィンドウボックス内および他の関数番号のウィンドウボックスに選択行をコピーすることができます。コピーを行う場合はウィンドウボックス内で<Ctrl>+<C>キーを押すと、カソール位置からコピーする行を示す選択ブロックとメッセージ“Copy Text to Clipboard. Press <Enter> to complete...”が表示します。この選択ブロックは<←><↑><↓><→>及び<PgUp><PgDn>キーで選択範囲を指定し<Enter>キーで確定します。選択範囲のテキストデータは TRiLOGI のバッファに格納され、一度コピーコマンドで格納されたデータは、次のコピーが行われるか、TRiLOGI が終了されるまでバッファーに格納されます。(Windows のクリップボードライク。)
コピーしたテキストデータをカスタム関数のテキストボックスに張り付けるには、挿入する行にカソールを合わせて<Ctrl>+<Enter>キーを押します。カソール後行にテキストが記述されている場合は、以後が繰り下げられます。

3. カスタム関数 CusFn の実行

カスタム関数 CusFn が PLC のプログラム実行時にどの様に実行されるかご説明します。

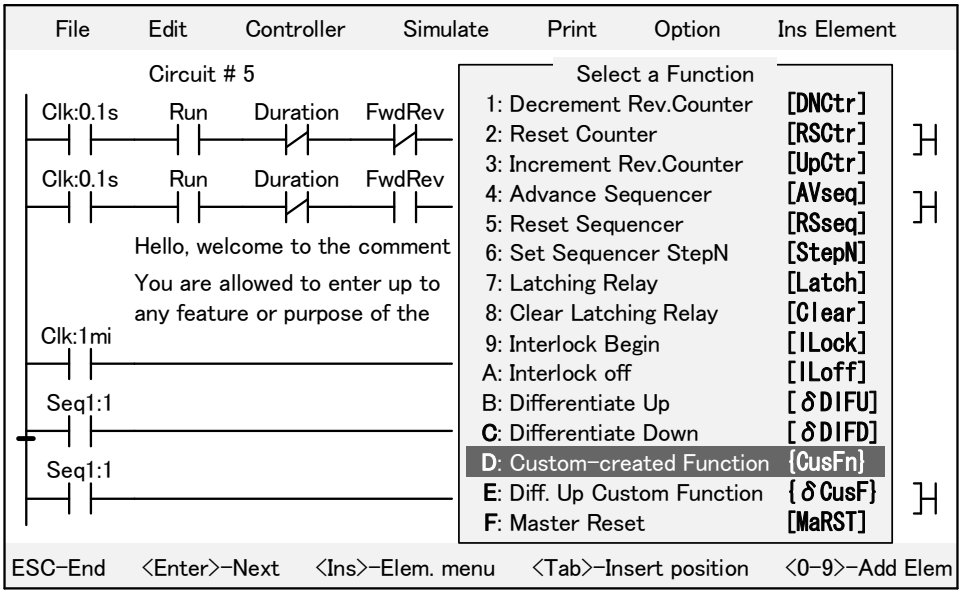
① ラダーロジックの特殊機能コイルとして呼び出し。---{CusFn}

TRiLOGI の回路作成モードで<Ins>キーを押して Ins Element のプルダウンリストの内部要素メニューをオープンします。アイテム“9: ---[Func]”か“0: └--[Func]”の特殊機能コイルを選択します。



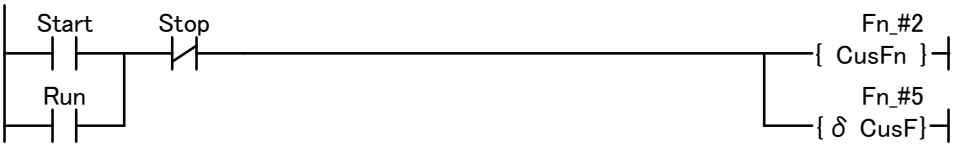
特殊機能コイルを選択します。すると、特殊機能コイルのアイテム選択メニューがポップアップします。

ここでカスタム関数 CusFn 選択用の“D: Custom-created Function {CusFn}”か“E: Diff. Up Custom Function {δCusF}”のアイテムを選択するとカスタム関数の関数番号“Enter Function # (1-128)”を入力するボックスがポップアップします。この入力ボックスに目的の関数番号を入力すると、ラダーロジック回路の実行（出力）コイルに、このカスタム関数 CusFn が接続されます。



“D: Custom-created Function {CusFn}”のアイテムはラダーロジックの入力側の要素（関数実行コイル起動条件）が ON になっていると、CPU スキャンタイム毎に、接続されたカスタム関数 CusFn を実行します。

“E: Diff. Up Custom Function {δCusF}”のアイテムはラダーロジックの入力側の要素（関数実行コイル起動条件）が OFF から ON になった場合（立ち上がり時）に、接続されたカスタム関数 CusFn を実行します。



カスタム関数 CusFn 作成上の注意事項

a. CPU はプログラムのラダーロジックの先頭回路よりスキャン開始し、ラダーロジック回路の出力コイルにカスタム関数 CusFn が接続定義されている場合、入力側の要素（関数実行コイル起動条件）によって、カスタム関数 CusFn を実行します。この場合、カスタム関数が定義接続されている以降の回路はラダーロジックが未実行のままです。カスタム関数 CusFn で直接 I/O ビットやデータを変更すると、以後のプログラムに影響する可能性があることをご注意ください。

b. INPUT[n]コマンドのデータは初めの CPU スキャン終了時のデータを得ますので、物理的な実際の入力データとは異なる場合があります。この様な場合は INPUT[n]コマンドを実行する前に、REFRESH ステートメントを使い、入力ビットを強制的にリフレッシュさせた後、INPUT[n]コマンドを実行します。

c. 同様に、SETBIT または CLRBIT ステートメントでの OUTPUT[n]コマンドのデータは、CPU スキャン終了時まで、出力ビットに物理的に書き込まれません。入力ビットの場合と同様に、すぐに書き込みが必要な場合は REFRESH ステートメントを使い、出力ビットを強制的にリフレッシュさせます。この場合、実行中のカスタム関数以降の回路はラダーロジックが未実行のままなので、他のプログラムへの影響を十分に考慮する必要があります。

② カスタム関数の外部割り込み入力ルーチン

外部割り込み入力機能で CPU は外部からの割り込み入力を検出すると、すぐにカスタム関数に作成した割り込みルーチンへ制御を移し、通常のラダーロジックスキャンタイムより高速に応答することができます。

a. 外部割り込み入力

M シリーズの外部割り込み用入力ビットに割り付けられた入力信号が、OFF から ON に、または ON から OFF の変化した場合に（エッジ検出トリガー）カスタム関数 CusFn に INTRDEF ステートメントで記述した関数番号のカスタム関数を呼び出して割り込みルーチンプログラムを作成することができます。

b. 高速カウンタ設定値トリガー

M シリーズの高速カウンタ用入力ビットに割り付けられた高速カウンタが設定値に達するとトリガーイベントが発生します。カスタム関数 CusFn に HSCDEF ステートメントで記述した高速カウンタの設定値の割り込みルーチンプログラムを作成することができます。

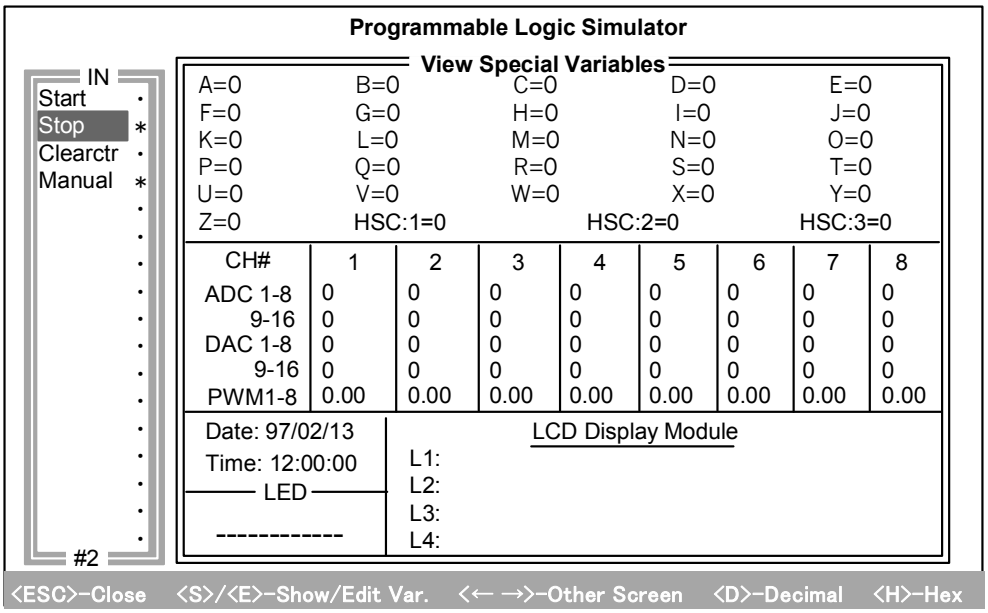
4. シュミレーション実行と TBASIC 変数

① カスタム関数 CusFn のシュミレーション実行

TRiLOGI Ver4.0 は TBASIC で記述したプログラムも完全にシュミレーション機能を活用できます。カスタム関数 CusFn を記述しラダーロジック回路を完成させた後、<F9>か<Ctrl>+<F9>を押すと、作成したプログラムのシュミレーションを行うことができます。シュミレーション画面でハイライトバーを入力カラムに合わせて<Enter>キーを押すと、この入力ビットの入力状態（ON/OFF）をトグル切り替えできます。この機能で作成したプログラムを画面上でシュミレーションできます。

