はじめに

このたびは、LT統合開発ソフトウェアLT Editor Ver.2.0をご採用いただき、まことにありがとうございます。
この製品を正しくご使用いただくために、マニュアル類をよくお読みください。
また、マニュアル類は必ずご利用になる場所のお手元に保管し、いつでもご覧いただけるようにしておいてください。

おことわり

(1) 「LT Editor Ver.2.0」(以下本製品といいます)のプログラムおよびマニュアル類は、すべて（株）デジタルの著作物であり、（株）デジタルがユーザーに対し「ソフトウェア使用許諾条件」に記載の使用権を許諾したものです。当該「ソフトウェア許諾使用条件」に反する行為は、日本国内外の法令により禁止されています。

(2) 本書の内容については万全を期して作成しておりますが、万一お気づきの点がありましたら、（株）デジタル「サポートダイヤル」までご連絡ください。

(3) 前項にかかわらず、本製品を運用した結果の影響および第三者のいかなる請求にも、（株）デジタルは一切責任を負いません。

(4) 製品の改良のため、本書の記述と本製品のソフトウェアとの間に異なった部分が生じることがあります。最新の説明は、別冊ないし電子的な情報として提供していますので、あわせてご参照ください。

(5) 本書は、（株）デジタルから日本国内仕様として発売された製品専用です。

(6) 本製品が記録・表示する情報の中に、（株）デジタルまたは第三者が権利を有する無体財産権、知的所有権に関わる内容を含むことがありますが、これは（株）デジタルがこれらの権利の利用について、ユーザーまたはその他の第三者に、何らの保証や許諾を与えるものではありません。また本製品に記録・表示された情報を使用したことにより第3者の知的所有権などの権利に関わる問題が生じた場合、（株）デジタルはその責を負いませんのであらかじめご了承ください。

(株)デジタル 2002 November

本書はLogiTouchをLTと称しています。
商標・商号の権利については「商標権などについて」をご覧ください。

商標権などについて

本書に記載の会社名、商品名は、各社の商号、商標(登録商標を含む)またはサービスマークです。本製品の表示・記述の中では、これら権利に関する個別の表示は省略しております。

商標等	権利者
Microsoft, MS, MS-DOS, Windows, Windows 95, Windows 98, Windows Me, Windows NT, Windows 2000, Windows XP, Windows エクスプローラ, Microsoft Excel 95	米国Microsoft社
Intel, Pentium	米国Intel社
Pro-face, Flex Network	(株)デジタル
NEC, PC-9800	日本電気(株)
Ethernet	米国Western Digital社
IBM, VGA, PC/AT	米国IBM社

なお、上記商号･商標類で、本書での表記が正式な表記と異なるものは以下の通りです。

本書での表記	正式な表記
Windows 95	Microsoft® Windows®95 オペレーティングシステム
Windows 98	Microsoft® Windows®98 オペレーティングシステム
Windows Me	Microsoft® Windows® Me オペレーティングシステム
Windows NT	Microsoft® Windows NT® オペレーティングシステム
Windows 2000	Microsoft® Windows® 2000 オペレーティングシステム
Windows XP	Microsoft® Windows® XP オペレーティングシステム
MS-DOS	Microsoft® MS-DOS® オペレーティングシステム

表記のルール

本書は、以下のルールで表記します。

■安全に関する注意表記

本製品のご使用上、安全に関して重要な説明には、以下の表示を添えています。

表示	意味内容
 警 告	この表示を無視して誤った取り扱いをすると、人が死亡または重傷を負う可能性が想定される内容を示します。
 注 意	この表示を無視して誤った取り扱いをすると、人が傷害を負ったり、物的損害の発生が想定される内容を示します。
重要	この表示の説明に従わない場合、機器の異常動作やデータの消失などの不都合が起こる可能性があります。
強制	必ず実施していただきたい操作、作業などを表します。
禁止	決して行ってはならない操作、作業などを表します。

■説明のための表記

本書では、説明の便宜のため、以下のように表記します。

表記	意味内容
	参考になることがら、補足的な説明です。
参照→	関連する説明が掲載されている項目（マニュアル名、章・節・項）を示します。
Esc Ctrl	パソコンのキーを表します。 **参照→■キーボード対応表**
接続機器	シリアル通信で接続された温調器やインバータなどの周辺機器を指します。ただし、Flex Network、DIO で接続する機器を除きます。
LT	（株）デジタル製グラフィック・ロジック・コントローラ「LT シリーズ」の総称です。
LT Editor	（株）デジタル製 LT 統合開発ソフトウェア LT Editor Ver.2.0（本製品）を指します。

■キーボード対応表

本書では、パソコンのキーを以下のように表記します。
機種によってやや異なりますが、この対応で読み替えてください。

表記＼機種	PC/AT互換		PC-9800シリーズ機
	日本語 106キーボード	英語 101キーボード	
Esc	Esc	Esc	ESC
Tab	Tab ⇤⇥	Tab ⇤⇥	TAB
Ctrl	Ctrl	Ctrl	CTRL
Shift	↑Shift	↑Shift	SHIFT
Alt	Alt	Alt	GRPH
Delete	Delete	Delete	DEL
Back space	Back space	Back space	BS
日本語入力※	Alt + 半角/全角	Alt + ~	CTRL + XFER

※ 日本語入力のオン / オフ操作は、使用する日本語 FEP によって異なります。

LT シリーズ

LT Editor で対応している LT の機種を以下に示します。

シリーズ名	タイプ	型式
LTシリーズ	Type A1	GLC150-BG41-XY32SK-24V
	Type A2	GLC150-BG41-XY32SC-24V
	Type B	GLC150-BG41-FLEX-24V
	Type B+	GLC150-BG41-XY32KF-24V
	Type C	GLC150-BG41-RSFL-24V
	Type H1	GLC150-BG41-ADK-24V
		GLC150-BG41-ADPK-24V
		GLC150-BG41-ADTK-24V
	Type H2	GLC150-BG41-ADC-24V
		GLC150-BG41-ADPC-24V
		GLC150-BG41-ADTC-24V

・ LT Editorで対応している接続機器の機種については、「機器接続マニュアル」を参照してください。
参照→マニュアルの読み方

マニュアルの読み方

■マニュアルの構成

本書はLT Editorの使用方法を説明するマニュアルの1冊、「オペレーションマニュアル ロジックプログラム編」です。本書以外に、3冊のマニュアルとオンラインヘルプがあります。まず、「オペレーションマニュアル 作画編 第1章 LT Editorの基本事項」をお読みになり、LT Editorの概要をご理解ください。**参照→「オペレーションマニュアル 作画編 1.6 マニュアルとヘルプ」**

これらマニュアル類のほか、データファイルとして補足説明や機能の追加・修正情報が添付されていることがあります。

[スタート]ボタンをクリックし、[プログラム(P)]→[Pro-face]→[LT Editor]の順にポイントし、[お読みください]をクリックし、表示された内容をご覧ください。

なお、LT本体に関する詳しい説明は、「LTシリーズユーザーズマニュアル」(別売)をご覧ください。

CD-ROM内に収録	オペレーションマニュアル 作画編	LT Editorを使うための操作手順と、ロジックプログラム開発を除くすべての機能について説明します。PDFデータで収録されています。
	オペレーションマニュアル ロジックプログラム編 (本書)	ロジックプログラムの開発について説明します。基本的な開発手順を説明する「導入編」・演習を通して操作方法を説明する「ロジックプログラム編」・LTとのソフトウェア的な設定を説明する「機能編」で構成されます。PDFデータで収録されています。
	パーツリスト	LT Editorにあらかじめ用意されている部品と図記号について説明します。PDFデータで収録されています。
	機器接続マニュアル	LTと各社の接続機器の接続方法について説明します。PDFデータで収録されています。
LT Editor上で起動	オンラインヘルプ	LT Editorの各ウィンドウやダイアログボックスの設定、ロジックプログラムの命令や機能、各ドライバの設定について説明しています。

- マニュアルに記載のアドレス設定は、あくまでも説明上のものです。実際にはそれぞれの使用環境に応じてアドレスを設定してください。**参照→機器接続マニュアル**
- わかりにくいところなどは「サポートダイヤル」までお問い合わせください。「サポートダイヤル」では、LTシリーズについての技術的なご質問・ご相談にお答えします。**参照→「オペレーションマニュアル 作画編 付.4 ソフトウェアトラブルリポート」**

  なお、パソコンやWindowsそのものに関することは、パソコンをお買い上げの販売店、メーカーにお問い合わせください。

■本書の構成

本書は「導入編」「プログラミング編」「機能編」で構成されます。
それぞれの章の主な内容は以下の通りです。

【導入編】

導入編は例題を用いてLT Editorでの基本的な開発作業を説明します。操作方法や変数・命令の詳しい説明はプログラミング編または機能編を参照してください。

【プログラミング編】

プログラミング編は、ロジックプログラムエディタの一連の操作を習得するためのチュートリアルになっています。実際に操作しながらロジックプログラムを完成させてください。

◆第1章 プログラムの作成

チュートリアルに従ってロジックプログラムを作成します。

◆第2章 ロジックプログラムを実行する

完成したロジックプログラムをLTに転送して実行します。

◆第3章 モニタリングモードでの動作確認

LTのコントローラで実行しているロジックプログラムに、モニタリングモードで動作を確認します。

◆第4章 エラーと警告

ロジックプログラムエディタでエラーチェックを行った際に表示されるエラーについて説明しています。

◆第5章　用語集

ロジックプログラムエディタで使用される主な用語について説明しています。

【機能編】

機能編では、LT本体での動作、ロジックプログラムで使用される命令や変数の一覧などについて説明しています。

◆第6章 機能

LTのコントローラの動作について説明しています。

◆第7章 変数

ロジックプログラムで使用する変数の定義と使用方法について説明しています。

◆第8章 システム変数

コントローラであらかじめ定義されているシステム変数の一覧です。

◆第 9 章 命令

ロジックプログラムエディタでサポートしている命令の一覧です。

◆第 10 章 LS エリアリフレッシュ

コントロール機能と表示機能や外部通信機器とのデータ共有に必要なLSエリアの利用方法について説明します。

◆第 11 章 I/O ドライバ

各I/Oドライバごとの説明です。自己診断、トラブルシューティングなどについて説明しています。

◆第 12 章 エラーと異常処理

LT Editorに関するエラーメッセージについて説明しています。

・ その他、ロジックプログラムエディタに関する詳しい説明は、
オンラインヘルプにあります。

使用上の注意

■ディスクの取り扱いについて

ディスクの破損･故障を防ぐため、以下の点にご注意ください。

強制
- **パソコン本体の電源の ON/OFF は、ディスクを抜いてから行ってください。**

禁止
- **ディスクドライブのランプが点灯しているときは、CD-ROM を取り出さないでください。**
- **CD-ROM の記録面に手を触れないでください。**
- **極端な高温や低温、湿気やホコリの多い場所にディスクを置かないでください。**

■本製品の使用について

誤動作や事故の原因となりますので、以下の点にご注意ください。

警 告

- **タッチパネルスイッチは非常停止用スイッチとして使えません。産業用ロボットほか、厚生労働大臣が指定する産業用機械設備の非常停止用スイッチとしては、必ず人間が直接操作するスイッチを設置することが関係法令で義務づけられています。また、これ以外の装置設備でも、安全確保のため、必ず同様のスイッチを設置してください。**

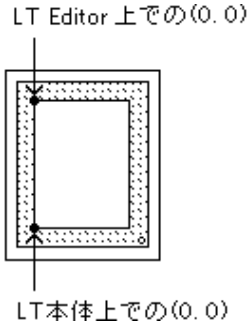
禁止
- **プログラム使用中に、パソコン本体の電源を OFF しないでください。**
- **テキストエディタなどを使用して、本製品のプロジェクトファイルの中身を変更しないでください。**

■制限事項について

本製品には、以下のような制限があります。

◆作画について

- LT Editor上の表示およびマニュアルでの表記は、LT 本体のLCDの設定が「リバース」の場合のカラー表示となっています。LCDの設定が「ノーマル」の場合はLT Editor上のカラーとLT画面上のカラーは白黒逆に表示されますのでご注意ください。**参照→「LTシリーズユーザーズマニュアル」**
- LT Editorはご使用のパソコン内部の文字フォントやグラフィック機能を使用して表示します。このため、これらの表示はLTへ転送後、LT上での表示とパソコン上での表示に多少の相違が生じる場合があります。あらかじめご了承ください。
- タイリングパターン上に文字を配置する場合、文字のドットパターンがタイリングパターンに重なり、文字も同色で配置されるとLT上で正しく表示されない場合があります。この場合、文字を1ドットずらして配置し直してください。
- LTを縦型にして使用する場合は、LT本体とLT Editorでは座標系が異なります。Dスクリプト等で座標を直接指定する場合はご注意ください。



- 日本語版作画環境で作画した画像データの全角文字は、英語版作画環境上では文字化けを起こし正しく表示できません。英語版作画環境での使用を想定する場合は半角英数文字のみ使用して作画してください。

◆機能設定について

- 機能および設定項目によっては、LT本体側でのみ対応しているものとLT Editor側でのみ対応しているものとがあります。

【LT本体側でのみ対応（LT Editor側では未対応）のもの】

- フォント設定
- 日付/時間の設定
- 自己診断機能

【LT Editor側でのみ対応（LT本体側では未対応）のもの】

以下の項目は**[システムの設定]**に含まれます。

- 「チェックサム検証」の設定
- 「階層画面切換」の設定
- スタンバイモード時間による画面切換の設定
- オフラインモード移行への移行方法の設定
- 設定値表示器の1スキャンタイムあたりの処理回数の設定
- LT内部メモリ（LSエリア）のバックアップの設定
- 「エラー表示の消去動作」の設定
- 「ウォッチドッグ」の設定（LTと接続機器の通信状態の監視）
- 通信監視時間設定(送信ウエイト数の設定)
- 「プリントタイプ」の設定（ツールコネクタを使用した印字機能の設定）

◆ロジックプログラムについて

- **変数は 32bit デバイスの Low/High 順で扱われます。**
- **部品による表示機能では実数を扱うことはできません。**
- **パソコンとLTとの実数の精度の違いによって、モニタリングで表示する値と入力した値が一致しない場合があります。**
- **LTのロジックタイムが長い場合、折れ線グラフのサンプリングタイムが守れないときがあります。**
- **メモリリンク方式を使用の場合、変数は、折れ線グラフの一括表示ができません。**
- **保持型変数のデータを保持するSRAMはリチウム電池でバックアップされていますが、バックアップ期間は初期状態(満充電)で約60日、電池寿命時で約6日となりますので、この期間以上にバックアップが必要な場合は、ホストでバックアップしたり、LT Editor でバックアップするシステム構成をとってください。**
- **ロジックプログラムと画面データで同じLSエリアを共有する場合は、全てLogicシンボルの LS 変数（LS<*>）で指定してください。**

お問い合わせ

LT Editor に関するご質問は「サポートダイヤル」までお問い合わせください。

■ **月曜日～金曜日　9:00～17:00**

東　京　　TEL (03)5821-1105
名古屋　　TEL (052)932-4093
大　阪　　TEL (06)6613-3115

■ **月曜日～金曜日　17:00～19:00**

専用ダイヤル　　TEL (06)6613-3206

■ **土曜日・日曜日・祝日（12月31日～1月3日を除く）　9:00～17:00**

専用ダイヤル　　TEL (06)6613-3206

ホームページからのお問い合わせには随時承ります。

URL　http://www.proface.co.jp/

以前のバージョンをご使用のお客様へ

Ver.1.04以前のロジックプログラムエディタをご使用のお客様は下記の注意事項をご覧ください。

Ver.2.0でセットアップを行ったLTに対して、Ver.1.04以前のロジックプログラムエディタではロジックプログラムのダウンロードおよびモニタリングは行えません。