② TBASIC 変数

シュミレーション画面で<V>キーを押すと、TBASIC 変数テーブルの View Special Variables の画面を表示できます。この画面の表は TBASIC で扱うすべての変数を 3 つのテーブル構造にして一覧表示します。View Special Variables の画面の 3 つの変数テーブルの切り替えは<←><→>キーを押して切り替えることができます。



a. システムテム変数テーブル

View Special Variables の 1 番目の画面は A～Z の 26 個の 32 ビット Integer 整数変数、HSC:1～HSC:3 の 3 つの高速カウンタ値、ADC 値、DAC 値、PWM 値、システム日付値･時刻値･曜日値、及び LCD ディスプレイコマンドが表示されます。シュミレーション実行中は日付値･時刻値･曜日値は PC 内部のリアルタイムク

ロックを参照しています。また ADC 値は実際に A/D チャンネルより変換データが格納された場合に表示されます。

b. **データメモリー変数テーブル**

View Special Variables の 2 番目の画面は DM[1]～DM[4000]の 16 ビット Integer 整数のデータメモリー変数を表示します。画面に表示されないデータメモリー変数は<PgUp><PgDn>キーを使って画面をスクロールできます。

c. **文字列変数テーブル**

View Special Variables の 3 番目の画面は A$～Z$までの 26 個の文字列変数を表示します。1 文字列変数に格納できる最大文字列数は 70 文字（1 バイト文字）です。

ラダーロジックの入力側の要素（関数実行コイル起動条件）が ON になっていれば、回路に接続されたカスタム関数 CusFn を連続して実行し、シュミレーション画面上の変数もリアルタイムに表示します。

③ カスタム関数エディターからの表示

カスタム関数エディターのウィンドウボックスで、プログラム作成中に<Ctrl>+<V>キーを押すと、直接、TBASIC 変数テーブルの View Special Variables の画面を表示することができます。各変数の格納状態を確認しながらプログラムを作成できます。

④ TBASIC 変数の変更

シュミレーションを実行中、TBASIC 変数テーブルの View Special Variables の画面がオープンしている時に、<E>キーを押すと直接次の TBASIC 変数の値を変更することができます。

A～Z、A$～Z$、	DM[n]、	DATE[n]、	TIME[n]、
INPUT[n]、	OUTPUT[n]、	RELAY[n]、	TIMERBIT[n]、
CTRBIT[n]、	TIMERPV[n]、	CTRPV[n]、	HSCPV[n]、

TBASIC 変数の変更方法は<E>キーを押して画面にポップアップする Edit Variable ボックスに“変数名 = 変更値”のステートメントで入力します。

例） A = 5000;
DM[99] = 5678;
OUTPUT[2] = &H01AB
B$ = “Welcome to TBASIC”

Edit Variable ボックスに入力した変更値は、すぐに View Special Variables 画面の変数テーブルやシュミレーション画面に反映されます。

⑤ TBASIC 変数のウォッチ表示

前項と同様に、TBASIC 変数テーブルの View Special Variables の画面がオープンしている時に、<S>キーを押すと TBASIC 変数の値を参照することができます。

TBASIC 変数の変更方法は<E>キーを押して画面にポップアップする Show Variable ボックスに“変数名”を入力します。

例）	A	<Enter>	（入力側）
	A = 1234		（応答側）
	OUTPUT[1]	<Enter>	（入力側）
	OUTPUT[1] = 4		（応答側）

INPUT[n]入力ビットや OUTPUT[n]出力の様に、View Special Variables 画面に表示されない変数を参照することができます。

⑥ Decimal 10 進と Hexadecimal 16 進の表示

TBASIC 変数テーブルの View Special Variables 画面はの値は通常 10 進で表示されます。<H>キーを押すと、この画面が表示している値を 16 進に変更できます。また<D>キーを押して 16 進の表示を 10 進に戻すことができます。

例）	A = 256	<H>	（10 進）
	A = 100		（16 進）
	A = 8FB	<D>	（16 進）
	A = 2299		（10 進）

5. オンラインモニタと TBASIC 変数

TRiLOGI Ver4.0 のオンラインモニタは E シリーズや H シリーズの Ver3.x と同様に、動作中の PLC とリアルタイムに PLC 内の変数を PC 画面上にオンラインモニタすることができます。オンラインモニタは PLC と PC を専用通信ケーブルで接続して、Controller メニューの On-line Mon/control コマンドを実行すると、オンラインモニタ・コントローラ画面が表示します。このオンラインモニタ・コントローラ画面はシュミレーション画面と同様の機能をサポートしていて、TBASIC 変数の確認の View Special Variables 画面、TBASIC 変数の変更 Edit Variable ボックス、TBASIC 変数のウォッチ表示 Show Variable

ボックス等シュミレーション画面と同様の操作が行えます。また TBASIC 変数の変更 Edit Variable ボックスで変更したデータはリアルタイムに PLC に反映され、ターゲット PLC をオンラインコントロールすることができます。

ターゲット PLC のリセットとポーズ

オンラインモニタ・コントロール実行中に、TBASIC 変数テーブルの View Special Variables の画面で<Ctrl>+<R>キーを押すと、ターゲット PLC 内部データをすべてリセットすることができます。また、<P>キーを押すと、動作中のターゲット PLC の実行プログラムをポーズ（停止）状態にすることができます。ポーズ状態のターゲット PLC を再び動作させるには、もう一度<P>キーを押してポーズ状態を解除できます。

6. エラーハンドリング

TRiLOGI Ver4.0 のコンパイラーは TBASIC でカスタム関数 CusFn エディタに作成した、プログラムのスペルミス・文法違い・無意味な表現など TBASIC の文法的なシンタックエラーを自動判定します。また実行時にはランタイムエラーを自動判定して的確なエラーメッセージを表示することができます。

① シンタックエラー

プログラムのスペル違い・文法違い・無意味な表現など文法的なシンタックエラーが検出された場合、自動的にカスタム関数 CusFn エディタを表示し、そのエラーが発生した行を強調表示してエラー内容を表示します。シンタックエラーがでる場合は、もう一度カスタム関数 CusFn エディタに記述したプログラムの見直しを行ってください。

	エラーメッセージ	コメント
1	Undefined symbol found	不明なシンボル。 TBASIC コマンドと TBASIC 変数を使用する。
2	Compiler internal error	重大トラブル。
3	“)” found without matching ”(”	“(” or “)” の欠落。
4	Integer expected	整数表現。
5	Value is out-of-range	コマンドの範囲外。
6	Duplicate line label number	重複した GOTO ラベル。 同一カスタム関数のユニークな定義

		同一カスタム関数のユニーク名で定義。
7	Undefined GOTO destination	GOTO ラベル不明。 GOTO ステートメントのラベル定義。
8	Invalid GOTO label	無効な DOTO ラベル。 @ #の範囲は 0～255 まで。
9	Type mismatch (numerics and strings may not mix)	型の不一致。 演算は STR$、VAL 等で型変換をする。
10	String is too long	文字列が多すぎる。 代入可能な文字数は最大 70 キャラクター。
11	Too many line label	ラベルが多すぎる。 同一カスタム関数内で使用できる GOTO ラベルは最大 20 まで。
12	Unknown Keyword	不明なキーワード。 コマンドおよびステートメントのスペルミス。
13	WHILE without ENDWHILE	WHILE … ENDWHIL ステートメントの ENDWHILE ブロックなし。
14	IF without ENDIF	IF … THEN … ELSE … ENDIF ステートメントの ENDIF ブロックなし。
15	FOR without NEXT	FOR … NEXT ステートメントの NEXT ブロックなし。
16	Expect keyword “TO”	FOR … NEXT ステートメントの ループ回数指定ブロックなし。
17	Must be an integer	整数値を指定。 整数値に文字列変数や定数は代入できない。
18	Must be an integer variable only	整数変数値のみ指定可。 整数変数値に整数定数値は代入できない。
19	Must be an integer constant only	整数定数値のみ指定可。 整数定数に整数変数は代入できない。
20	Must be a string	文字列を指定可。 文字列に整数変数や定数は代入できない。
21	Must be a string variables only	文字列変数のみ指定可。 文字列変数に文字列定数は代入できない。
22	Must be a string constant only	文字列定数のみ指定可。 文字列定数に文字列変数は代入できない。
23	Incomplete Expression	不完全な表記
24	String constant missing closing “	文字列定数が ” で完結されていない。
25	Must be Integer A to Z only	FOR … NEXT ステートメントのループカウンタは A～Z の変数で構成する。

② 実行時エラー

実行時エラーは前項の TBASIC の文法上のシンタックエラーとは別に、プログラムの実行時エラーを自動判定することができます。例えば、“A = B / C”は文法的には正しい表現ですが、もし C = 0 であれば 0 による除算になり、実行時エラーが発生します。作成したプログラムをシュミレーションで動作させ、もし実行時エラーが発生した場合、自動的にカスタム関数 CusFn エディタや回路を表示し、そのエラーが発生した行を強調表示してエラー内容を表示します。

	エラーメッセージ	コメント
1	Divide by zero	0 による除算
2	Call stack overflow! Circular CALL suspected !	Call によるスタックオーバーフロー
3	FOR-NEXT loop with STEP 0	FOR … NEXT のループカウンタ 0
4	SET_BIT position Out-of-range	SET_BIT は範囲外の設定値
5	CLR_BIT position Out-of-range	CLR_BIT は範囲外の設定値
6	TEST_BIT position Out-of-range	TEST_BIT は範囲外の設定値
7	STEPSPEED channel Out-of range	STEPSPEED 範囲外チャンネル値
8	Illegal Pulse Rate for STEPMOVE	STEPMOVE は違法な設定パルス
9	Illegal acceleration for STEPMOVE	STEPMOVE は違法な設定速度
10	STEPMOVE channel Out-of-range	STEPMOVE 範囲外のチャンネル値
11	STEPSTOP channel Out-of range	STEPSTOP 範囲外のチャンネル値
12	ADC channel Out-of-range	ADC は範囲外のチャンネル値
13	DAC channel Out-of-range	DAC は範囲外のチャンネル値
14	LED Digit # with (1-12) Only!	LED 表示は(1-12)のみ可能
15	PWM channel Out-of-range	PWM は範囲外のチャンネル値
16	LCD Line # must be (1-4) Only!	LCD 表示行は(1-4)のみ可能
17	PM channel Out-of-range	PM はは範囲外のチャンネル値
18	System Variable Index Out-of-range	システムインデックス変数は範囲外
19	Shifting of (A-Z) Out-of-range	A-Z のシフトは範囲外
20	Illegal Opcode - Please Inform Manufacture!	CPU オペコード異常
21	Timer or Counter # Out-of-range	タイマ・カウンタ値は範囲外

第3章　TBASIC ステートメント及び関数オペレータ

1. TBASIC ステートメント及び関数

TBASIC のステートメント及び関数の記述は大文字・小文字を厳密に区別しません。‘PRINT’も‘Print’も同様とします。しかし本マニュアルはプログラムの可読性から明白にするために場合別けして表記します。

① ステートメント

TBASIC のステートメントは連続した文字構文で構成され、ステートメント単独で使用するもの、決められた引数を必要をとするものがあります。引数が必要な場合はその引数の指定順にカンマ‘,’で区切ります。

② 関数

関数は決められた引数の値を関数にあたえて、その結果の値を返します。返される値は関数の定義によって Integer 整数値や String 文字列の場合があります。
また、関数の記述は‘関数名’+‘(’と‘)’で構成します。

例)　A$ = “Total is ¥” + STR$(B+C)

Integer 変数 B と C を加えてた値を文字列に変換し、先の文字列につなげて、文字列変数 A$に代入する。

ABS(n), ADC(n), MID$(A$,n,m), STRCOMP(A$, B$) 等

③ プログラムの記述

TBASIC のプログラムは複数行のステートメントにわたり構成され場合、それぞれのステートメントは改行で区別されます。例えば‘IF’ステートメントは以下の様に構成されますが、

```
IF A > B THEN
    C = D * 5
ELSE
    C = D / 5
ENDIF
```

また、前記のプログラムを一行に記述する場合は、コロン‘:’で区切り、以下の様に簡略化して構成することができます。

```
IF A > B * 5 : ELSE : C = D / 5 : ENDIF
```

2. 整数値

TBASIC は 32 ビット幅の Integer 整数演算をサポートしています。しかし組み込み変数 A～Z の Integer 整数値は 32 ビット幅ですが、正負データを表現するため、格納できるデータは-2^{31}～2^{31}です。また他のシステム変数やデータメモリーDM[n]の TBASIC 変数はすべて Integer 整数値は 16 ビット幅になります。この場合組み込み変数 A～Z と同様に正負データを表現するため、格納できる数値は-2^{15}～2^{15}(−32768～32767) になります。そして、これらの変数どうしの演算はすべて符号付 32 ビット幅で行うことができます。

① Integer 整数値

TBASIC は通常の Integer 変数に数値を代入する場合、通常表記はの 10 進値で代入されますが、プリフィックス‘&H’を付けて 16 進値で代入できます。

```
例)  A = 1007        (10 進の 1007)
     A = &H3FF       (16 進の 1007)
```

また、次の様に Integer 整数値を定数として指定することもできます。

```
A = 12345
IF A * 30 + 1345 / 123 > 100 THEN
    B$ = OK
ENDIF
```

整数値演算の注意事項

a. もし、Integer 変数に数値に 32 ビット幅を超える値を代入すると、変数に格納できないデータはオーバーフローし、符号データも異なってしまいます。この様なオーバーフローを防止するためも、格納データの大きさには十分にご注意ください。

b. 変数に 16 進値で負の値を代入する場合、代入する値の符号の表現にご注意ください。例えば、組み込み変数 A に数値‘-1234’を 16 進値で代入する場合、‘&HFB12E’でななく‘&HFFFFFB2’になります。

例えば、16 ビット変数のデータメモリーDM[1]に 10 進値の数値‘-1234’が格納されていて、プログラムのステートメントからこのデータメモリーの数値を 16 進値の数値で演算する場合、必ず 32 ビット変数のデータメモリー数値‘&HFFFFFB2’で数値を表現しなければなりません。

```
IF DM[1] <> &HFB2E CALL 5 : ENDIF                (誤)
IF DM[1] <> -1234 CALL 5 : ENDIF                 (正)
IF DM[1] <> &HFFFFFB2E CALL 5 : ENDIF                    (正)
```

② 組み込み変数

TBASIC の組込変数は、すべてのカスタム関数のプログラムから実行及び参照することができるグローバル値として使用することができます。

a. Integer 整数組み込み変数

A～Z の 26 個の 32 ビット幅の組み込み整数変数は、プログラムから型宣言をする必要なく、A～Z の文字を使用してプログラムを作成できます。

```
A = 5
B = 7
C = A * B
```

b. データメモリー変数

DM[1]～DM[4000]の 16 ビット幅のデータメモリー整数変数は、PLC のデータメモリーに格納できるデータメモリーです。プログラムから型宣言をする必要なく、DM[n] (n は 1～4000) の文字を使用してプログラムを作成できます。

```
DM[1] = 5          (データメモリーの 1 番に 5 を格納)
DM[A + B * 5]      (A, B は Integer 整数組み込み変数使用可能)
```

b. システム変数

システム変数は PLC のロジック状態を 16 ビット幅の値として表現します。

入力 Input/出力 Output/タイマ Timer/カウンタ Counter

TBASIC のカスタム関数で入力 Input/出力 Output/タイマ Timer/カウンタ Counter ビットを操作するには次の表の様なコマンドで目的のビットの操作を行うことができます。

I/O 番号表

割付番号	48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
ビット番号	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ワード番号	INPUT[3] OUTPUT[3] RELAY[3] TIMERBIT[3] CTRBIT[3] / INPUT[2] OUTPUT[2] RELAY[2] TIMERBIT[2] CTRBIT[2] / INPUT[1] OUTPUT[1] RELAY[1] TIMERBIT[1] CTRBIT[1]

INPUT[1] = 6 （入力割付番号 Input2 と Input4 ビット ON）

OUTPUT[3] = 1 （出力割付番号 Output33 ビット ON）

タイマ Timer/カウンタ Counter の現在値 PV

PLC のタイマ Timer[1]～[128]/カウンタ Counter[1]～[128]の現在値 PV を参照するには次の表の様なコマンドで目的の値を参照することができます。

```
A = timerPV[1]
IF ctrPV[1] = 100 THEN
```

日付 DATE および時間 TIME

PLC に内臓されている RTC リアルタイムクロックの日付 DATE および時間 TIME のデータを参照する場合は次の様なコマンドで目的の値を参照することができます。

日付 Date		時間 Time	
年 Year	DATE[1]	時 Hour	TIME[1]
月 Month	DATE[2]	分 Minute	TIME[2]
日 Day	DATE[3]	秒 Second	TIME[3]
週 Week	DATE[4]		

DATE[1] （cf. 1998）

DATE[4] （cf. 1=月, 2=火～7=日）

高速カウンタ

M シリーズは外部フィードバックエンコーダ等から入力される高周波数のパルス入力をカウントする高速カウンタ HSC をサポートしています。このパルスを高速カウンタ入力機能に割り当てた場合、入力された高速カウンタの現在値 PV を参照するには、コマンド HSCPV[1]～[8]で入力される高速カウンタの現在値 PV を参照できます。この場合 HSCPV[n]は 32 ビット幅になります。