目次

第 2 章　ロジックプログラムを実行する

第3章　モニタリングモードでの動作確認

第4章　エラーと警告

第5章　用語集

機能編

第6章　コントローラ 機能

第7章　変 数

第8章　システム変数

第9章　命 令

第 10 章　LS エリアリフレッシュ

第 11 章　I/O ドライバ

第 12 章　エラーと異常処理

付録

索引

導入ガイド
（入門レッスン）

本章では、LT Editorを使って、LTのアプリケーションを開発する基本的な手順を説明します。ロジックプログラムエディタや画面エディタについての説明はオペレーションマニュアル「ロジックプログラム編」や「作画編」、オンラインヘルプを参照してください。

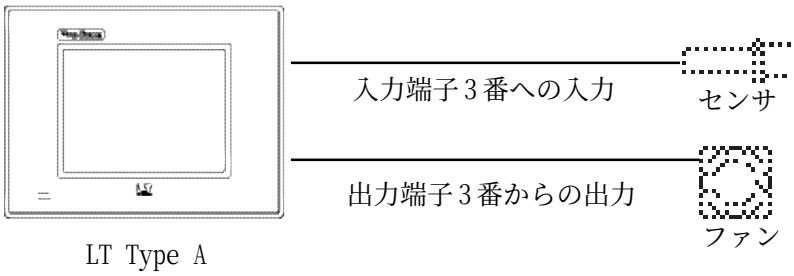
レッスンを始める前に

本章のレッスンでは、LTでアプリケーションを開発するための手順や基本的な機能・操作について、演習を通じて説明します。はじめてLTでアプリケーション開発される方は是非演習を行ってください。
ここでは、以下の機器を使ってサンプルアプリケーションを作成します。
LTの開発作業を行うにはLT Editorがインストールされている必要があります。

■使用環境

本体	LT Type A
I/Oユニット	Type A 内蔵（入力16点出力16点）
ケーブル	画面転送ケーブル
ファン	DC24Vファン
センサ	DC24V近接スイッチ

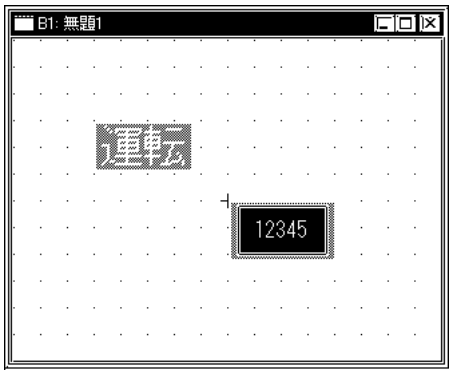
■構成図



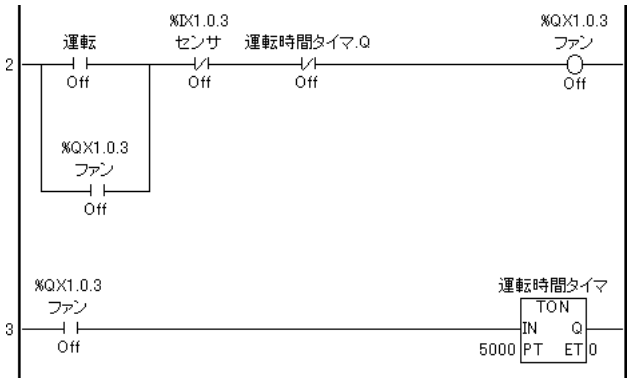
■完成アプリケーション

本演習では次のような画面およびロジックプログラムを作成します。

作成する画面



作成するロジックプログラム



■動作仕様

- LT画面上の「運転」タッチスイッチをタッチしたときにファンが回り、5秒後に停止する
- LT画面上の設定値表示器をタッチして表示されるキーボードで、ファンの停止時間を変更できる
- ファンの回転中、センサが反応するとファンが停止する

■開発作業の流れ

LT Editorでのアプリケーション開発作業は､次のような流れになります｡この流れにしたがってレッスンを進めます｡(LT Editorはインストールされているものとします｡)

1.LT Editorの起動

LT Editorを起動します。
使用機種、接続機器を選択します。

2.外部I/Oへの変数の割り付けとI/O使用可設定

ロジックプログラムエディタのI/Oコンフィギュレーションで、I/O入出力の端子番号とロジックプログラムで使う変数名（デバイスアドレス）を対応させます。

3.内部変数の作成

内部リレーやレジスタ、タイマ、カウンタなどを変数として作成します。

4.ロジックプログラムの作成

ロジックプログラムエディタでロジックプログラムを組みます。

5.画面の作成

画面エディタで画面を作成します。

6.画面・ロジックプログラムの転送と動作確認

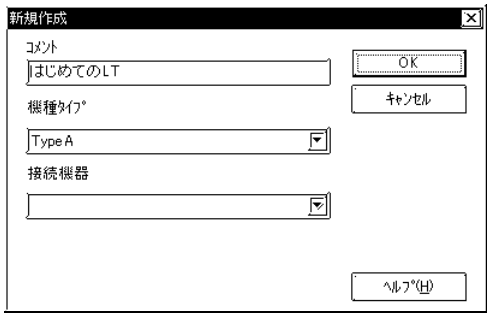
画面とロジックプログラムを転送し、設計どおりの動作ができることを確認します。

7.運転

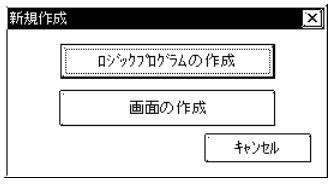
1.LT Editor の起動

1. Windows の［スタート］ボタンをクリックし、［プログラム］、［LT］の順にポイントし、［プロジェクトマネージャ］をクリックします。

2. ［新規］アイコンをクリックし、［新規作成］ダイアログボックスを表示します。
次のように設定し、［OK］ボタンをクリックします。
・［コメント］　：はじめての LT
・［機種タイプ］　：LT Type A
・［接続機器］　：なし（Type C の場合のみ選択します。）



3. ［新規作成］ダイアログボックスで初期設定が行われた後、エディタ起動用のダイアログボックスが表示されます。本レッスンでは［ロジックプログラムの作成］ボタンをクリックし、ロジックプログラムエディタを起動します。

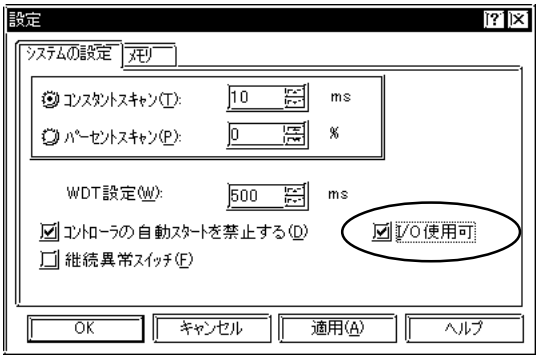


2. 外部 I/O への変数の割り付けと I/O 使用可設定

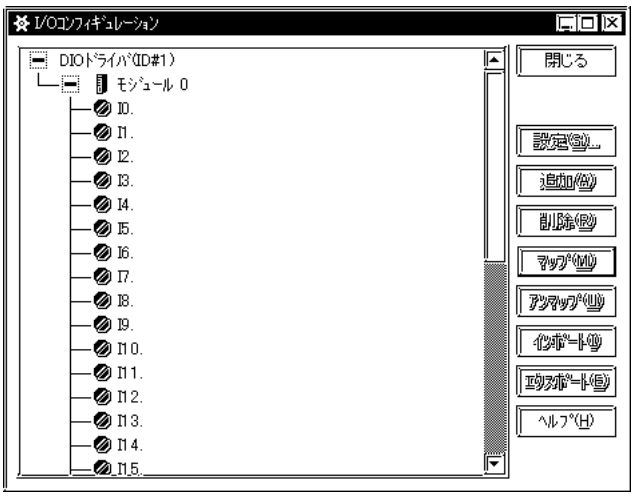
一般のPLCでは、外部I/Oアドレスを入出力デバイスアドレスという形で各メーカー独自の表現をしています。
LT Editorではこのようなデバイスアドレスの代わりにI/Oアドレスに任意の名前を割り付けて使用します。これを**変数**と呼びます。
変数は変数タイプなどのパラメータによって内部リレーやタイマなどに使い分けることができます。作成できる変数の数はメモリ変数エリアの容量に依存し、種類別の使用制限はありません。**参照→**7.2 変数タイプ

外部I/Oを使用するにはロジックプログラムエディタの**I/Oコンフィギュレーション**で変数名を外部 I/O に割り付ける必要があります。

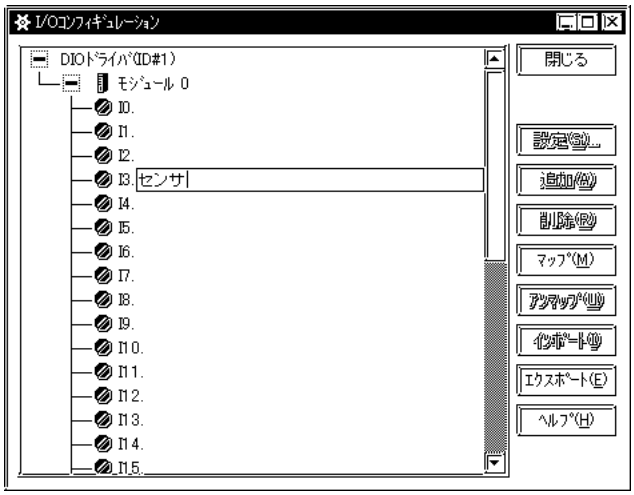
1. 外部 I/O を使用可にするための設定を行います。
   ロジックプログラムエディタの［コントローラ］メニューから［設定］を選択し、［設定］ダイアログボックスを表示します。
   ［システムの設定］タブで［I/O 使用可］を指定し、［OK］ボタンをクリックします。
   ［I/O 使用可］にチェックがない場合、外部からの入出力が行えません。（入出力を切り離した状態でのデバッグなどに利用します。）



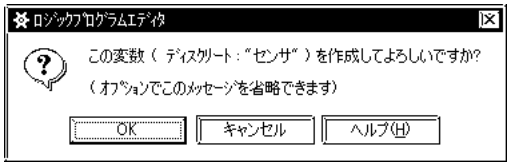
2. 外部 I/O に「センサ」、「ファン」という変数名を割り付けます。
   ［データ］メニューから［I/O コンフィギュレーション］をクリックし、［I/O コンフィギュレーション］ウィンドウを表示します。



3. DIO ドライバのモジュール 0 に I0 から I15 と Q0 から Q15 が表示されます。**I は入力、Q は出力の外部 I/O** を指します。
   今回は I3 をダブルクリックして「センサ」と入力し、[ENTER] キーを押します。

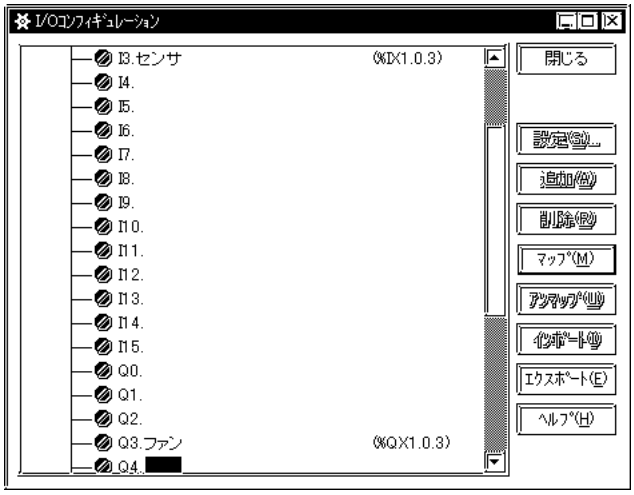


4. 表示されたダイアログボックスで [OK] ボタンをクリックします。名前を付けることで I3 に該当する変数が作成され、実際の入力端子に割り付けられたことになります。

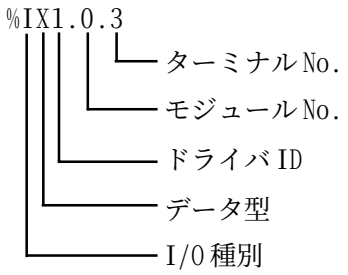


「ディスクリート」はビット単位で扱う変数であることを指定する変数タイプです。
参照→7.2 変数タイプ

5. 同様に Q3 に「ファン」と名前を付けます。
   I/O コンフィギュレーションで外部 I/O 端子に割り付けた変数名はロジックプログラム・画面ソフトから外部機器へのアクセスに使用されます。
   本レッスンの場合、「センサ」をロジックプログラムの中で a 接点や b 接点命令に割り付けることで外部入力端子から入力を得られます。同様に「ファン」を OUT 命令に割り付けることで外部出力端子へ出力できます。



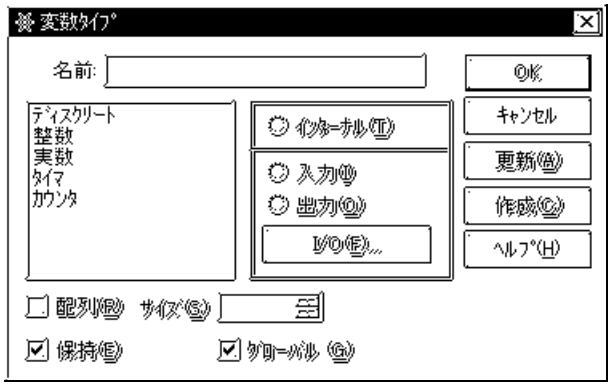
I/0コンフィギュレーションウィンドウに表示される「%IX1.0.3」などはI/0アドレスです。I/0アドレスの内容は次のようになります。



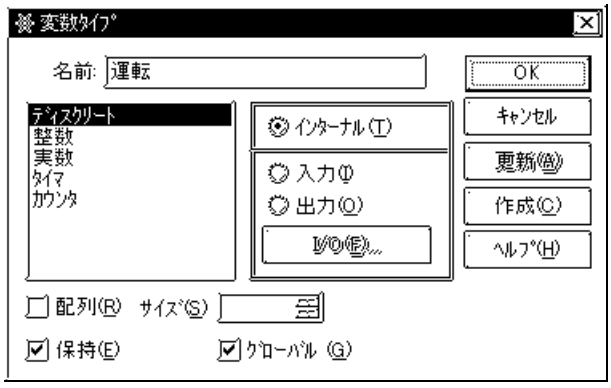
3．内部変数の作成

内部リレー・レジスタ・タイマ・カウンタなどにも外部I/0と同様に任意の名前を付けた変数を作成して使用します。

1．「運転」という内部リレーに相当する変数を作成します。
ロジックプログラムエディタの［データ］メニューから［変数タイプ］をクリックし、［変数タイプ］ダイアログボックスを表示します。