③ 整数演算子 Integer Operator

TBASIC の整数演算子は数値の算術やビット演算および関係・理論をサポートしています。

a. 代入演算子

A～Z の 26 個の 32 ビット幅の組み込み整数変数やシステム変数等に、‘=’で数値を代入することができます。

```
H = 1000
X = H * I + J + len(A$)
```

b. 算術演算子 Arithmetic Operator

演算子	意味	使用例
+	加算	A = B + C + 25
-	減算	Z = TIME[3] - 10
*	乗算	PRINT #1 X * Y
/	除算	X = A (100 + B)
MOD	整数除算のあまり	Y = Y MOD 10

b. ビット演算子 Bitwise Logical Operator

ビット演算子を使用して、2 つの 16 ビット幅の数値を演算できます。

演算子	意味	使用例
&	理論積	IF Input[1] & &H02 ...
|	理論和	Output[1] = A | &H08
^	排他的理論和	A = RELAY[2] ^ B
~	不定ビット反転	A = ~TimerPV[1]

c. 関係・理論演算子 Relational Operator

関係・理論演算子を使用して、IF THEN や WHILL 条件判定ステートメントに構成できます。

演算子	意味	使用例
=	等しい	IF A = 100
<>	等しくない	WHILL CtrPV[0] <> 0
>	大きい	IF B > C / (D + 10)
<	小さい	IF TIME[3] < 59
>=	大きいか等しい	WHILL X >= 10
<=	小さいか等しい	IF DM[1] <= 5678
AND	かつ	IF A > B AND C <= D
OR	または	IF A <> 0 OR B = 1000

d. **関数演算子** Functional Operator

TBASIC は関数演算子として次の演算子をサポートしています。

ABS(n), ADC(n), CHR$(n), HEX$(n), STR$(n)

上記の詳細は後章をご参照ください。

④ 数値演算子の優先順序

TBASIC の整数値演算子は数値の算術やビット演算および関係・理論は、その計算実行に優先順序かあります。

優先順序	演算子	説明
高い	( )	カッコで囲まれたもの
↑	* / MOD	乗算/除算
	+ −	加算/減算
	−	負数演算
↓	& | ^ ~	ビット演算子
低い	= <> > >= < <=	関係・理論演算子

たとえば、X = 3 + 4 * (5 − 2) の計算を実行する場合、初めにカッコで囲まれた 5 − 2 を実行し、これに 4 を乗じ、そして 3 と加算した値が X に代入されます。ゆえに X の値は 15 になります。

3. 文字列データ String Data

文字列データは英数字 8 ビット ASCII の連続したデータとして扱うことができます。

① 文字列データ

文字列データはダブルクォーテーションマーク" "で囲まれた最大 70 キャラクターのデータ長の文字列を操作することができます。

"TBASIC made PLC numeric processing a piece of cake !"
"$102,345.00"

② 文字列変数 String variable

TBASIC の 26 個の組み込み文字列変数 A$～Z$には、最大 70 キャラクターのデータ長の文字列を代入・操作することができます。

③ 文字列演算 String Operator

a. 文字列変数への代入

TBASIC の 26 個の組み込み文字列変数 A$～Z$およびシステム変数等を参照して、文字列変数へ文字列を代入・操作することができます。

```
A$ = “Hello, Welcome To TBASIC”
Z$ = MOD$(A$, 3, 5)
```

b. 文字列変数の連結

2 つ以上の文字列を ‘+’ 演算子を使用して、文字列の連結をすることができます。

```
A$ = “TBASIC”
M$ = “Welcome, ” + A$ + “ world”
( M$ は Welcome, TBASIC world )
```

c. 文字列変数の比較

2 つの文字列の比較は STRCOMP(A$, B$) コマンド使用して、文字列が等しいか、等しくないかを演算します。TBASIC は文字列変数の比較に ‘=’ ‘<>’ 演算子はサポートしておりません。

```
A$ = “TBASIC”
```

d. 関数演算子 Functional Operator

TBASIC は関数演算子として次の演算子をサポートしています。

```
LEN(x$), MID$(A$, x, y), VAL(x$)
SETLCD 1, x$ PRINT #1 A$...
```

上記の詳細は後章をご参照ください。

第4章　TBASIC Programming Language Reference

ABS(x)

機能

値 x の絶対値を返します。

例

A = ABS(2 * 16 - 100)

コメント

値 A は 68 になります。

ADC(x)

機能

A/D 変換チャンネルに入力されるアナログ値をデジタル値に変換ます。
(変換チャンネル x は 1～16)

例

A = ADC(2)

コメント

T100MX は 12 ビットユニポーラ 0～4096 までの数値を格納できます。
TBASIC は 16 ビットバイポーラの-32768～32768 までの A/D 変換をサポートしておりますが、実際の入力は PLC の設定モデルによって異なります。
また、x に 1～16 以外の範囲外のチャンネルの数値を設定すると TBASIC はランタイムエラーになります。

ASC(x$, n)

機能

文字列 x$の n 番目の文字に対する ASCII コードを返します。

例

B = ASC("Test String", 6)

コメント

変数 B は"S"の ASCII 値 83 が代入されます。もし、n が 1 より小さい数値や指定した文字列より大きい場合は 0 を返します。

関連

CHR$( )

CALL n

機能

サブルーチンとして他のカスタム関数 CusFn を呼び出します。
呼ばれた側のカスタム関数が終了すると、呼び出したカスタム関数内の次のステートメントからプログラムは実行されます。
n はカスタム関数番号 1～128 を指定します。

例

IF B > 5 THEN CALL 8 : ENDIF

関連

RETURN

CHR$(n)

機能

n で指定した ASCII コードに対する文字を返します。
n で指定できるコードは 0～255 です。

例

C$ = "This is Message #" + CHR$(&H35)

コメント

CHR$(&H35)は 5 を返します。その結果 C$は This is Message #5 になります。

関連

ASC( )

CLRBIT v, n

機能

指定した I/O 番号 v のビット番号 n を OFF (0) にクリアーします。
ビット番号 v は Relay[n]、Output[n]等の I/O 番号 n で指定した 0～15 のビット番号のフラグを指定します。
また、ビット番号 v が 32 ビット幅の場合は、下位ビット 16 ビット演算操作になります。

例

CLRBIT Output[2], 11

コメント

Output 出力ビットの 28 番を OFF にします。

関連

SETBIT, TESTBIT

CLRIO labelname {M+モデルのみ}
SETIO labelname
TOGGLEIO labelname
TESTIO labelname

機能

ロジックの状態を操作して、入力、出力、リレー、タイマー、あるいはカウンターとのコンタクトビットが CusFn の範囲内にあります。
Labelname はラベルを参照します､名前がインプット､アウトプット､リレー、タイマーあるいはカウンターテーブルを定義します。
SETIO は ON をビットにセットします。
CLRIO は OFF でビットをクリアします。
TOGGLEIO は I/O ビットの現在のロジック状態を反転します。
TESTIO はファンクションを繰り返し、1 でビット ON、０でビット OFF を実行します。

例

SETBIT alarm
IF TESTBIT(alarm) THEN … ELSE … ENDIF

コメント

これらのファンクションはI/Oビットを操作するとき、SETBITとCLRBITを比較していっそう効率的に使い分け出来ます。
しかし、SETBITとCLRBITファンクションはコマンドによって影響を与えられているI/Oビットの状態に対して、影響を与えられるインデックスとビットポジションがコンパイル時間に固定していることを示すためにそれらが変数を使うことが出来るという利点を持っています。

ノート

ビットがカスタムファンクションで変えたアウトプットのリフレッシュコマンドが実行されていないなら、ラダーロジック操作の終わりにおいて物理的なアウトプットが更新されるであろうことを注意してください。

関連

SETBIT,CLRBIT,TESTBIT

DELAY n {M+モデルのみ}

機能

プロセスにｎミリ秒（0.1s）の時間の遅延を供給します。

例

DELAY 100

コメント

現在のカスタムファンクションに100msec(0.1s)の遅延を供給します。これが“brute force”遅延方法であり注意して使わなければ行けないことが重要です。Delay functionが実行される時、CPUは“Delay”によって指定された期間が終わるまでステートメントにおいて待ちます。この方法で全ての含まれるラダープログラムと他のカスタムファンクションのシステムサービスでの入力条件の変更応答ストップ（シリアルインプット、カウントダウンタイマー、ホストリンクコマンド等々）と同様CusFns作業間に入力作業を中断させ周期的に遅延させます。これは、もしプロセスの残りの入力が速い変化の応答をしなくてはならない場合望ましくないかもしれません。Delayのために0.1sより長くよい方法がDelayの終わりにおいてもうひとつのカスタムファンクションを機能させるために通常のPLCタイマーを呼び出し、タイマーを起動させ

使用することが出来ます。

T100MX と T100MD のためにこのファンクションにより供給される最小遅延は 10ms でその結果の Delay は 10ms です。このことは Delay155 を実行するならば実際の Delay は 160ms で Delay154 を実行するならば Delay は 150ms になります。

FOR … NEXT

機能

FOR と NEXT で囲まれた範囲を所定の回数だけ繰り返します。
所定の回数は、ループカウンター用の整数変数 A に代入し、初期値 x から最終値 y まで 1 回繰り返すたびに、きざみ値 z で示された値ずつ加減または減少させていった回数のループを繰り返します。また、きざみ値 z を省略した場合はデフォルトで 1 になります。

書式

```
FOR A = x To y STEP z
    …
NEXT
```

例

```
FOR I = 1 To 10
    FOR J = 100 TO 1 STEP -10
        DM[J] = DM[I]
    NEXT
NEXT
```

コメント

FOR … NEXT は多重ループも構成できます。

関連

WHILL … ENDWHILL

GetCtrSV(n)
GetTimerSV(n)

機能

カウンタおよびタイマ n 番の SV 設定値を返します。
n は 1～128 の数値を指定します。

コメント

また、カウンタおよびタイマ n 番の PV 現在値の参照は TimerPC[n]、CtrPV[n] のコマンドを使って、目的のカウンタおよびタイマ n 番の現在値データを直接参照することができます。

関連

SetCtrSV, SetTimerSV

GETHIGH16(v)

機能

23 ビット整数変数の上位 16 ビットを取り出します。

例

```
DM[1] = GETHIGH16(A)
SAVE_EEP GETHIGH16(&H12345678), 10
```

コメント

データメモリーやユーザ解放 EEPROM に 32 ビットデータを上位・下位 16 ビットずつ区切り格納する様な場合に使用します。

関連

SETHIGH16

GOTO @n

機能

現在のカスタム関数内の@n ラベルを持つプログラム行に無条件分岐をします。

例

```
@156 SETBIT 0, 3
...
GOTO @156
```

コメント

1 つのカスタム関数に定義できる GOTO ラベルの n は 1～255 までです。また指定した GOTO ラベルが未定義の場合エラーメッセージが表示されます。

HEX$(n)
HEX$(n,d) {M+モデルのみ}

機能

数値ｎを 16 進法の文字列に変換します。
HEX$(n,d)を使う場合、このファンクションは‘ｄ’の文字列に変換します。

例

```
A$ = HEX$(1234)
B$ = HEX$(1234,7)
```

コメント

A$に 1234 の 16 進データ 4D2 が出力されます。
B$に 1234, 7 の 16 進データ 00004D2 が出力されます。

関連

HEXVAL( ),STR$( ),VAL( )

HEXVAL(x$)

機能

文字列 x$を 16 進の数値に変換します。

例

B = HEXVAL(“123”) * 100

コメント

HEXVAL(“123”)は 10 進の 291 になり、B は 29100 が代入されます。

関連

HEX$(), STR$(), VAL()

HSTIMER n

機能

指定した PLC の通常タイマ n を HST 高速タイマ機能として割り当てます。

コメント

通常のタイマは 0.1sec ステップでカウントダウンしますが、HST 高速タイマ機能として割り当てると 0.01sec ステップでカウントダウンします。n で指定しないタイマは通常のタイマとして機能します。

HSCDEF ch, fn_num, value

機能

高速カウンタ機能のチャンネル ch 番を有効にして、パラメータを設定します。

ch	高速カウンタチャンネル番号 1～8
Fn_num	トリガー実行時のカスタム関数番号
value	トリガー設定値（32 ビット幅）

例

```
HSCPV[1] = 0
HSCDEF 1, 19, -3310003
```

コメント

高速カウンタの PV 現在値を 0 にクリアーした後、カウンタ値が-33100003 に達したら、カスタム関数 CusFn19 を実行します。

関連

HSCOFF

HSCOFF ch

機能

高速カウンタ機能のチャンネル ch 番を無効にします。

ch 高速カウンタチャンネル番号 1～8

コメント

CPU の負荷を軽くするために、不要になった、高速カウンタ機能をチャンネルを無効にします。

関連

HSCDEF

IF … THEN … ELSE … ENDIF

機能

条件判断のフロー制御ステートメントを構成します。
条件 expression が真の場合 THEN～ELSE のステートメントを実行し、偽の場合は ELSE 以降のステートメントを実行します。また ELSE ステートメントを省略した場合、条件 expression が偽の場合はステートメントを実行せすステートメントは無視され実行されません。

書式

```
IF expression [THEN]
    …
[ELSE]
    …
ENDIF
```

例

```
IF A >= 100
    B$ = “OK”
ELSE
    B$ = “NG”
ENDIF
```

INCOMM(ch)

機能

シングルを返すために 8 ビットのバイナリ―データが COM から得られます。チャンネル ch

ch は 1～8 間の数値の定数となります。実際のターゲットハードウェアーは、正しいポート＃を決定します。このファンクションは、もしシリアルポートにデータが待機していないなら 1 を返します。

例

```
FOR I = 1to100
    DM[ I ] = INCOMM(2) :
    IF DM[ I ] < 0 RETURN :ENDIF
    NEXT
```

コメント

通常 PLC は、プログラムにシリアルデータが無いか COM ポートをチェックする必要が無いように、その COM ポートに接続してデータをバッファに入れます。プログラムがそれまで COM バッファで全てのデータを読み込むために上記の例である FOR..NEXT ループが確認し、使うことの出来るデータを処理する準備が出来ている時バッファからであることを示します。

関連

INCOMM( ),PRINT#,OUTCOMM

INPUT$ (ch)

機能

指定したシリアルポート COM のチャンネル ch に入力される文字列を返します。

ch　　COM シリアルポートのチャンネル番号 1～8

例

```
D$ = INPUT$ (2)
```

コメント

シリアルポート COM2 の文字列を文字列変数 D$に代入します。
外部からシリアルポートに入力される文字列データはキャリッジリターン CR または ASCII 値 13 で文字列の最後と判定します。

関連

PRINT

INTRDEF ch, fn_num, edge

機能

外部割り込み入力のチャンネル ch 番を有効にして、パラメータを設定します。

ch	外部割り込み入力のチャンネル番号 1～8
Fn_num	トリガー実行時のカスタム関数番号
value	トリガー検出（正数=立ち上がり / 0 はたは負数=立ち下がり）

関連

INTROFF

INTROFF ch

機能

外部割り込み入力のチャンネル ch 番を無効にします。

関連

INTRDEF

LEN(x$)

機能

文字列 x$のキャラクター数を返します。

例

L = LEN(“This is a test string” + CHR$(13))

コメント

変数 L には 22 が代入されます。

LET

機能

変数に表現の文字を割り当てます。

文章

[LET] variable = expression

例

LET D = 11

A$ = “Welcome to TBASIC”

コメント

LET の内容は任意です：例えば、表現を可変的な名前に割り当てるのに十分です。等しい側面の両側の上で可変的なタイプはこれも同様です。例えば、文字が数値に割り当てられないときは、逆もまた割り当てることが出来ません。

重要

a) 16 ビットの変数を 32 ビットの整数に割り当てるとき、ただ 32 ビットの整数より低 い 16 ビットだけが割り当てられます。それは、プログラマーがもし 32 ビットの数が 16 ビットの数内｛-32768～32767 の間にある｝から出ているなら、特別な指示をしなくてはなりません。

b) もし否定的な 16 ビットの数が 32 ビットの整数変数に割り当てられるならば、指示されたビットは 32 ビットに延長されます。

```
例 DM [ 1 ] = -123.
   A = DM [ 1 ]
```

-123 の 16 ビットの 16 進法の値は、&HFF85 です、しかし、A が 16 進法 &HFFFFFF85 を割り当てます。しかしそれらの少数表現は同じです。

LOAD_EEP(addr)

機能

SAVE_EEP コマンドによってユーザ開放 EEPROM にストアーした 16 ビット整数値を返します。

addr EEPROM アドレス（1～2000）

※ PLC の設定モデルによりサイズは異なります。

例

Relay [1] = LOAD_EEP(10) : A = LOAD_EEP(2)

関連

SAVE_EEP

LSHIFT i, n

機能

整数変数 i (Relay[n]、Output[n]、DM[n] 等) のデータ 1 ビット左にシフトします。整数変数 I を 1 ビット左にシフトする場合、連続した n 個のワード番号の I/O も同時にシフトできます。1 つのをワード番号の I/O をシフトする場合は n に 1 を指定します。

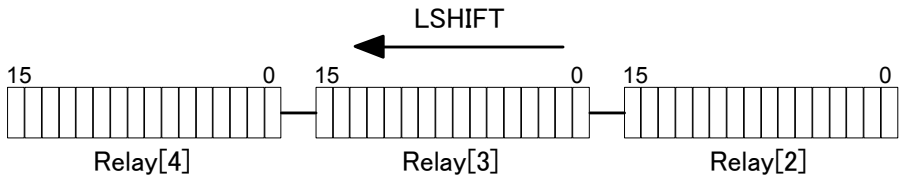
i	整数変数 i
n	連続するワード番号数

例

LSHIFT Relay[2], 3

コメント

Relay 内部リレーのワード番号 Relay[2]-Relay[3]-Relay[4]をまとめて 1 ビット左にシフトします。Relay[4]の最上位ビット番号 15 のビットデータは左シフトのため消失します。



関連

RSHIFT

MID$(x$, n, m)

機能

文字列 x$の n 開始位置から m 個のキャラクターを取り出し返します。

n	開始位置 1～255
m	取り出すキャラクター数 0～255

例

A$ = MID$(“Welcome to TBASIC”, 4, 7)

コメント

A$には come to が代入されます。

NETCMDS$ (ch,x$)

機能

このファンクションは、別の M シリーズかあるいは、H シリーズの PLC にリンクコマンド文字列がシリアルポート＃sh により x$で指定したマルチポイントのホストへ行かせます。それは、他のＰＬＣから指定された応答時間を待ちます、そしてこの応答文字列はそこから返されます。

Ch－コミュニケーションポート＃に参照します。詳細は、ターゲット PLC を参照してください。

X$－フレームチェックシーケンス（FCS）とターミネーターキャラクターを除いてマルチポイントのフォーマットで正確なホストコマンドを含みます。NETCMDS$ファンクションが自動的に FCS を計算して、おわりに x$とターミネータキャラクターを添えます、そして COM #ch により他の PLC に送られます。