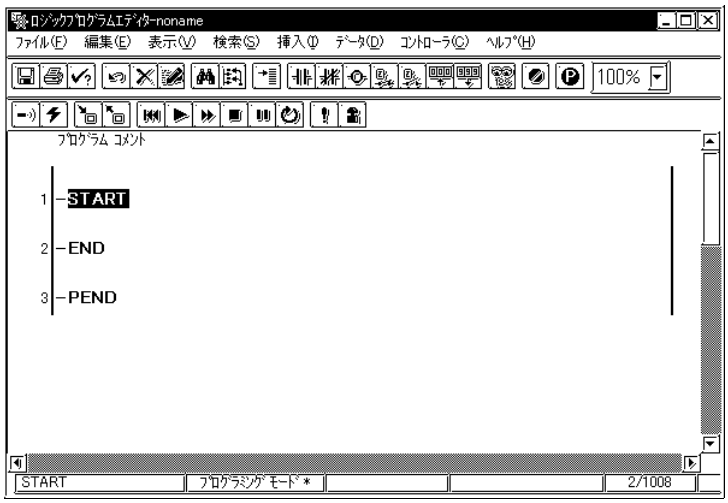
2．名前欄に「運転」と入力し、変数タイプの中からビット単位で扱う変数であることを指定する「ディスクリート」を選択します。
作成する変数が内部変数であることを指定する「インターナル」を選択し、［OK］ボタンをクリックします。
この操作で「運転」という名前の内部リレーが作成されます。



4．ロジックプログラムの作成

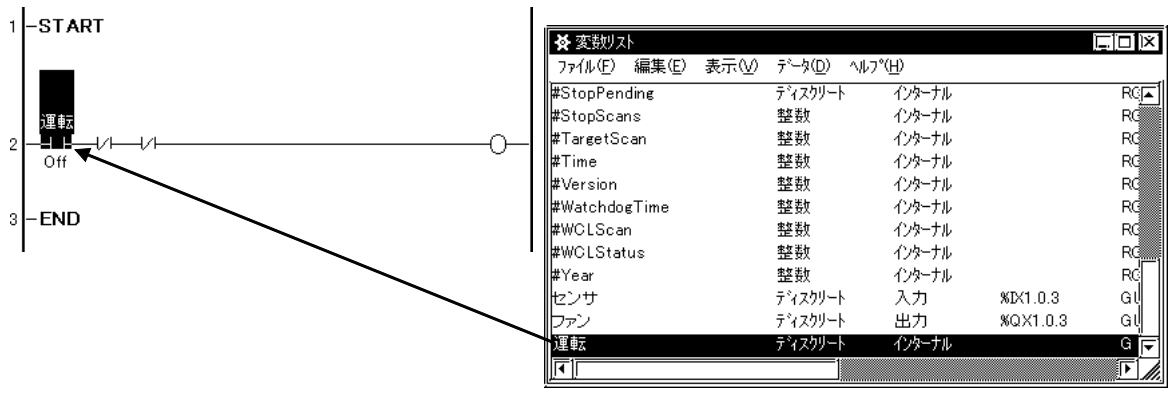
ラング（命令と命令をつなぐ線）に命令を挿入することでロジックプログラムを作成します。

1．ラング1の［START］をクリックし、ツールバーの をクリックします。
1本目のラングを作成する場合、必ず［START］をクリックします。



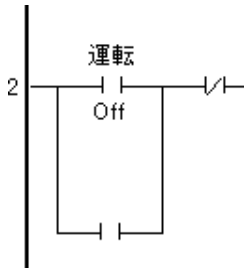
2．ツールバーの をクリックし、ラング2にa接点を作成します。
同様に をクリックし、b接点を2つ作成します。続いて をクリックし、コイルを作成します。

3．［データ］メニューから［変数リスト］をクリックし、［変数リスト］ウィンドウを表示します。変数リストには作成した「運転」が表示されています。
「運転」をa接点にドラッグ＆ドロップします。

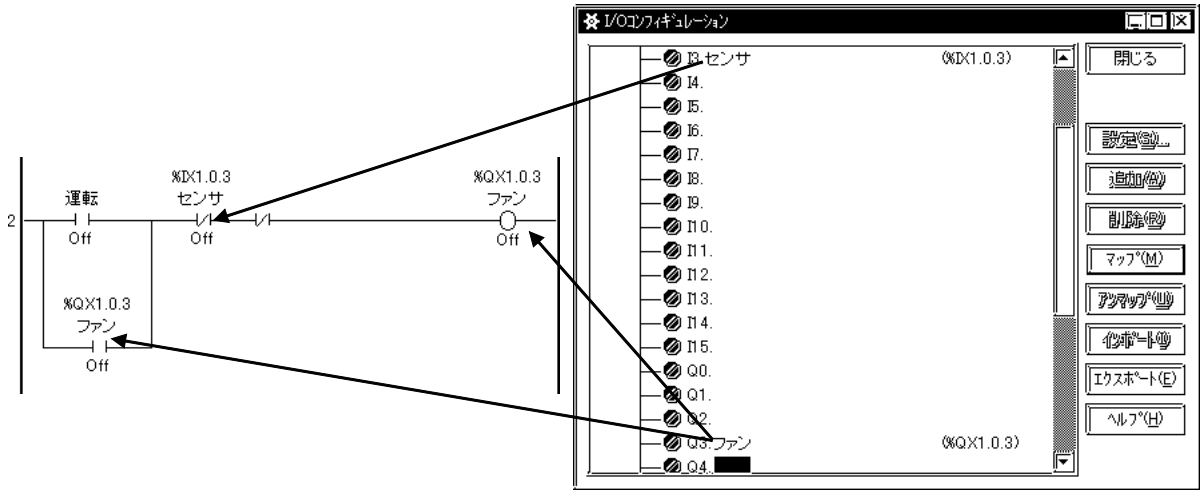


4．自己保持回路を作成するため、a接点「運転」の左側の接続線をドラッグし、a接点「運転」の右側にドロップします。

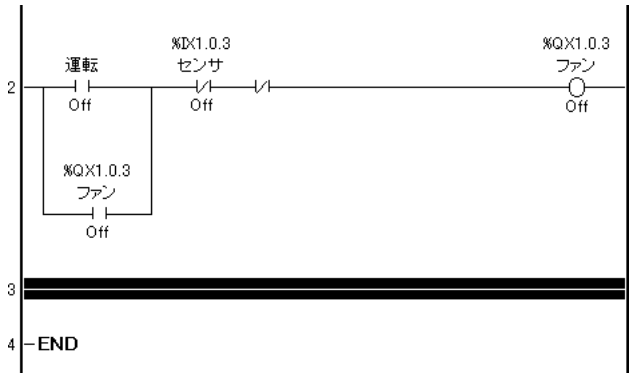
5．できたOR回路の分岐の下側を選択した状態でをクリックし、a接点を作成します。



6．I/Oコンフィギュレーションから変数を割り付けます。「センサ」を1つ目のb接点、「ファン」をOR回路のa接点とコイルにドラッグ＆ドロップします。



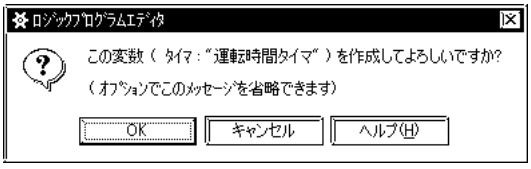
7. ラング2を選択した状態でツールバーの[]をクリックします。
新しいラングは選択したラングの下に挿入されます。



8. ツールバーの[]をクリックし、ラング3にa接点を作成します。

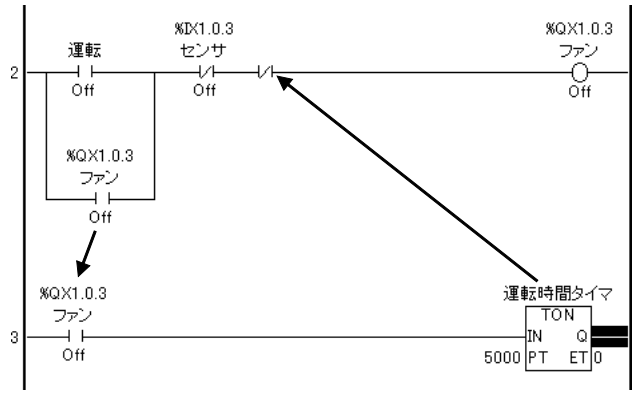
9. []をクリックしてオンディレータイマ（TON）命令を作成します。「運転時間タイマ」と入力し、［ENTER］キーを押します。

10. 表示されたダイアログボックスで［OK］ボタンをクリックし、変数名「運転時間タイマ」を作成します。オンディレータイマに割り付けた名前（変数）は変数タイプ「タイマ」に設定されます。
接点やコイルなどに割り付ける変数も同様に作成できます。



11. オンディレータイマの左下、PTの左にある数字「0」をダブルクリックし、［データ値変更］ダイアログボックスを表示します。
運転時間を5秒にするため、値欄に「5000」（単位：ミリ秒）と入力し、［OK］ボタンをクリックします。

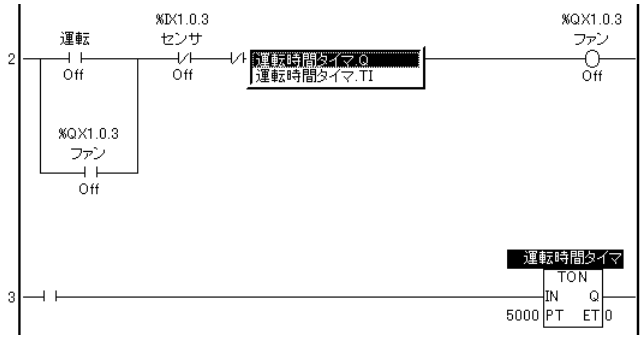
12. ラング2からa接点の変数名「ファン」をラング3のa接点にドラッグ＆ドロップします。同様にラング3のオンディレータイマの変数名「運転時間タイマ」をラング2の2つ目のb接点にドラッグ＆ドロップします。



13.「運転時間タイマ」をドラッグ＆ドロップして表示された一覧から、「運転時間タイマ」の出力ビットを表す変数である［運転時間タイマ.Q］をダブルクリックします。

変数「運転時間タイマ.Q」は手順10で運転時間タイマを作成した時に自動的に作られる専用変数で、運転時間タイマがタイムアップしたことを示すビット情報（接点情報）です。

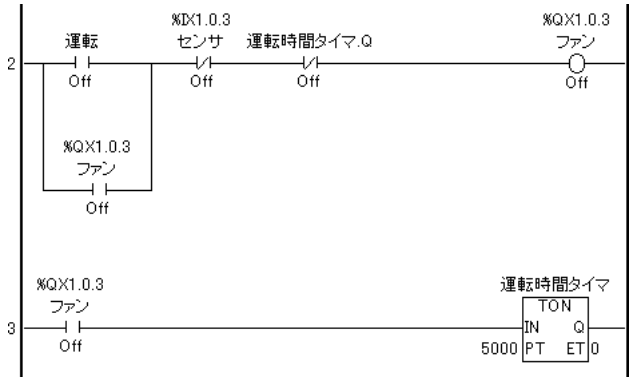
参照→7.2 変数タイプ ■タイマ・カウンタ



14. 以上でロジックプログラムは完成です。

ツールバーの[保存]をクリックして作成したロジックプログラムを保存します。

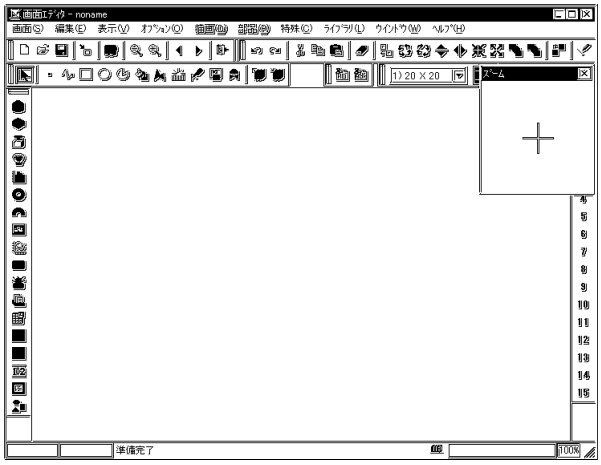
ロジックプログラムを保存することで変数の情報が画面エディタへインポートされます。画面エディタで画面を作成する前に必ずロジックプログラムを保存してください。



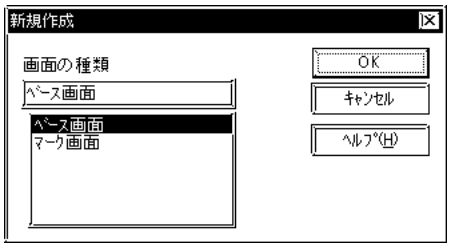
5．画面の作成

LTに表示する画面を作成します。ロジックプログラムエディタは起動したままにしておきます。

1．プロジェクトマネージャで［画面］アイコンをクリックします。
表示された画面エディタの🗋をクリックします。

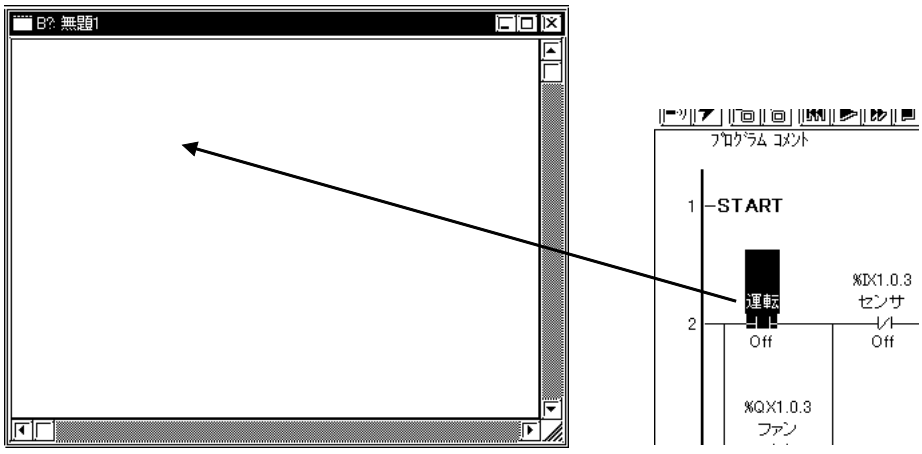


2．［ベース画面］を選択し、［OK］ボタンをクリックします。

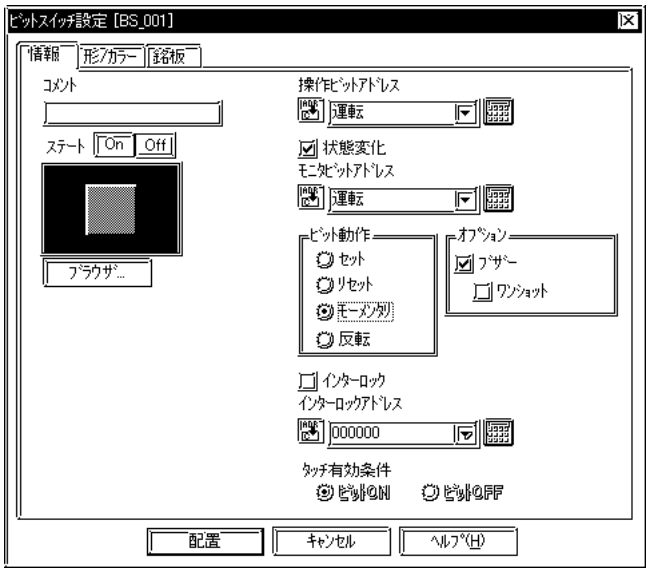


3．ロジックプログラムエディタのラング2にあるa接点「運転」を選択し、ベース画面へドラッグ&ドロップします。
ドラッグ&ドロップする際は**変数名だけでなく命令ごと選択**します。

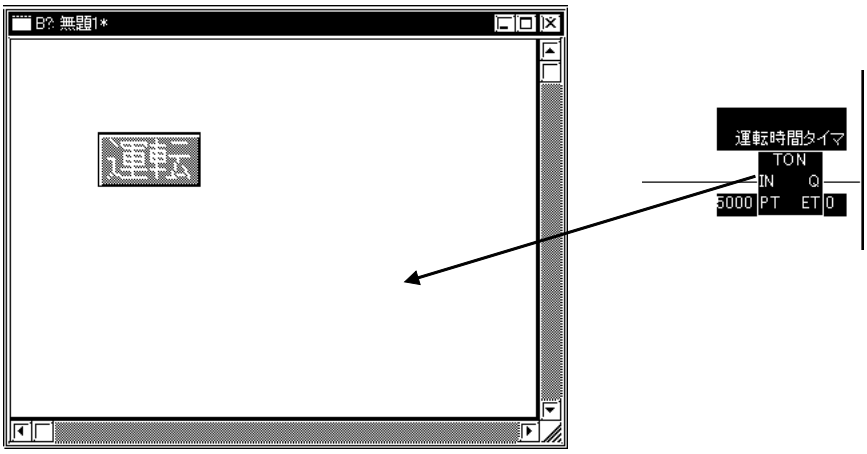
作成したロジックプログラムをロジックプログラムエディタで保存してから命令をドラッグ&ドロップする必要があります。



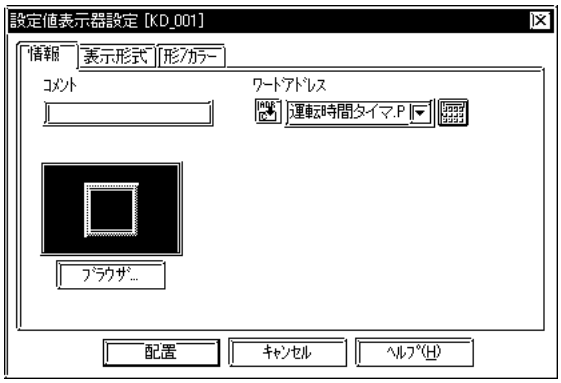
4. 表示された［ビットスイッチ設定］ダイアログボックスで、「ビット動作」からタッチしている間ビットがONとなる「モーメンタリ」を選択します。
   続いて［配置］ボタンをクリックし、画面上に配置します。ここでは銘板（スイッチ上の文字）やスイッチの形状なども設定できます。



5. 同様にオンディレータイマをベース画面上にドラッグ&ドロップします。
   オンディレータイマはベース画面上で設定値表示器として扱われます。



6．表示された［設定値表示器設定］ダイアログボックスで［配置］ボタンをクリックし、画面上に配置します。配置された設定値表示器はLT画面上でタッチするとキーボードが表示され、任意の数値を入力することができます。

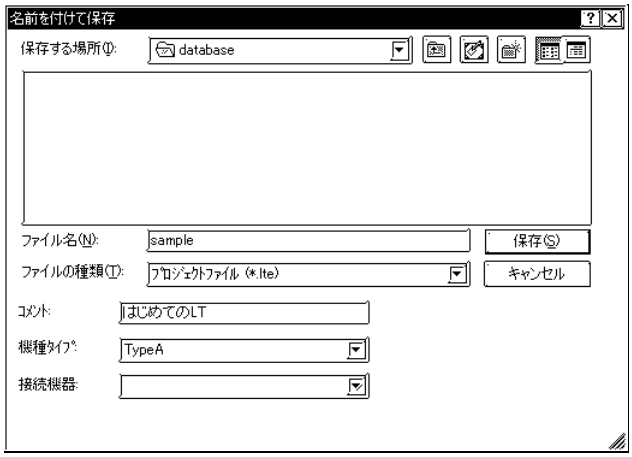


7．ツールバーの  をクリックし、［名前を付けて保存］ダイアログボックスを表示します。画面番号に「1」を入力し、［OK］ボタンをクリックします。
画面番号を１に設定するとLTが起動した際の初期画面となります。

6．画面・ロジックプログラムの転送と動作確認

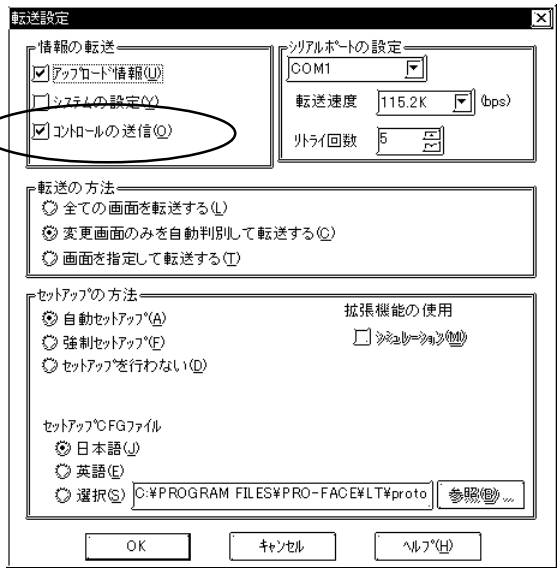
作成したロジックプログラムと画面を転送し、動作確認を行います。
転送を行う前にプロジェクトファイルを保存する必要があります。

1．ロジックプログラムエディタと画面エディタを終了し、プロジェクトマネージャで［プロジェクト］メニューから［名前を付けて保存］を選択します。
ファイル名欄に保存するプロジェクトファイル名を入力し、[保存]ボタンをクリックします。



2．プロジェクトマネージャで［転送］アイコンをクリックします。

3．ツールバーの(転送設定)をクリックし、[転送設定]ダイアログボックスを表示します。「情報の転送」の「コントロールを送信」を指定します。
続いて「シリアルポートの設定」で画面転送ケーブルを接続するポートを選択し、[OK]ボタンをクリックします。



4．シリアルポートとLTを画面転送ケーブルで接続し、ツールバーの(画面とコントロールを送信）をクリックします。

5．転送が完了すると、LTがリセットされて作成した画面が表示されます。
作成した画面とロジックプログラムが正しく動作するか確認します。確認事項は次の3つです。

・ タッチパネル上の「運転」をタッチするとファンが回り、5秒後に停止する
・ タッチパネル上の設定値表示器をタッチするとキーボードが表示され、停止時間が変更できる
・ ファンの回転中、センサが反応するとファンが停止する

以上でアプリケーションを開発する基本的な手順は終了です。
ロジックプログラムエディタや画面エディタの詳細な説明に関しては、それぞれのマニュアルやヘルプを参照してください。

プログラミング編

第1章
プログラムの作成

本章では、ロジックプログラムエディタを使って、プログラミングモードでロジックプログラムを作成する方法を、順序を追って説明します。
ロジックプログラムエディタの起動方法については**「オペレーションマニュアル 作画編 1.2 起動から終了まで」**を参照してください。また、ロジックプログラムエディタの各部の詳細説明は、**機能編、オンラインヘルプ**を参照してください。

レッスンを始める前に

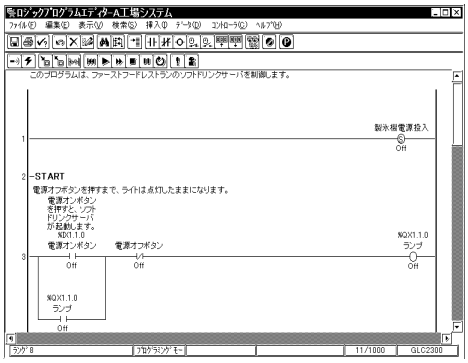
本章の各レッスンは、ロジックプログラムエディタの操作手順について、練習例に沿って説明しています。この様に練習例に沿って演習することをチュートリアルといいます。
ここでは、ロジックプログラムエディタを使ってファーストフードレストランのソフトドリンクマシンの動作を想定して制御するロジックプログラムを作成します。このマシンには、以下の機能があります。

- **ボタンを1度押すことによって、大/中/小のサイズのカップサイズに適した量のドリンクを自動的に吐出する**
- **カップがディスペンサーの下にある場合だけ、氷またはソーダを吐出する**
- **機械に電源を投入したあとに、機械によって装填されたカップの数をカウントする**

◆ロジックプログラムおよび画面の完成例

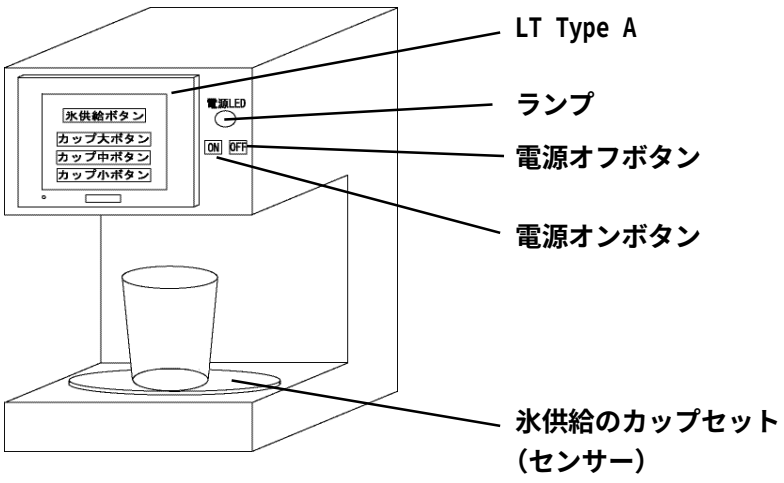
本レッスンのロジックプログラム及び作画画面の完成品は、「C:\Program Files\Pro-face\LT\SAMPLE」のフォルダの中に「Soda.lte」ファイルとして付属しています。作成方法が分からない場合や検索などの練習の場合は、このファイルを開いて参考にしてください。ロジックプログラムエディタの各部の詳細説明は、オンラインヘルプを参照してください。

＜ロジックプログラム＞



＜画面＞

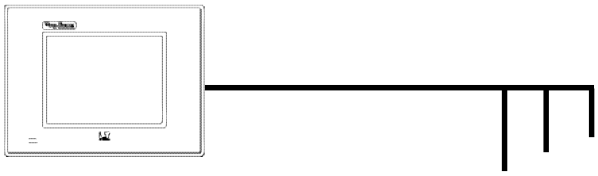
◆ソフトドリンクマシン



◆ハードウエア構成

構成図

LT Type A



電源オンボタン、ランプなど

◆I/Oコンフィギュレーションの割り付け

「氷供給ボタン」、「カップ大ボタン」、「カップ中ボタン」、「カップ小ボタン」は、LTからのタッチパネル入力とするので割り付けません。

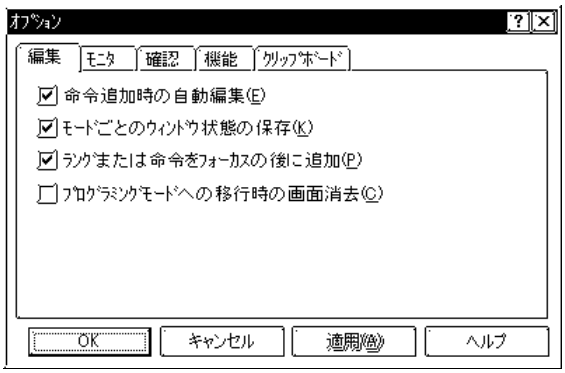
変数名	端子タイプ	端子番号
電源オンボタン	入力	I0
氷供給のカップセット	入力	I2
電源オフボタン	入力	I6
ランプ	出力	Q0
氷供給	出力	Q1
ソーダバルブ	出力	Q2

ロジックプログラムを作成する前にオプション画面で設定を行う

ロジックプログラムの作成を始める前に、好みにあった動作にロジックプログラムエディタの操作設定を合わせることができます。[オプション]ダイアログボックスで、プログラムを作成し実行する方法をカスタマイズできます。

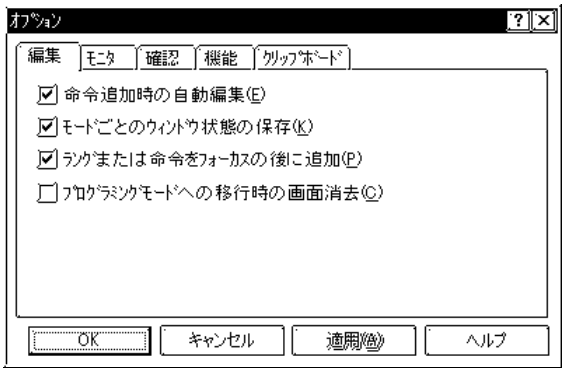
■オプション画面の設定手順

1. [ファイル]メニューから、[オプション]を選択します。[オプション]ダイアログボックスが表示されます。



2. 各項目のチェックボックスをクリックすると、その項目が選択またはクリアされます。[オプション]ダイアログボックスの中の各項目の意味は、次頁の表の通りです。

◆編集タブ

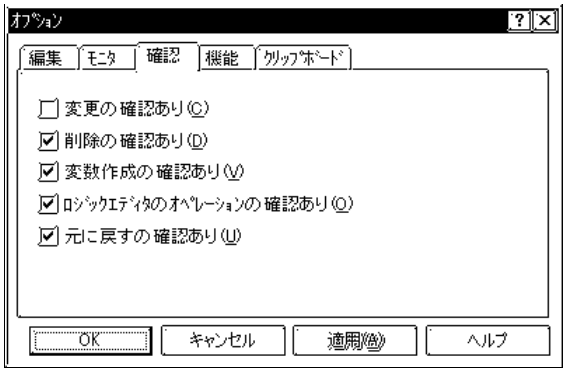


項目	説明
命令追加時の自動編集 (デフォルト=選択)	プログラムの中に挿入された新しい命令について、自動的に[命令パラメータ]ウィンドウが開きます。
モードごとのウィンドウ状態の保存 (デフォルト=選択)	ロジックプログラムエディタの再起動やモード切り替えの際、前回の作業終了時に開いていたウィンドウが、そのときの状態で開かれます(ウィンドウのサイズ、位置など)。この設定は[データ値表示リスト]ウィンドウにも適用されます（[データ値表示リスト]ウィンドウは、現在のプログラムがモニタリングモードの場合、その変数の値を表示します。
ラングまたは命令のフォーカスの後に追加 (デフォルト=選択)	この項目が選択されている場合、新しい命令は選択した命令の右側に挿入されます。また、ラベル、ラング、サブルーチンなどのオブジェクトは、選択しているラングの下に挿入されます。 この項目をクリアした場合、新しい命令は、左側、新しいオブジェクトは上側に挿入されます。命令の設定をされていない横ライン(接続線)を選択している場合は、設定に関わらず、命令はライン上に挿入されます。
プログラミングモードへの移行時の画面消去 (デフォルト=クリア)	モニタリングモードからプログラミングモードに移るときにロジックプログラムエディタをクリアします。

◆モニタタブ

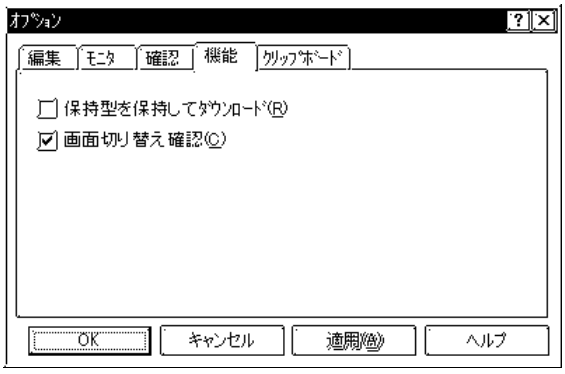
	項目	説明
モニタリングモードの更新	パワーフロー (デフォルト=選択)	コントローラの実行中にパワーフローが表示されます。パワーフローとは、コントローラがRUN中に導通しているラング（ロジックプログラムで命令を記述する横方向のライン）を強調表示します。
	状態フロー (デフォルト=クリア)	コントローラの実行中に状態フローが表示されます。状態フローとは、コントローラがRUN中に導通している命令を強調表示します。パワーフローと状態フローを同時に表示できます。
周期(ミリセカンド) (デフォルト=500msec)		ロジックプログラムエディタがパワーフロー、状態フロー、データ値、およびステータスバーを更新するためにコントローラから新しいデータを要求する頻度を指定します。