ノート

1) もしターゲット PLC から応答が無いときは、このファンクションはストリングを返します。

2) このファンクションは、応答ストリングの FCS をチェックします、FCS が間違っている場合それはエラーでシリアルレセプションとストリングスを返します。

例

A$ = NETCMD$ (3,”@05RI00)”

コメント

ID = 05 で PLC のインプットチャンネル＃０を読むのに COM#3 の PLC に接続します。応答ストリングは、A$に割り当てられます。

OUTCOMM n,x

機能

このステートメントは、COM ポート#n により 8 ビットの 1 バイトのデータ‘x’を送ることが出来ます。このコマンドは、PRINT#n コマンドが CHR$(０)を送ることが出来ないために追加されました。ゼロが TBASIC で文字列の終わりとして認識されたとき、CHR$(０)を送るために PRINT#ｎのステートメントを使う場合は、無効になります。

例

OUTCOMM 2,225

PAUSE

機能

カスタム関数 CusFn 内に PAUSE コマンドを記述して、プログラムにブレークポイントを置けます。プログラムが PAUSE コマンドに達すると、実行中のプログラムは中断し、関連した変数等をデバックすることができます。プログラムの再実行は<P>キーを押してポーズ状態を解除します。

PIDdef ch, lmt, P, I, D

機能

比例･積分･微分制御（PID コントロール）の設定パラメータを使用する出力チャンネルに指定します。

ch	出力チャンネル番号 1～16
lmt	演算結果の最大値
P	比例ゲイン(K_P)
I	積分ゲイン(K_I)
D	微分ゲイン(K_D)

PID コントロールの変換演算式

$$G(s) = K_P + \frac{K_I}{s} + K_D s$$

$$K_P = \text{Proportional Gain} = \frac{1}{\text{比例幅}}$$

$$K_I = \text{Integral Gain} = \frac{1}{\text{内部タイムコンスタント}}$$

パラメータの lmt、P、I、D は 16 ビットおよび 32 ビット整数値を指定します。lmt は PIDcompute(ch, E)コマンドで得られた値を超えて設定できません

関連

PIDcompute( )

PIDcompute(ch,E)

機能

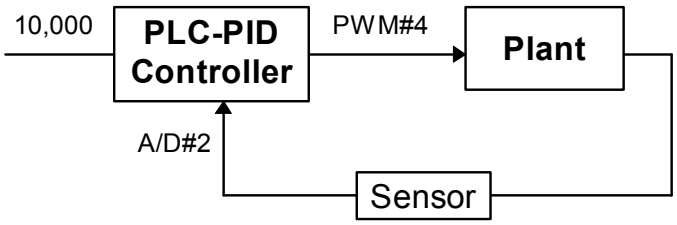
PID コントロールの設定コマンド PIDdef で定義されたチャンネルの PID コントロールパラメータに基づき演算実行をするコマンドです。微分データや積分データは PLC 入力チャンネル内部に格納され、PID コントロール変換演算式によって 32 ビット自動演算します。PIDcompute()は PIDdef()で設定した lmt を参照して自動コントロールします。

ch　　チャンネル番号 1～16

E　　クローズドループエラー (Err)

PID コントロールは AD・高速カウンタ・PULSEFREQUENCY 等のフィードバックデータを得ることができます。得た結果値はスケールされた Err 値のセットポイントに割当られます。

例



E.g. Implementing Closed-loop Digital Control with PID computation function

```
E = 10000 - ADC(2) * 20
A = PIDcompute(5, E)
setPWM 4, (A + 8000) / 100
```

コメント

セットポイント値 10,000、AD チャンネル#2 から外部センサ信号がフィードバックで入力されます。PID コントロールチャンネル#5 を使用して、目的の出力ビット PWM#4 出力に PID コントロールされたデータを出力します。

PRINT #n x$; y; z ...

機能

PLC の指定した COM シリアルポートに ASCII キャラクターの文字列･数値を送出します。

#n	シリアルポートチャンネル番号 1～8
x$	送信文字列
y (z)	送信数値

PRINT ステートメントで送出するデータは ASCII データに変換されで指定したシリアルシリアルポートに送られます。送出する文字列･数値が複数ある場合、1 つの PRINT 以後のステートメントが送出されます。この時、; セミコロンを付けると自動的に送出データ末尾に CR(ASCII13)キャリッジリターンが付加されます。

例

```
PRINT #2 “The value of A + B = ” ; A + B ;
PRINT #2 A$
```

コメント

高速カウンタの PV 現在値を 0 にクリアーした後、カウンタ値が-33100003 に達したら、カスタム関数 CusFn19 を実行します。

関連

INPUT$( )

PMON ch
PMOFF ch

機能

PMON は指定したチャンネル ch のパルス計測を有効にし、PMOFF は PMON で指定したチャンネル ch のパルス計測を無効にします。
パルス計測を有効にした後、PULSEPERIOD(ch)、PULSEWIDTH(ch)コマンドでパルス計測の設定パラメータを指定します。

例

PMON 1
PMOFF 5

関連

PULSEPERIOD( ), PULSEWIDTH( )

PULSEFREQUENCY (ch)
PULSEPERIOD (ch)
PULSEWIDTH (ch)

機能

PULSEFREQUENCY(ch) は指定したパルス計測チャンネル ch の最終入力パルスの周波数 Hz を返します。PULSEPERIOD(ch) は指定したパルス計測チャンネル ch の入力パルスのパルス幅を 1/000sec で返します。
また、PULSEWIDTH(ch) は指定したパルス計測チャンネル ch の入力パルスのパルス周期を 1/000sec で返します。

ch　　パルス計測入力チャンネル番号 1～8

これらのコマンドは PMON(ch)でパルス計測をするパルス計測チャンネルを有効にしなければ使用することはできません。

例

A = PULSEWIDTH(1)

関連

PMON, PMOFF

READMODBUS (ch,DeviceID,address) {M+モデルのみ}

機能

自動的に MODBUS ASCII デバイスを探り MODBUS ASCII プロトコルを使っている 16 ビットのレジスタデータを返します。コミュニケーションポートのボーレートは、SETBAUD コマンドにより変えられなかったときその COM のデフォルトのボーレートとなります。

Ch	PLC の COMM ポート番号（１～８）
DeviceID	MODBUS デバイス（1～255）のデバイス ID
Address	MODBUSデバイスでホールドされているゼロオフセットによるアドレス

例

relay [3] = READMODBUS(3,5,101)

コメント

リレーはID=05でレジスタのオフセットアドレス101（MODBUS内では＃40102とホールドしていることになります）から MODBUS デバイスが得られた 16 ビットのデータを含みます。リレーの中にこれを読む［ ］チャンネルがラダーロジックによりビットレベルに扱います。これはどのデータメモリにでも同様に読み込むことが出来ます。

関連

NETCMDS$( )

REFRESH

機能

強制的に物理的な入出力ビットを直ちに I/O リフレッシュされます。
通常プログラムから SETBIT・CLRBIT コマンドで Output[n]出力ビットを操作した場合、ラダーロジックスキャンの最終時に I/O リフレッシュが実行され

ますが、REFRESH コマンドを実行することにより、実際の変更のタイムラグを防止することができます。

REM (or ‘)

機能

プログラムリストにコメントを記述します。
プログラムに REM または ’ がある場合、コンパイラはそのコメント行を無視します。

例

```
REM     Waiting for the right time to turn on
‘       This is also a remark line.
```

RESET

機能

カスタム関数 CusFn プログラムから実行中の PLC プログラムを強制的にリセットします。DAC 出力や PWM 出力はすべて OFF 状態になります。このコマンドは TRiLOGI ラダープログラムの[MaSRT]マスターリセット機能と同等な働きをします。またラダープログラムに[1st.Scan]ファーストスキャンフラグを使用している場合は 1 スキャン実行後の ON 状態になります。

RETURN

機能

実行中のカスタム関数 CusFn プログラムから強制的に抜け出すステートメントです。通常、カスタム関数のプログラム最終行でこのカスタム関数のプログラムから自動的に抜け出しますが、RETURN ステートメントが記述されている行がある場合、その後のプログラム行は無視してこのカスタム関数から強制的に抜け出します。

関連

CALL

RSHIFT i, n

機能

整数変数 i (Relay[n]、Output[n]、DM[n] 等) のデータ 1 ビット右にシフトします。整数変数 I を 1 ビット右にシフトする場合、連続した n 個のワード番号の I/O も同時にシフトできます。1 つのをワード番号の I/O をシフトする場合は n に 1 を指定します。

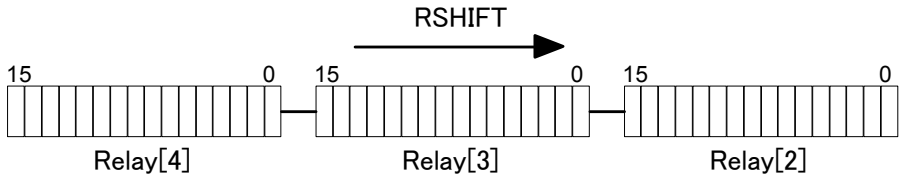
I	整数変数 i
n	連続するワード番号数

例

RSHIFT Relay[2], 3

コメント

Relay 内部リレーのワード番号 Relay[2]-Relay[3]-Relay[4]をまとめて 1 ビット右にシフトします。Relay[2]の最下位ビット番号 0 のビットデータは右シフトのため消失します。