◆確認タブ



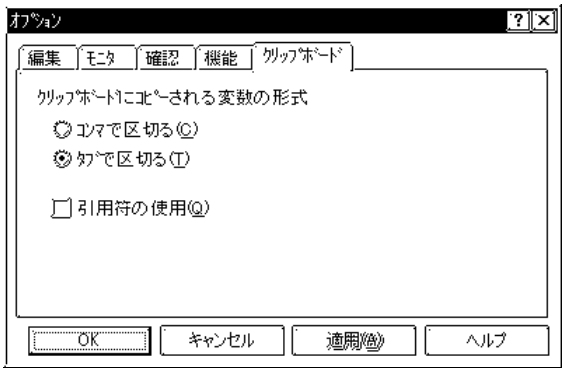
項目	説明
変更の確認あり (デフォルト=クリア)	ロジックプログラムエディタは、[更新]をクリックしたときにだけ変更を受け入れます。この項目をクリアした場合、ロジックプログラムエディタは変更の確認を要求します。
削除の確認あり (デフォルト=選択)	ロジックプログラムエディタは、プログラムの作成中に、すべての削除について確認を要求します。
変数の作成の確認あり (デフォルト=選択)	ロジックプログラムエディタは、プログラムの中での新しい変数の作成について確認を要求します。これは、オフライン環境にだけ適用されます。
ロジックプログラムエディタのオペレーションの確認あり (デフォルト=選択)	ロジックプログラムエディタは、コントローラの動作の変更(スタート／ストップ、読み出し/書き込みなど)について確認するように要求します。
元に戻すの確認あり (デフォルト=選択)	[元に戻す]を実行する前に、確認するかどうかを指定します。

◆機能タブ



項目	説明
保持型変数を保持してダウンロード (デフォルト=無効)	コントローラへの書き込みを行うときに、保持型変数の値を保持することが可能です。 **有効** コントローラへの書き込みを行うときに、保持型変数の値を保持することが可能です。 確認タブの[ロジックエディタのオペレーションの確認あり]を指定していない場合はメッセージダイアログボックスが表示されません。 **無効** コントローラへの書き込みを行うときに、全ての変数の値が初期化されます。
画面切り替え確認 (デフォルト=有効)	ロジックプログラムまたはPLCで、「#Screen」と「LS[8]」、「LS0008」、「PLCに割り付けた切り替え画面番号デバイス」に切り替え画面番号をセットして画面切り替えを行った場合に、画面の切り替え完了の確認をするかどうかを指定します。 **有効** **・ダイレクトアクセス方式の場合** 画面の切り替えを確認した（システムデータエリアの表示中画面番号と切り替え画面番号が一致した）後に「#Screen」と「LS[8]」、「LS0008」、「PLCに割り付けた切り替え画面番号デバイス」に0を書き込みます。 **・メモリリンク方式の場合** 画面の切り替えを確認した（システムデータエリアの表示中画面番号と#Screenが一致した）後に#Screenに0を書き込みます。 **無効** **・ダイレクトアクセス方式の場合** 画面の切り替えを確認しても、「#Screen」と「LS[8]」、「PLCに割り付けた切り替え画面番号デバイス」の値はセットされた切り替え画面番号を保持します。 **・メモリリンク方式の場合** 画面の切り替えを確認しても、「#Screen」の値はセットされた切り替え画面番号を保持します。

◆クリップボードタブ



	項目	この項目が選択された場合の動作
クリップボードにコピーされる変数の形式	コンマで区切る (デフォルト=クリア)	ロジックプログラムエディタの変数リストからクリップボードにコピーされたフィールドがコンマで区切られます。 例) My_variable,Discrete,adescription
	タブで区切る (デフォルト=選択)	ロジックプログラムエディタの変数リストからクリップボードにコピーされたフィールドがタブで区切られます。 例) My_variable TAB Discrete TAB adescription
	引用符の使用 (デフォルト=クリア)	ロジックプログラムエディタの変数リストからクリップボードにコピーされたフィールドが区切り記号で区切られ、引用符で囲まれます。 例) "My_variable","Discrete","adescription"

このチュートリアルでは、デフォルト設定値を使用します。
[キャンセル]をクリックし、[オプション]ダイアログボックスを閉じます。
キャンセルをクリックすることで、デフォルト設定値のままダイアログボックスを閉じることができます。

本演習のあらすじ

1．LT Editor を起動する

参照→「1.1 起動のしかた」

2．新規作成で LT のタイプ / 接続機器を決める

参照→「1.1 起動のしかた」

3．ロジックプログラムの開発

1．変数を決める

ロジックプログラムエディタで作成するロジックプログラムの動作を設定する方法について説明します。また、使用する変数の作成と削除、初期値の設定方法を説明します。

参照→「1.2 変数の作成」

2．ロジックプログラムを作成する

ラングの作成、命令や分岐の挿入方法、ラングおよびラングに関連づけられている命令や分岐の削除方法について説明します。

参照→「1.3 ラング、命令、および分岐の挿入」

3．ロジックプログラムに変数を割り付ける

ロジックプログラムの中の命令に変数を割り付ける方法について説明します。

参照→「1.4 命令への変数の割り付け」

4．コメントを入れる

ロジックプログラムにコメントを付ける方法について説明します。プログラム全体、特定のラング、個別の命令にコメントを付ける方法です。

参照→「1.5 ロジックプログラムのドキュメント化」

5．追加

ラングのコピー、切り取りおよび貼り付け方法を説明します。

参照→「1.6 ラングのコピー、切り取りおよび貼り付け」

6．サブルーチン

ロジックプログラムにサブルーチンやラベルを挿入する方法について説明します。

参照→「1.7 サブルーチンおよびラベル」

7．検索

ロジックプログラムの中で、目的の回路をすばやく検索し移動する方法について説明します。

参照→「1.8 ロジックプログラム内の移動」

8．I/O割り付け

ロジックプログラムの中の論理変数を実I/Oに割り付ける方法について説明します。

参照→「1.9 I/Oの割り付け」

9．エラーチェック

ロジックプログラムのエラーをチェックする方法について説明します。

参照→「1.10 プログラムエラーチェック」

10．印刷

ロジックプログラムを印刷する方法について説明します。

参照→「1.11 ロジックプログラムの印刷」

11．インポートとエクスポート

ロジックプログラムの「読み込み」と「書き出し」ついて説明します。

参照→「1.12 ロジックプログラムのインポート / エクスポート」

4．画面プログラムの開発

ロジックプログラムとリンクした画面を画面エディタで作成します。

参照→「1.13 画面プログラムの開発」

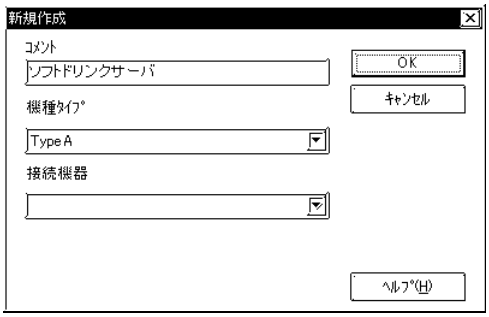
1.1 起動のしかた

ロジックプログラムエディタでロジックプログラムの作成を行うために、まずプロジェクトマネージャを起動します。

1. [スタート]ボタンをクリックし、[プログラム(P)]、[LT]の順にポイントし、[プロジェクトマネージャ]をクリックします。
2. プロジェクトマネージャが起動します。

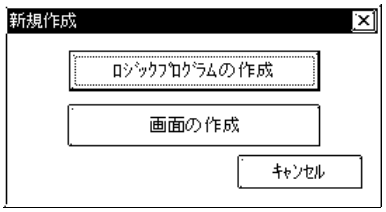
3. プロジェクトマネージャで[プロジェクト(P)]から[新規作成(N)]を選択するか、 

ボタンをクリックします。ここで下記のように設定し、[OK]ボタンを押してください。



コメント　　：ソフトドリンクサーバ
機種タイプ　　：LT Type A
接続機器　　：なし

4. ロジックプログラムまたは画面を作成するかの問い合わせがあります。[ロジックプログラムの作成]をクリックして、ロジックプログラムエディタを起動させます。



1.2　変数の作成

ロジックプログラムエディタの動作の設定方法について説明します。また、その中で使用する変数の作成と削除、初期値の設定方法を説明します。
ここで使用するチュートリアルプログラムの完成品は、C:\Program Files\Pro-face\LT\SAMPLEのフォルダの中にSoda.lteファイルとして用意されています。
参照→「第7章 変数」

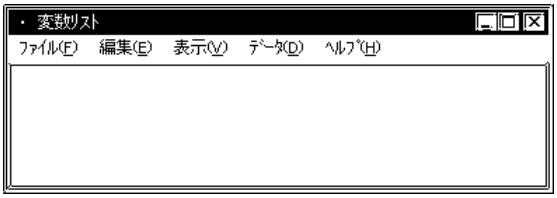
1.2.1　変数リストの作成

変数はロジックプログラムの作成中にいつでも追加できますが、あらかじめ設定することもできます。このチュートリアルで使用する変数のリストをここで作成しておくと便利です。

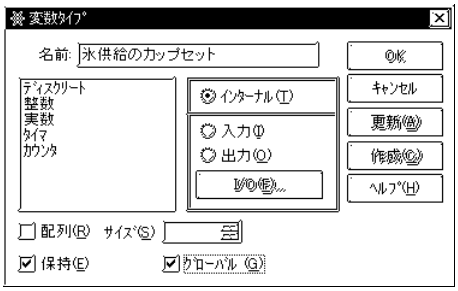
■変数リストの作成方法

メニュー内の詳細な説明は、オンラインヘルプをご参照ください。

1. [データ]メニューから、[変数リスト]を選択します。
   [変数リスト]ウィンドウが表示されます。



2. [編集]メニューから、[変数の追加]を選択します。
   [変数タイプ]ダイアログボックスが表示されます。



3. 名称フィールドに「氷供給のカップセット」と入力します。
   変数名の制限事項については、**参照→「第7章 変数」**

1.2.2　変数タイプの指定

現在、[変数リスト]ウィンドウに変数「氷供給のカップセット」が表示されています。その下のリストの中の[未定義]という語が反転表示されています。これは変数「氷供給のカップセット」に変数タイプが指定されていないことを表しています。ここでは、ディスクリート、入力を指定します。変数タイプの詳細については、「7.2 変数タイプ」を参照してください。

■変数タイプの指定方法

1. [変数タイプ]ダイアログボックスから[ディスクリート]を選択します。
2. [入力]を選択します。
3. 「保持」のチェックボックスをクリアにします。これで電断、LT本体のリセットによるデータの保持は、非保持になります。
4. [作成]をクリックします。これで変数「氷供給のカップセット」がディスクリート、入力として指定されました。
   このとき、[変数タイプ]ダイアログボックスで変数「氷供給のカップセット」に対して行った変数タイプの設定が[変数リスト]上に表示されています。
   [OK]をクリックした場合には、[変数タイプ]ダイアログボックスは閉じられます。ラングや命令の編集操作(挿入、ドラッグ＆ドロップ、クリックなど)上メリットがありますので、ここでは[変数リスト]や[変数タイプ]ダイアログボックスは開いたままにしておいてください。

参考:[変数リスト]ウィンドウに表示する変数タイプを選択するには、[表示]メニューで表示する変数タイプを選択します。選択した変数タイプの横にチェックマークが表示されます。

これで変数を作成し、変数タイプの設定方法がわかりました。次に、下の表に示す変数を作成します。変数は[変数タイプ]ダイアログボックスで直接作成できます。

変数名	変数タイプ	I/Oタイプ	保持/非保持	グローバル
電源オンボタン	ディスクリート	入力	非保持	非グローバル
氷供給のカップセット	ディスクリート	入力	非保持	非グローバル
氷供給ボタン	ディスクリート	インターナル	非保持	グローバル
カップ大ボタン	ディスクリート	インターナル	非保持	グローバル
カップ中ボタン	ディスクリート	インターナル	非保持	グローバル
カップ小ボタン	ディスクリート	インターナル	非保持	グローバル
電源オフボタン	ディスクリート	入力	非保持	非グローバル
氷供給	ディスクリート	出力	非保持	非グローバル
ソーダバルブ	ディスクリート	出力	非保持	非グローバル
ランプ	ディスクリート	出力	非保持	非グローバル
ソーダ注入時間	タイマ	インターナル	保持	非グローバル
カップ大個数	カウンタ	インターナル	非保持	非グローバル
カップ中個数	カウンタ	インターナル	非保持	非グローバル
カップ小個数	カウンタ	インターナル	非保持	非グローバル

完了したら、[変数タイプ]ダイアログボックスを閉じます。

参考:変数名を入力ミスした場合は、[変数リスト]ウィンドウの中の編集メニューの[名前の変更]で変更できます。

[変数リスト]をウィンドウ表示中、[Insert]キーを押すと[変数タイプ]ダイアログが表示されますので、すばやく変数を作成できます。

1.2.3　プログラムの保存

作成データの安全性のため、ロジックプログラムを定期的に保存しておくことをお勧めします。

■保存方法

ロジックプログラムエディタの[ファイル]メニューから「保存」を選択します。

参考：ツールバーの  をクリックするか、またはCTRL+Sを押すことによってプログラムを保存することもできます。

・ ロジックプログラムを保存すると、ロジックプログラムエディタで作成したグローバル変数は自動的にシンボルエディタにLogicシンボルとして登録され、画面エディタの表示機能でも共有できるようになります。

参照→「オペレーションマニュアル 作画編 4.2.5 シンボルエディタ」

まとめ

このレッスンでは、以下の操作を学習しました。

・変数の作成と変数に関係したダイアログボックスの使い方
・変数タイプの決め方
・プログラムの保存

1.3　ラング、命令、および分岐の挿入

ロジックプログラムを作成する最初のステップは、ラングの挿入です。
画面には、下のような空白のプログラムが表示されます。使用するチュートリアルプログラムの完成品は、C:\Program Files\Pro-face\LT\SAMPLE のフォルダの中に Soda.lte ファイルとして付属しています。

1.3.1　ラングの挿入

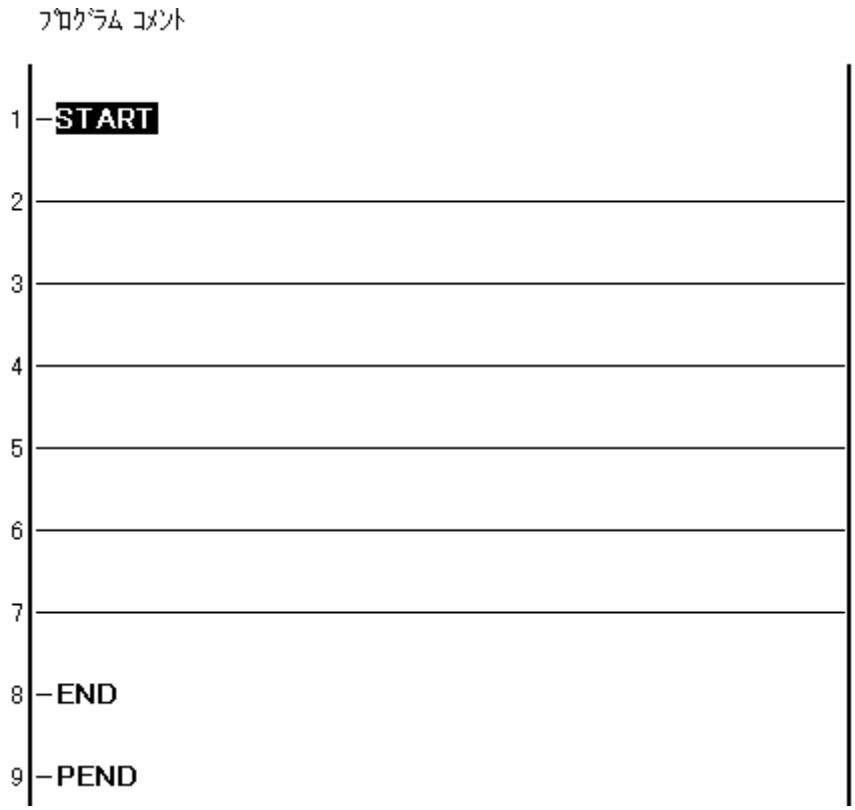
ロジックプログラムを新規作成します。
新しいプログラムの左側には、START、END、および PEND というラベルが付いた3つのラングがあります。

- START ラングは、メインプログラム領域の開始を表します。
- END ラングは、メインプログラム領域の終わりを表します。
- PEND ラングは、全プログラム領域の終わりを表します。PEND ラングの後にはラングを挿入できません。

START と END の間のラングは、毎スキャン実行されます。START の上に挿入されるラングは初期化プログラムを作成する領域です。この領域は、電源を投入した最初の1スキャンだけ実行されます。END と PEND の間の領域は、サブルーチンプログラムのために予約されます。
START、END、および PEND ラングの詳細については、オンラインヘルプの[リファレンスガイド]を参照してください。

■ラングの挿入手順

1. START の左側のラング番号1をクリックします。
   ラング1が選択されます。
2. 右クリックします。
   ショートカットメニューが表示されます。
3. [ラングの挿入]を選択します。(または、「挿入」メニューより「ラング」を選択します。)
   新しいラングが番号2（START ラングの下）に表示されます。
4. 上の方法を繰り返して、START ラングの下にさらに5つのラングを挿入します。
   画面は、次頁のようになります。

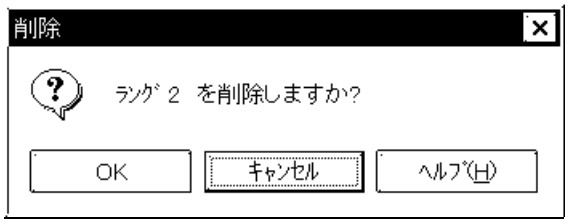


参考：[挿入]メニューから[ラング]を選択するか、ツールバーの をクリックすることでも、ラングを挿入することができます。

1.3.2　ラングの削除

■ラングの削除手順

1. 削除するラングを選択します。この例では、ラング2の左の番号2(ラング番号)をクリックします。
2. [Delete]キーを押します。または、右クリックしショートカットメニューより[ラングの削除]を選択します。[削除]ダイアログボックスが表示されます。



3.[OK]をクリックします。

参考：[編集]メニューから、[元に戻す]を選択するか、ツールバーの中の をクリックすると、ラング単位に1つ前の状態に戻すことができます。

1.3.3　命令の挿入

ロジックプログラムに命令を挿入し、それらに変数を割り付ける方法は数とおり用意されています。このチュートリアルでロジックプログラムを作成する際に、これらの方法について説明し、実際に試してみます。

命令の詳細については、**参照→「第7章 変数、オンラインヘルプ」**

■命令を挿入するラングの選択手順

1. ここでは命令をラング2に挿入します。
   ラング2のライン上のどこかをクリックします（ただし2という番号そのものをクリックしてはいけません）。選択されたラングが下のように表示されます。

2. 選択したラングにツールバーから命令を挿入できます。ロジックプログラムエディタのツールバーには、下に示すようなボタンがあります。  これらのボタンをクリックすると、選択したラングに命令が挿入されます。
   ボタンの意味は、下表のとおりです。

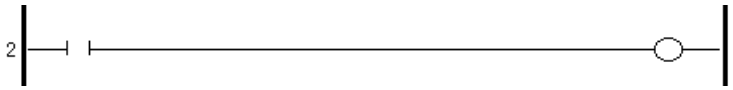
	**a接点(NO)**
	**b接点(NC)**
	**アウト・コイル(OUT)**
	**オンディレータイマ(TON)**
	**オフディレータイマ(TOF)**
	**アップカウンタ(CTU)**
	**ダウンカウンタ(CTD)**

■命令挿入手順1:ツールバーからの挿入

1. をクリックします。下のようにラング2が表示されます。

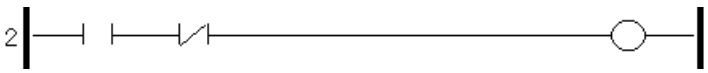
命令の上に入力ボックスが表示され、その中でカーソルがフラッシュしています。これは、命令パラメータボックスです。通常では、このボックスは命令に関連付ける変数名を入力しますが、ここでは無視してください。

2. をクリックします。ラング2の右側にアウト・コイルが挿入されます。命令パラメータボックスがフラッシュしていますが、ここでも無視してください。変数の入力方法はあとで説明します。**参照→1.4 命令への変数の割り付け**



3. ラング2上のa接点(NO)とアウト・コイル(OUT)命令の間をクリックします。

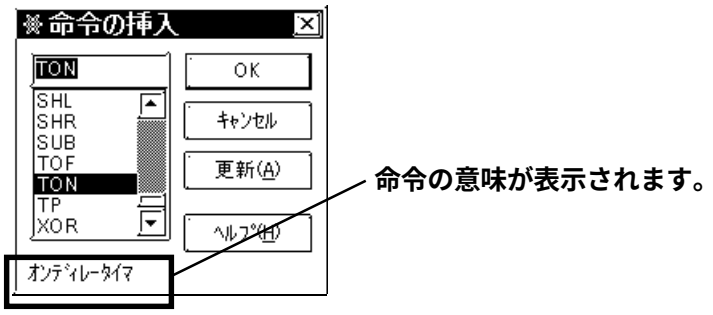
4. b接点(NC)ボタンをクリックします。
ラング2にb接点(NC)が挿入され、下のように表示されます。



参考:各ツールボタンの機能の説明は、カーソルをそのボタンの上に置いたときにステータスバー上に表示されます。ツールバーは、頻繁に使用する命令を挿入するのに便利ですが、ロジックプログラムエディタで利用できるすべての命令を含んでいるわけではありません。次に説明する2つの方法を使って命令を挿入することもできます。

■命令の挿入手順2：ダイアログボックスでのリスト選択による挿入

1. ラング3上のどこかを右クリックします。ショートカットメニューが表示されます。
2. [命令の挿入]を選択します。[命令の挿入]ダイアログボックスが表示されます。または、[挿入]をクリックして[命令]を選択しても同様に[命令の挿入]ダイアログボックスが表示されます。



このダイアログボックスには、ロジックプログラムエディタでロジックプログラムを作成するために使用できるすべての命令が含まれています。各命令を入力またはクリックすると、その命令の意味がダイアログボックスの下に表示されます。

参考:挿入メニューから[命令]を選択するか、ラングを選択した後で[INSERT]キーを押すことによって、[命令の挿入]ダイアログボックスを表示することもできます。各命令の詳しい説明は、命令を選択した状態で[ヘルプ]ボタンをクリックすることで見ることができます。

3. ここでは、オンディレータイマを選択します。オンディレータイマ(TON)が見つかるまで、[命令の挿入]ダイアログボックス内の命令リストをスクロールします。
4. オンディレータイマ(TON)を選択します。
   [変数タイプ]ダイアログボックスと同様に、選択を確認する際に、[OK]または[更新]のどちらかをクリックします。ここでは、ロジックプログラムにさらに命令を挿入しますから、[命令の挿入]ダイアログボックスを開いたままにしておくために[更新]をクリックします。

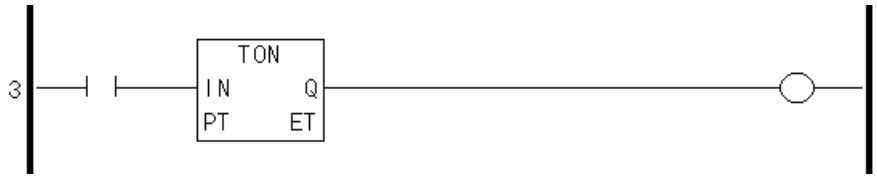
5. 先ほど挿入したオンディレータイマ(TON)命令の左側のラングをクリックします。
6. a接点(NO)が見つかるまで、[命令の挿入]ダイアログボックス内の命令のリストをスクロールします。
7. a接点(NO)をダブルクリックします。

■命令挿入手順 3:ダイアログボックスでの入力による挿入

1. 命令リストの上のフィールドにOUTと入力します。
参考：命令リストが自動的にスクロールして、アウト・コイル(OUT)命令がリストの最上段に表示されます。また、その名前がダイアログボックスの左下隅に表示されます。

2. オンディレータイマ(TON)命令の右側のラングを選択します。

3. [更新]をクリックします。
ラング3が下のように表示されます。

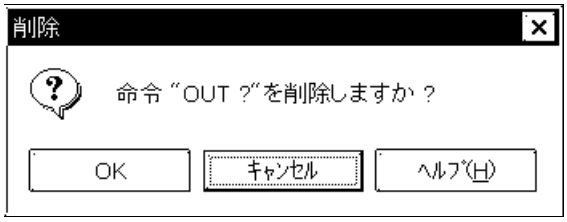


1.3.4　命令の削除

ラング3に挿入したアウト・コイル(OUT)命令を削除します。

■命令の削除手順

1. ラング3のアウト・コイル(OUT)命令を右クリックします。
ショートカットメニューが表示されます。
2. [命令の削除]を選択します。
[削除]ダイアログボックスが表示され、この命令を削除してもよいかどうかを尋ねます。



3. [OK]をクリックします。

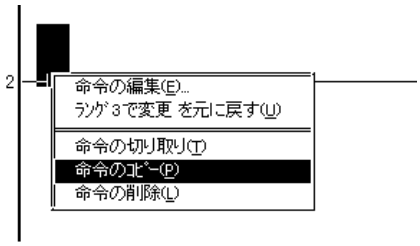
参考:削除する命令を選択して[Delete]キーを押すか、またはツールバーの[×]をクリックすることでも削除できます。

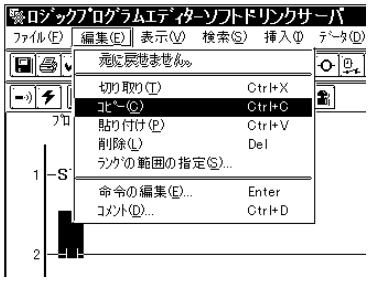
1.3.5　命令のコピー＆貼り付け

ここでは、命令のコピー＆貼り付けを練習します。

■命令をコピーする手順

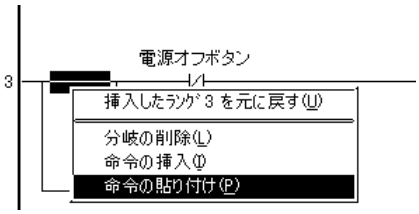
1. コピーしたい命令をクリックします。
2. 右クリックで[命令のコピー(P)]もしくは、メニューより[編集(E)]→[コピー(C)]を選択します。

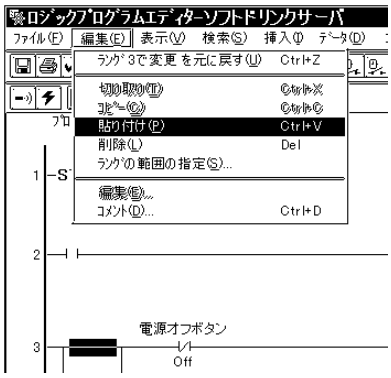




■命令を貼り付けする手順

1. 命令を配置したい場所を選択します。
2. 右クリックで[命令のコピー(P)]もしくは、メニューより[編集(E)]→[貼り付け(P)]を選択します。





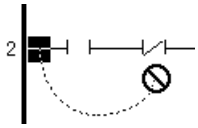
3. 以下のように命令が貼り付けられます。

1.3.6　分岐の挿入

この練習では、分岐をラング2のa接点(NO)とb接点(NC)の間に挿入します。この分岐はソフトドリンクサーバのライトを自己保持するために設計されています。

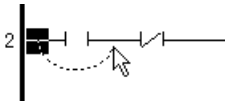
■分岐を挿入する手順

1. ラング上の分岐を開始したいポイントにカーソルを置きます。この場合は、ラング2のa接点(NO)のすぐ左です。
2. マウスをクリックし右にドラッグします。カーソルが⃠に変わり、そこから破線が描かれます。

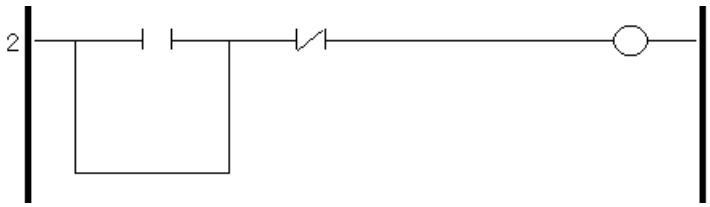


分岐の終点が無効な位置にある場合、カーソルが⃠のように表示されます。分岐の終点が有効な位置にある場合、カーソルは通常に戻ります。カーソルが通常の形で表示されるポイントでマウスを離すと、開始点とマウスを離した終点の間に分岐が挿入されます。カーソルが⃠のように表示されているときにマウスを離すと、分岐は作成されません。

3. マウスをクリックして、カーソルがa接点(NO)とb接点(NC)の間に置かれ、⃠ではなく通常の形で表示されるまで右にドラッグします。

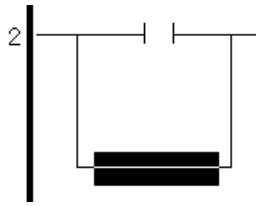


4. マウスを離します。これでa接点(NO)とb接点(NC)の間に分岐が挿入され、ラング2は下のように表示されます。

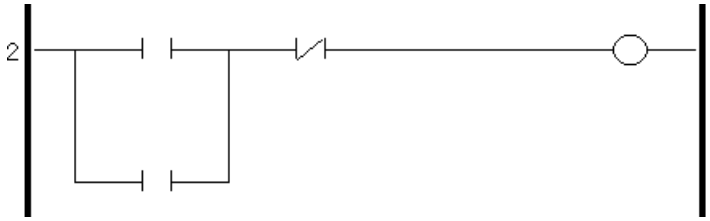


■分岐に命令を挿入する手順

1. 分岐の下の部分をクリックして、分岐を選択します。



2. [命令の挿入]ダイアログボックスはまだ開かれているはずです。
   開いていない場合は、前述のいずれかの方法を使って、このダイアログボックスを開きます。
3. [命令の挿入]ダイアログボックスからa接点(NO)命令を選択し、前述のいずれかの方法を使ってそれを挿入します。ラング2が下のように表示されます。