関連

LSHIFT

SAVE_EEP data, addr

機能

PLC のユーザ開放 EEPROM に 16 ビット整数データを格納します。また、32 ビットデータを格納する場合は上位 16 ビットデータは無視され、下位 16 ビットデータのみ格納されます。32 ビットデータをすべて格納する場合は、GETHIG16()関数を使用して 32 ビットデータを上位 16 ビット・下位 16 ビットデータに別けて格納することができます。

data	16 ビット整数データ
addr	EEPROM アドレス（1～2000）

※ PLC の設定モデルによりサイズは異なります。

例

SAVE_EEP Relay[1], 100

関連

LOAD_EEP, SETHIGH16(), SETHIGH16()

SETBAUD ch, baud_no

機能

PLC の指定した COMM シリアルポートチャンネル ch の通信速度を設定します。M シリーズ PLC の通信パラメータは 8 ビットデータ・1 ビットストップ・ノンパリティです。通常では、一度そのチャンネルに通信速度を設定すると、その設定は以降引き継がれて有効になります。

ch　　シリアルポートチャンネル番号 1～8

baun_no　　設定通信速度 1～4

※ PLC の設定モデルにより設定通信速度は異なります。

例

SETBAUD 3, 3 ‘ Set COMM3 9600 bps

SETBIT v, n

機能

指定し I/O ワード番号 v のビット番号 n を ON (1) にセットします。
ビット番号 v は Relay[n]、Output[n]等の I/O 番号 n で指定した 0～15 のビット番号のフラグを指定します。
また、ビット番号 v が 32 ビット幅の場合は、下位ビット 16 ビット演算操作になります。

例

SETBIT Output[2], 11

コメント

Output 出力ビットの 28 番を ON にします。

関連

CLRBIT, TESTBIT

SetCtrSV n, value
SetTimerSV n, value

機能

カウンタおよびタイマ n 番の SV 設定値を設定します。

n	1～128 の数値を指定します。
value	0～9999 の数値を指定します。

例

SetCtrSV 10, 1234
SetTimerSV 3, GetTimerSV(3) + 10

コメント

カウンタ Counter#10 の SV 設定値を 1234 にします。
タイマ Timer#3 の SV 設定値を 10 倍します。
また、カウンタおよびタイマ n 番の PV 現在値の参照は TimerPC[n]、CtrPV[n] のコマンドを使って、目的のカウンタおよびタイマ n 番の現在値データを直接参照することができます。

関連

GetCtrSV(), GetTimer()

SETDAC n, x

機能

PLC の指定した D/A 変換チャンネル n からアナログ信号を出力します。

ch　　D/A 変換チャンネル番号 1～16

x　　D/A 出力用整数値

※ PLC の設定モデルにより設定は異なります。

一度 D/A 変換チャンネル n を指定してアナログ出力を行うと、次のコマンド実行まで、この出力状態を維持します。

例

SETDAC 5, (A + B) *16

SETHIGH16 v, data

機能

32 ビット整数変数の上位 16 ビットを割当ます。これは、EEPROM または DM[n] のいずれかから獲得された 2 つの 16 ビットデータを使って 32 ビット整数値を組み立てます。

例

A = DM[2]
SETHIGH16 A, DM[1]

関連

GETHIGH16( )

SETLED n,m,value

機能

nth デジットからデジットのｍ番号を占有し、PLC の組み込みの 7 セグメントの LED ディスプレイに整数値を表示します。先行するゼロがもし値がｍにより指定した者より少ないデジットを占めるならディスプレイにおいて左側に表示されます。

しかしもしｍが１以下（例えばｍ＝０）なら値が数の値としてよりも表示されるべきひとつの 8 ビットの ASCII 文字として認識されます。特別なシンボル

がもし LED ドライバに対応する ASCII 文字を表示させることが出来るなら LED パネル上に表示されます。

ｎは 1 から 16 の間の数となります。デジットポジションは左から右へ表示されます。LED デジットで数えられるデジットは＃1 です。TRiLOGI は最高 16 の LED デジットをサポートしています。PLC 上の LED の実際の数は 0～16 までで変動します。この場合利用可能なデジットだけ表示されます。

Value は 16 または 32 ビットの整数値となります。以前にセットされた LED ディスプレイは、同じデジット上の次の SETLED ステートメントが実行されるまで値を変えずにとどめておきます。TRiLOGI のシュミレータ上に SETLED の結果は特別なスクリーンと共に表示されこれは、シュミレーション状態のとき＜V＞キーを押すことにより見ることが出来ます。

例

SETLED 5,4,89

コメント

LEDデジットは5～8(表示は左から右へ)ディスプレイは0089 を表示します。

SETLCD n, offset, x$

機能

M シリーズオプションの外部 LCD 表示器 MDS100 に文字列を設定します。
MDS100 は 4 行×20 キャラクターの文字列を表示できます。

n	表示行 1～4
offset	表示位置オフセット
x$	送出文字列

一度 SETLCD ステートメントで MDS100 に書き込まれた文字列は、次の書き込みが実行されるまで、表示された文字列は維持されます。

例

SETLCD 1, “This is a 1x20 LCD”

SETPWM n, x, y

機能

PLC の PWM 出力のチャンネル n にデューティー比 x/100%で設定周波数 y Hz の PWM(パルス幅変調)出力を行います。

n　　PWM 出力チャンネル 1～8

x　　デュティー比 x/100% 0～10000

y　　設定周波数 Hz

一度 SETPWM ステートメントによって PWM チャンネルの出力されたデータは、次の SETPWM ステートメントまで設定出力状態を維持します。また、x の値を 10000 以上にすると、デューティー比 100%になります。

例

SETPWM 1, 4995

コメント

PWM 出力チャンネル 1 にデューティー比 49.95%で PWM 出力を行います。
実際の PWM 出力の分解能は PLC のモデルに依存します。M シリーズの場合、PWM 出力の分解能は 10 ビットになりますので、1/1024 = 0.1%になります。この場合、設定出力の 49.95%は自動的に 50%に丸められます。

SETPASSWORD string {M+モデルのみ}

機能

このステートメントが実行される時、PLC は正確にホストリンクコマンドが他のコマンドに送った "PWxxxx…xx" が STEPASSWORD コマンドで定義されたのと同じ文字列 "xxxx…xx" (19 以上の文字では不可)を含んでいないと応答しません。全ての他のコマンドが“password error”状態を示しているとき“PWER”の応答を受け取ります。正しいパスワードを入力したときに、PLC は正常に作動して、全てのホストリンクコマンドに応答します。どのストリングでもない“PW”がホストリンクコマンドの実行が無許可のアクセスを防ぐためにパスワードロックを返します。

例

SETPASSWODO “I LOVE TRiLOGI”

TL4.1.EXE を使うとき、エディターはもしそれが今までパスワードによりロックされた PLC に通信するなら、自動的にパスワード入力をするように促します。パスワードが大文字、小文字の違いを識別することに注意してください。ロックされたパスワードは、TRiLOGI の以前のバージョンによってのアクセスをすることが出来ません。

コメント

この特徴は、自動応答モデムにリンクされるつながりのない PLC を保護するために使用します。パスワード保護なしでも TL4.1.EXE でダイアルインし PLC のフル制御を行うことも出来ます、しかしそれはセキュリティーの上で問題になるかもしれません。TL4.1.EXE がモデムの接続をきるとき、自動的に別のシーケンサの呼び出しにより無許可のアクセスがないように、パスワードロックを再度掛ける“PW”コマンドを実行します。PLC ソフトウェアの上で、周期的にこのコマンドにより最大の保護を行うために PLC を再度保護させるようにタイマーを使うことも出来ます。同日の異なった時間に違うパスワードを使用したり、より安全を保つためにパスワードをローテーションさせて使うことも出来ます。

STEPCOUNT (ch)

機能

STEPMOVE コマンドでステッピングモータのパルス出力チャンネル ch からステッピングモータコントローラへ駆動用出力パルスを送出している場合、この STEPCOUNT を使って、ステッピングモータに送出しているパルス出力を計測・モニタすることができます。

ch　　ステッピングモータパルス出力チャンネル

STEPCOUNTABAS (ch) {M+モデルのみ}

機能

ステッピングモーター を元のポジションに返すチャンネル(#ch)。もし STEPHOME コマンドが実行されたときに、ステッピングモーターが動作しない場合ゼロに返します。

STEPHOME ch {M+モデルのみ}

機能

ステッピングモーターの現在のポジションカウンターをゼロにセットして下さい。これはステッピングモーターの新しい“HOME”ポジションを示します。このコマンドは、ステッピングモーターが HOME ポジションとみなされる特定のポジションにあるときのみ実行されます。STEPMOVEABS コマンドがその後実行した全ての定義された HOME ポジションに比例します。