参考：命令が挿入されている分岐を削除するには、最初にその分岐の中のそれぞれの命令を選択して削除しなければなりません。

1.3.7　初期化ロジックプログラム

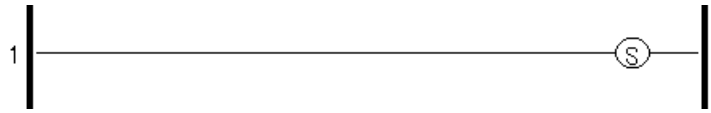
START ラングの上に挿入されているロジックは、初期化ロジックプログラムです。ここに記述されたロジックプログラムは、コントローラを RUN したときに最初のスキャン1回だけ実行されます。

■初期化ロジック挿入手順

1. STARTラングの上にある[プログラムのコメント]フィールドをクリックします。
   このフィールドが表示されていない場合は、[表示]メニューから[コメント]を選択してから、[プログラム]を選択します。
2. ショートカットメニューから[ラングの挿入]を選択します。
   ラングが START ラングの上に挿入されます。

参考：以降のラングは1つずつ繰り下がります(2番目にあったラングがラング3になります)。

3. 初期化ラング(ラング1)を右クリックします。
4. ショートカットメニューから[命令の挿入]を選択します。
5. [命令の挿入] ダイアログボックスから SET 命令を選択し、[OK] をクリックします。



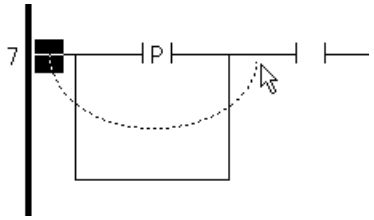
このラングは、ソフトドリンクサーバの製氷機の電源をオンにするために使用します。これは一度オンに設定しておけば、ソフトドリンクサーバの電源がオンになっている間、オンのままになっています。

参考：[オプション]ダイアログボックスで[ラングまたは命令をフォーカスの後に追加]を選択していない場合、ラングなどのオブジェクトは選択したライン上に挿入されます。初期化ラングを挿入するためには START ラングを選択してください。

これでロジックプログラムのラング3、4、および初期化ロジックプログラムの1つのラングが完成しました。次のページに示すようにラング5～7を作成してください。ロジックプログラム中の｜ P ｜命令とは、PT(立ち上がり接点)命令のことです。

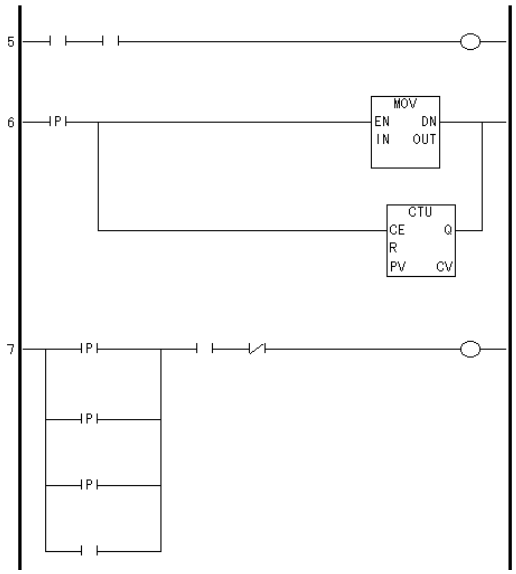
■ラング7の複数の分岐を挿入する手順

1. 最初の分岐を前述の方法で挿入します。
2. 2番目の分岐を挿入するために、前の分岐と同じポイントをクリックし、そこからドラッグします。
3. カーソルを、前の分岐を越えて、新しい分岐を挿入するラングのポイントまでドラッグします。



マウスを離すと、新しい分岐が前の分岐の上に挿入されます。

下の例では、命令がラング5～7に挿入されています。



まとめ

このレッスンでは、以下の操作を学習しました。

・ラングの挿入と削除
・命令の挿入と削除
・分岐の挿入と削除

1.4　命令への変数の割り付け

命令に変数を割り付ける方法を学びます。
「1.2 変数の作成」で、チュートリアルロジックプログラムで使用する変数リストを作成しました。ここでその[変数リスト]ウィンドウを開きます。

■[変数リスト]ウィンドウのオープン手順

1. [データ]メニューから、[変数リスト]を選択します。
2. このウィンドウを画面の左下隅に移動します。[命令の挿入]ダイアログボックスがまだ開かれている場合は、キャンセルをクリックして、閉じます。

1.4.1　命令パラメータボックス

前のレッスンで、最初に命令をラングに挿入したとき、カーソルがフラッシュしているフィールドが表示されたことを思い出してください。このフィールドが命令パラメータボックスです。このボックスに、命令に関連付ける変数を入力します。

■基本命令パラメータボックスへのアクセス手順

1. ラング3のアウト・コイル(OUT命令)をダブルクリックします。命令の上にテキストフィールドが表示され、その中のカーソルがフラッシュします。これがこの命令の命令パラメータボックスです。

参考：命令パラメータボックスは、命令をクリックして[Enter]を押すか、または命令を右クリックして、ショートカットメニューから[命令の編集]を選択することによってもアクセスすることができます。

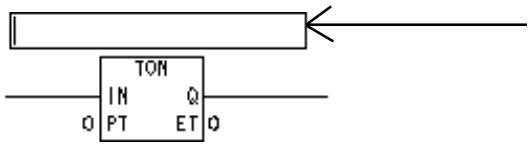
応用命令には、2つ以上の命令パラメータボックスがあります。たとえば、オンディレータイマ(TON命令)には、2つの命令パラメータボックスがあります。1つの命令パラメータボックスで変数を割り付け、もう1つの命令パラメータボックスでは、設定時間（ミリ秒単位）を入力します。

■応用命令パラメータボックスへのアクセス手順

1. ラング4のオンディレータイマ(TON命令)をクリックします。命令が下のように表示されます。

オンディレータイマ(TON命令)の上部の領域が黒く反転表示されています。この領域に、命令に割り付ける変数を入力します。下部の黒く反転表示されている領域は、設定時間(PT)の専用変数です。この領域には、設定時間（ミリ秒単位）を入力します。

2．オンディレータイマ(TON 命令)の上の黒く反転表示されている領域をダブルクリックします。これで命令パラメータボックスが選択されました。このとき、命令にタイマ変数(変数名)を割り当てることができます。



ここに変数名を入力します。
（命令パラメータボックス）

3．オンディレータイマ(TON命令)のPTのすぐ左にある領域をダブルクリックします。[データ値変更]ダイアログボックスが表示されます。このウィンドウには、タイマ出力(Q)がオンになるまでの設定時間（ミリ秒単位）を入力します。命令に変数を割り付ける方法については、次の項で説明します。

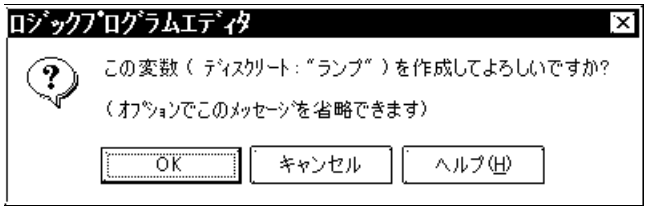
4．[データ値変更]ダイアログボックスを閉じます。

1.4.2　変数の入力

命令パラメータボックスに変数を入力する1つの方法として、命令パラメータボックスの中に直接に入力することができます。

■命令パラメータボックスへのテキスト入力手順

1．命令をダブルクリックして、ラング3のアウト・コイル(OUT命令)の命令パラメータボックスを選択します。
2．このボックスに「ランプ」と入力します。
3．[Enter]キーを押します。下のようなウィンドウが表示され、変数の作成を行うか確認します。



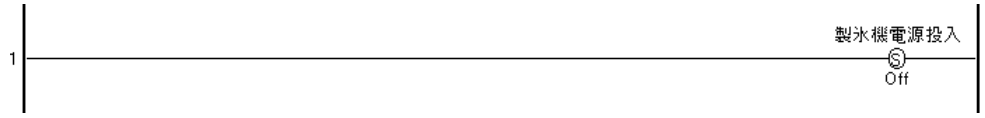
4．[OK]をクリックします。[変数リスト]ウィンドウのリストの中に「ランプ」という変数が表示されています。ロジックプログラムエディタは、命令に必要な変数タイプを自動的に変数に割り付けます。この場合、変数タイプとしてインターナルのディスクリートが割り付けられています。

参考：ロジックプログラムエディタは、命令に対して作成された新しい変数に必要な変数タイプを自動的に割り付けます。また、変数リストにすでにある変数を命令パラメータボックスの中に直接入力することもできます。入力が完了すると、自動的に変数が割り付けられます。上記の方法を使って、変数「電源オフボタン」をラング3のb接点(NC命令)に割り付けます。

参考：コイル命令(OUT、SET、RST、NEG)に割り付けた変数を保持型に変えると自動的に保持型のコイル命令(M、SM、RM、NM)に変わります。

ラング3は、下のように表示されます。

初期化ラングのセット・コイルに変数「製氷機電源投入」を割り付けます。この変数は、命令パラメータボックスに直接入力することによって作成できます。
初期化ラングが下のように表示されます。

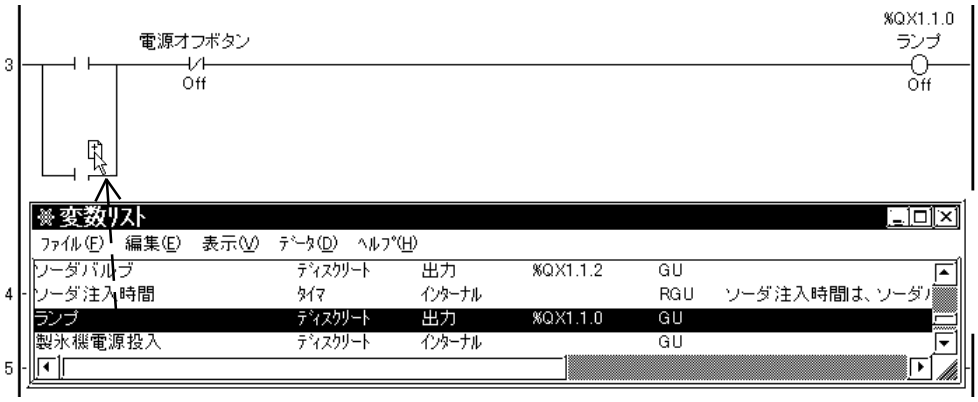


命令に変数を割り付けるもう1つの方法として、[変数リスト]ウィンドウの中の変数を命令にドラッグ&ドロップすることもできます。この方法は、多数の命令に同じ変数を割り付ける場合に便利です。この方法を使う利点は、「1.9 I/0の割り付け」で詳細を説明します。

■[変数リスト]ウィンドウによるマッピング手順

1. [変数リスト]ウィンドウの「ランプ」をクリックし、そのままマウスボタンを押したままにします。
2. マウスボタンを押したまま、「ランプ」をラング3の分岐の上のa接点(NO命令)までドラッグします。分岐を挿入する場合と同様に、カーソルは最初は🚫になります。
   カーソルがこの状態のとき、命令に変数を割り付けることはできません。
3. カーソルをラング3の分岐の上のa接点(NO命令)の上にドラッグします。

   カーソルは、下のように表示されます。



   カーソルが 🖱 のときにマウスを離すと、変数が割り付けられます。
   これで変数「ランプ」がラング3の分岐のa接点(NO命令)に割り付けられました。

4. 変数「電源オンボタン」をクリックし、ラング3上の他のa接点(NO命令)にドラッグします。
ラング3は、下のように表示されます。

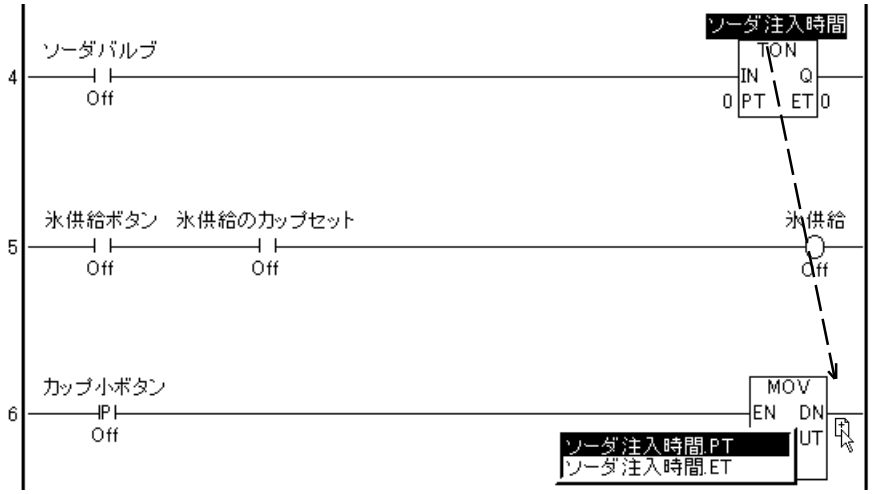
[参考]カーソルを命令のごく近くまで移動すると、カーソルは自動的に⃠から に変わります。この時点でマウスを離すと、変数がその命令に割り付けられます。
定数を変数として割り付けることもできます。入力方法は、通常の変数の場合と全く同じですが、ウィンドウからドラッグできないため、手操作で入力する必要があることです。

1.4.3　作業の完了

これで、命令に変数を割り付ける方法がわかりました。プログラムの残りのラングを完成させてください。完成したラングは、次ページの図のようになります。
ラング6のMOV命令とラング7のb接点(NC命令)には、それぞれ変数「ソーダ注入時間.PT」と「ソーダ注入時間.Q」が割り付けられています。これらの変数は、「ソーダ注入時間」変数が割り付けられているタイマのPTおよびQを参照します。
これらの変数を入力する手順には、以下の3つがあります。

- 命令パラメータボックスを選択し、その中に変数「ソーダ注入時間」を直接入力する。
- [変数リスト]ウィンドウから「ソーダ注入時間」変数をクリックしドラッグして入力する。
- 命令パラメータボックスをコピーしたい命令にドラックし、専用変数リストから選択して入力する。この方法について以下に示します。

操作手順
1. コピー元になる命令パラメータボックスを選択します。
2. カウンタ・タイマの変数をコピーしたい命令にドラックします。
3. 専用変数リストボックスより要素を選択し、ダブルクリックします。



[重要]

ラング6、7などで使用される。「ソーダ注入時間.PT」や「ソーダ注入時間.Q」のように応用命令の専用変数は、変数名+拡張子の形で使用できます。

命令によりますが、拡張子として、

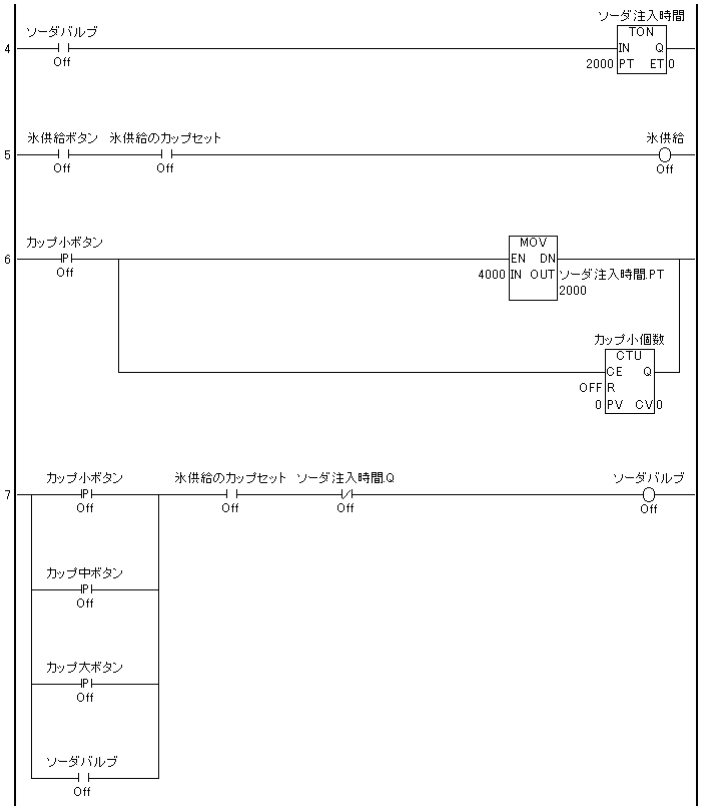
***.CV	現在値
***.PT	設定値
***.Q	出力ビット
***.R	リセットビット