STEPSPEED ch, pps, acc

機能

PLC のステッピングモータパルス出力のチャンネル ch に動作スピード pps 加速・減速 scc パラメータを設定します。ch にステッピングモータパルス出力チャンネルを指定し、pps に動作スピードを 1 秒間あたりの発生パルス数を指定します。また、acc に停止状態からフルステップに至るまでのステップ数とフルステップから停止状態に至るまでのステップ数の加速・減速ステップ数を指定して、モータ駆動用の自動台形パルスを自動演算します。

ch	ステッピングモータ駆動用出力チャンネル 1～8
pps	動作スピードパルス
acc	加速・減速パルス

STEPSPEED コマンドは STEPMOVE コマンドを実行する前に必ず設定を行います。一度 STEPSPEED コマンドよって設定されたステッピングモータパルス出力チャンネルデータは、次の STEPSPEED コマンドまで設定出力状態を維持します。

例

STEPSPEED 2, 2000, 20

コメント

PLC のステッピングモータパルス出力のチャンネル#2 は、STEPMOVE コマンドが実行されると、20 パルスで加速した後、フルステップ 2000 パルスで回転するように自動台形パルスを演算したパルスをステッピングモータパルス出力のチャンネル#2 に出力します。

$$a = \cdot\frac{V^2}{2S} = \cdot\frac{2000^2}{2x20} = 100{,}000\ pulse/s^2$$

STEPMOVE ch, count, r

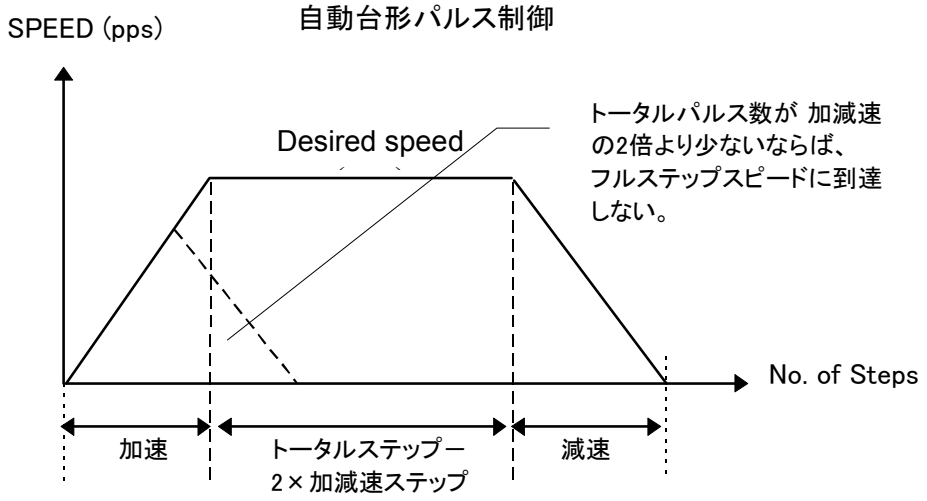
機能

PLC のステッピングモータパルス出力のチャンネル ch にステッピングモータ駆動用パルス出力します。出力されるパルスは、このコマンドを実行する前に STEPSPEED コマンドで設定したパラメータをもとに自動台形パルスを出力し、count 出力パルス数で指定したパルス(位置)で停止します。出力パルス数を自動カウントして停止すると、r 番の指定した Relay #r 内部リレーが ON し、出力終了フラグのシグナルとして使用することができます。

ch	ステッピングモータパルス出力チャンネル 1～8
count	出力パルス数 32 ビット整数値 (1～2,147,483,647 [2^{31}])
r	出力終了フラグ用内部リレー

再度 STEPMOVE コマンドが実行されると r 番の指定した Relay #r 内部リレーはパルスが出力される前に OFF にクリアーさ、出力パルス数を自動カウントして停止すると ON しれます。

また、STEPMOVE コマンドが実行中に、再度同じ出力チャンネルに STEPMOVE コマンドを実行すると、動作中のコマンドを優先して無視されます。は STEPSTOP コマンドでタパルス出力を停止することができます。



例

STEPMOVE 1, 5000, 10

コメント

PLC のステッピングモータパルス出力のチャンネル#1 に出力パルス数 5000 パルスを出力し、出力終了フラグ用の内部リレーRelay10 を指定します。

関連

STEPCOUNT( ), STEPSPEED, STEPSTOP

STEPMOVEABS ch,position,r {M+モデルのみ}

機能

この新しいコマンドは、ステッピングモーターのチャンネル(#ch)をポジションパラメータにより示された元のポジションに戻すことを許可します。動作の終了時にリレー(#r)は ON となります。ポジションは、-2^{31}～＋2^{31} の間となります。（±2×10^9）元のポジションは“HOME”ポジションから最後の動作に関して計算されます。（HOME ポジションは、STEPHOME コマンドが実行される時、セットされます。）スピードと加速プロフィールは、オリジナルのコマンドセットの STEPSPEED コマンドにより決定されます。

このコマンドは、自動的にパルスの番号と現在の位置に関して、新しいポジションにステッピングモーターを移動させるよう要求された指示の値を計算

します。現在の位置は STEPCOUNTABS（）ファンクションによりいつでも決定できます。

STEPMOVEABS コマンドが実行される時に、このコマンドあるいは、STEPMOVE コマンドの再実行が、全ての動作が STEPSTOP コマンドにより終了されるか、あるいは中止されるまで実行されません。

関連

STEPCOUNTABS,STEPHOME,STEPSPEED,STEPMOVE,STEPSTOP,
STEPCOUNT

STEPSTOP ch

機能

STEPMOVE コマンドでステッピングモータのパルス出力チャンネル ch からステッピングモータコントローラへ駆動用出力パルス出力を停止します。

ch　　ステッピングモータパルス出力チャンネル

例

STEPSTOP 2

コメント

STEPSTOP コマンドで停止した場は、出力します。STEPMOVE コマンドで指定した内部リレーRelay #r は ON しません。

関連

STEPCOUNT( ), STEPSPEED, STEPMOVE

STR (n)
STR$ (n,d)　{M+モデルのみ}

機能

数値ｎを 10 進法の文字列に変換して返します。

STR$(n,d)を使うときこのファンクションは‘ｄ’の文字列に変換して返します。

例

```
A$ = STR$(-1234)
B$ = STR$(-1234,7)
```

コメント

A$が - 1234 の文字列を返します、B$が - 001234 の文字列を返します。

STRCOMP(A$, B$)

機能

文字列 A$と B$の大小関係を比較して結果を返します。
A$と B$が等しければ 0 を返します。A$が B$より ASCII オーダーで大きければ正数(1)を返します。また A$が B$より ASCII オーダーで小さければ負数(-1)を返します。

例

```
IF STRCOMP$(A$, B$) = 0 THEN
    STEPMOVE 1, 1000, 1
ENDIF
```

コメント

文字列 A$と B$が同じなら、ステッピングモータパルス出力のチャンネル 1ch にステッピングモータ駆動用パルス出力します。

STRUPR$(A$)

機能

文字列 A$をすべて大文字に変換して返します。

例

```
B$ = STRUPR$(A$)
```

コメント

A$が abcd ならば B$に ABCD が代入されます。

STRLWR$(A$)

機能

文字列 A$をすべて小文字に変換して返します。

例

B$ = STRLWR$(A$)

コメント

A$が ABCD ならば B$に abcd が代入されます。

TESTBIT(v, n)

機能

I/O ワード変数 v のビット n のロジック実行状態を返します。
そのビットのロジック実行状態が ON なら 1 を返し、OFF なら 0 を返します。

v　　I/O ワード番号 (Input[n], Output[n]等)
n　　ビット番号 (0～15)

例

```
IF TESTBIT(Input[2], 3) = 1 THEN
    A$ = “ON”
ELSE
    A$ = “OFF”
ENDIF
```

コメント

I/O 割付番号 Input20 が ON ならば A$に文字列 ON を代入し、ON でなければ A$に文字列 OFF を代入します。

WHILE expression ... ENDWHILE

機能

条件式 expression が真の間、前判定反復のループを繰り返します。
WHILE の expression を評価し、その結果が真であれば以後の処理を行い、偽であれば処理を行わずループから抜けます。

書式

```
WHILE expression
    ...
ENDWHILE
```

例

```
WHILE S = 1
    IF Input[1] & &H0002 : S = 0 : ENDIF
ENDWHILE
```

コメント

I/O 割付番号 Input2 が ON で S =0 になるまでループを繰り返します。

WRITEMODBUS ch,DeviceID,address {M+モデルのみ}

機能

自動的に MODBUS ASCII プロトコルを使っている MODBUS ASCII デバイスに 16 ビットのデータを書き込みます。コミュニケーションのボーレートは、SETBAUD コマンドにより変えられなかったならばその COM ポートのデフォルトのボーレートとなります。

Ch	PLC の COM ポート数（1-8）
DeviceID	MODBUS デバイス（1～255）のデバイス ID
Address	MODBUS デバイスでの保存されたゼロオフセットによるアドレス
Data	MODBUS デバイスに書き込まれる 16 ビットのデータ

例

```
WRITEMODBUS 3, 8, 1000, 1234
```

コメント

データ 1234 は、保存されているオフセットアドレス 1000 において ID＝08 で MODBUS デバイスに書き込みます。（MODBUS の定義ではこれはレジスタ＃41001 を保存します。）

関連

READMODBUS( ),STATUS(2),NETCMD$( )

VAL(A$)

機能

文字列 A$を 10 進数値に変換して返します。

例

B = VAL(“123”) * 100

コメント

B には数値 12300 が代入されます。