などがプログラム上で使用できます。

参照→「7.2 変数タイプ」

＜チュートリアルプログラムの作成例＞

ここまでのレッスンで以下のロジックプログラムが作成されます。



まとめ

このレッスンでは、以下の操作を学習しました。

・命令に変数を割り付ける

1.5　ロジックプログラムのドキュメント化

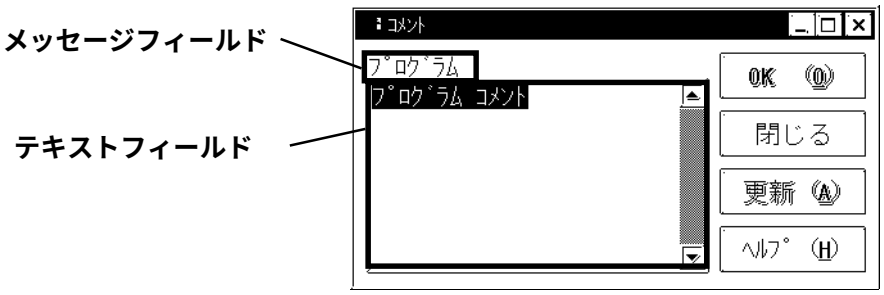
ロジックプログラムエディタでは、1. プログラム全体、2各ラング、3. 各変数にコメントを付加することができます。ロジックプログラムにコメントを付加しておくと、再読性が上がりデバックや追加修正の際に有効です。

1.5.1　プログラムコメントの追加

ロジックプログラムに追加する最初のコメントは、プログラムの機能のコメントです。

■プログラムコメントの追加手順

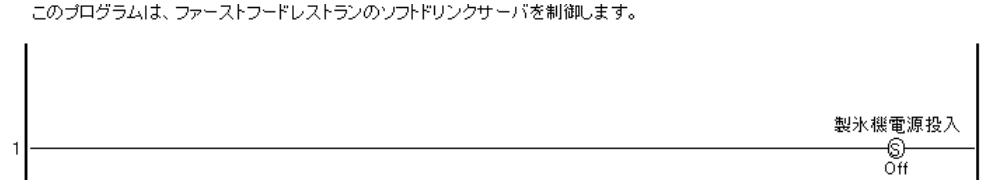
1. 画面最上段の[プログラムコメント]をダブルクリックします。
   画面に[コメント]ダイアログボックスが表示されます。



このウィンドウでプログラムのコメントを入力してください。

参考:[コメント]ダイアログボックスのメッセージフィールドの上に、[プログラム]と表示されています。これは、このテキストフィールドがプログラムのコメントの入力ボックスであることを示しています。

2. [プログラムコメント]テキストを選択します。
3.「このプログラムは、ファーストフードレストランのソフトドリンクサーバを制御します。」と入力します。
4. [OK]をクリックします。プログラムコメントがロジックプログラムの最上段に表示されます。表示されない場合は、画面を上にスクロールしてください。



参考:下側のステータスバーをダブルクリックしても、[プログラムコメント]の追加または編集ができます。

1.5.2　ラングコメントの追加

ロジックプログラムエディタでは、プログラムの各ラングにコメントを追加できます。下の例では、ラング5にコメントを追加します。

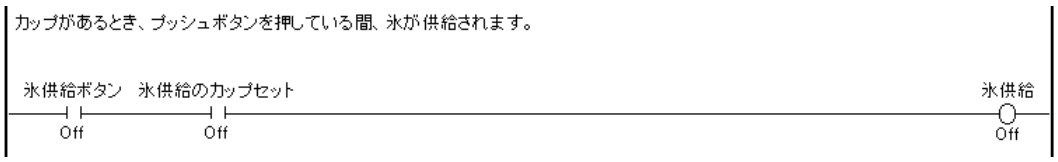
■ラングコメントの追加手順

1. ラング5の左にある番号5を右クリックします。
2. ショートカットメニューから[コメント]を選択します。[コメント]ダイアログボックスが開きます。このウィンドウは、プログラムコメントを挿入したときにオープンしたウィンドウと同じです。唯一の違いは、メッセージフィールドの上のコメントが[プログラム]ではなくラング5になっていることです。

参考：編集メニューから[コメント]を選択するか、またはツールバーの  をクリックすることでも、[コメント]ダイアログボックスをオープンできます。

プログラムのラング5は、アイスディスペンサーの制御のために使用されます。

3. [コメント]ダイアログボックスのテキストフィールドをクリックします。
4. 「カップがあるとき、プッシュボタンを押している間、氷が供給されます。」と入力します。
5. [更新]をクリックします。



[コメント]ダイアログボックスを開いておくと、プログラムの残りのラングにコメントを追加するのが簡単になります。

■ラング3へのコメントの追加手順

1. ラング3上の命令パラメータボックス以外の場所をクリックします。
   [コメント]ダイアログボックスの最上段のコメントがラング3に変わります。
2. テキストフィールドをクリックします。
3. 「電源オフボタンを押すまで、ライトは点灯したままになります」と入力します。
4. [更新]をクリックします。このチュートリアルでは、ラング3および5だけを例にあげドキュメント化しました。

1.5.3　変数へのコメントの追加

ロジックプログラムの各変数にコメントを追加できます。ラベルまたは定数にはコメントを追加できません。

■変数コメントの追加手順

1. 本チュートリアルの流れでは、[変数リスト]ウィンドウが開かれているはずです。
   開いていない場合は、[データ]メニューから[変数リスト]を選択して開きます。
2. [コメント]ダイアログボックスも開かれているはずです。
   開いていない場合は、[編集]メニューから[コメント]を選択して開きます。
3. 変数「ソーダ注入時間」を表示している命令パラメータボックスをクリックします。
   [コメント]ダイアログボックスのメッセージフィールド上に「ソーダ注入時間」というメッセージが表示され、[変数リスト]ウィンドウでも「ソーダ注入時間」が反転表示されます。
4. [コメント]ダイアログボックスのテキストフィールドをクリックします。
5. 「ソーダ注入時間は、ソーダバルブを開けておく時間を決定します。動作時間は設定値に依存します。」と入力します。
6. [更新]をクリックします。

参考：変数にコメントを追加する方法として、変数をロジックプログラムから選択する方法のほかに、[変数リスト]ウィンドウから選択する方法もあります。

■変数リストからコメントを追加する手順

この例では、電源オンボタンにコメントを追加します。

1. [変数リスト]ウィンドウの中の変数「電源オンボタン」をクリックします。
   [コメント]ダイアログボックスのメッセージフィールドに「電源オンボタン」というメッセージが表示されます。
2. [コメント]ダイアログボックスのテキストフィールドをクリックします。
3. 「電源オンボタンを押すと、ソフトドリンクサーバが起動します。」と入力します。
4. [更新]をクリックします。

このチュートリアルでは、「ソーダ注入時間」変数と「電源オンボタン」変数のコメントだけを例にあげ追加しました。他のすべての変数のコメントも、同じ方法で作成できます。

1.5.4　[コメントリスト]ウィンドウ

[コメントリスト]ウィンドウは、プログラム中のすべての変数およびラングのコメントを一覧表示します。

■[コメントリスト]ウィンドウを表示する方法

[表示]メニューから、[コメントリスト]を選択します。

■[コメントリスト]ウィンドウから詳細なコメントを表示する方法

[コメントリスト]ウィンドウの中の「ソーダ注入時間」変数をダブルクリックします。
[コメント]ダイアログボックスに、「ソーダ注入時間」変数の詳細なコメントが表示されます。
[変数リスト]、[コメント]、[コメントリスト]の表示内容はプログラム中のラングや変数を選択するのに追従し選択表示されますが、[変数リスト]、[コメント]、[コメントリスト]表示内容を選択してもロジックプログラムの上は追従しません。
ロジックプログラムエディタでは、検索機能により特定の変数を簡単に見つけることができます。これは、「1.8　ロジックプログラム内の移動」で説明します。

まとめ

このレッスンでは、以下の操作を学習しました。
・プログラム、ラング、および変数にコメントを追加する
・[コメントリスト]ウィンドウを表示する

1.6　ラングのコピー、切り取りおよび貼り付け

ロジックプログラムでは、いくつかのラングで同じ命令シーケンスを入力しなければならないことがあります。すでに作成したラングをコピーして貼り付けることによって、工数を削減できます。

1.6.1　ラングのコピー

以下のチュートリアルでは、2つのラングがラング5とラング7の間に追加されます。これらの追加のラングには、ラング6と同じ命令があり、異なる変数が割り付けられています。

■ラングのコピー手順

1. ラングの左の番号6をクリックして、ラング6全体を選択します。
2. [編集]メニューから、[コピー]を選択します。

参考：切り取りまたはコピーするラングの範囲を選択する方法。
選択する範囲の中の先頭ラング番号をクリックします。次に、[Shift]キーを押したまま、選択する最後のラング番号を選択します。こうすると2つのラングの間にあるすべてのラングが選択され、切り取りまたはコピーすることができます。一度の操作でコピーできるラングの数は、25ラング程度が目安です(ラング内の命令数によって前後します)。

1.6.2　ラングの貼り付け

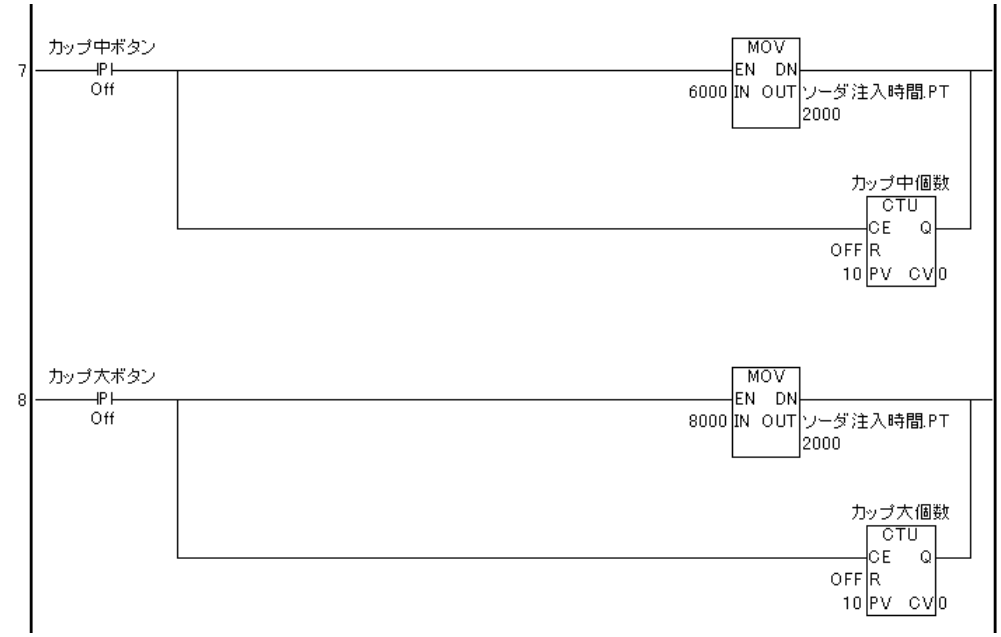
ラング全体が選択されている場合を除いて、ロジックプログラムエディタは、コピーまたは切り取りされたラングを、現在選択されているラングの下に貼り付けます。
[オプション]ダイアログボックスで[ラングまたは命令をフォーカスの後に追加]が選択されていない場合、コピーされたラングは、現在のラングの上に挿入されます。

【重要】
切り取り、コピーされたラングは、一旦内部のクリップボードにロードされます。
クリップボードからラングを貼り付けるとき、ラング全体を選択している場合、ロジックプログラムエディタは、全ラングをクリップボード上のラングに置換します。

■ラングの貼り付け手順

1. ラング6上のどこかをクリックします。
2. 編集メニューから[貼り付け]を選択します。ラング6とラング7が同じになります。
3. ラング6上のどこかをクリックします。
4. 編集メニューから、[貼り付け]を選択します。ラング6～8がすべて同じように表示されます。

参考:ラングを貼り付けしたとき、そのラングに関連付けられているすべての変数およびコメントもそのまま貼り付けられます。目的のプログラムに合わせてラングを編集してください。このチュートリアルでは、ラング7および8の変数を、下図のように変更してください。



5. 上の例のようにラング上のPT命令の変数名を変更します。

1.6.3　[切り取り]コマンドの使用

ロジックプログラムエディタでは、プログラムの一部からラングまたはラングの一部を取り出し、それを他の部分へ移動できます。これは、[切り取り]コマンドを使って実行します。以下の練習では、ラング4をプログラムの最後のラングへ移動します。

■[切り取り]コマンドの使用手順

1. ラング4のラング番号をクリックします。ラング4全体が選択されます。
2. 編集メニューから、[切り取り]を選択します。
   ラング4全体が、ロジックプログラムから切り取られ、クリップボードに入れられます。
3. ラング8上のどこかをクリックします。
4. 編集メニューから、[貼り付け]を選択します。ラング4がラング8の下に付加されます。
   プログラムの終わりは、下の図のようになります。

参考:ラング全体をプログラムの他の部分に移動するには、最初にラングを選択し、その中央部をドラッグして、新しい位置まで移動できます。

まとめ

このレッスンでは、以下の操作を学習しました。

・ラングをコピー、切り取りおよび貼り付ける

1.7　サブルーチンおよびラベル

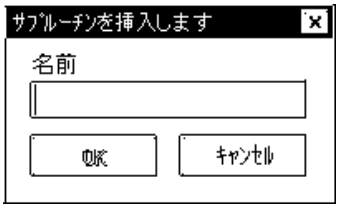
JSR(ジャンプサブルーチン)またはJMP(ジャンプ)命令が挿入されると、コントローラは、そのサブルーチンまたはラベルまでジャンプして実行します。サブルーチンとラベルの主な違いとして、JSR命令では、ENDとPENDの間に作成された同一名サブルーチンを実行したあと、ロジックプログラムの中のJSR命令の次の位置に戻りますが、JMP命令ではJMP命令と同名のラベルにジャンプしロジックプログラムの実行を続け、そのままでは元のJMP命令の位置には戻りません。
JMP命令およびJSR命令の詳細については「9.2.44 JMP（ジャンプ）/9.2.45 JSR（ジャンプサブルーチン)」を参照してください。

1.7.1　サブルーチンの挿入

すべてのロジックプログラムの終わりには、ENDおよび PENDというラベルが付いています。ENDラベルは、メインプログラム領域の終わりを知らせます。ロジックプログラムエディタは、スキャンごとにSTART～ENDまでの命令を実行します。ENDラベルとPEND(プログラムの終わり)ラベルの間の領域はサブルーチンのために予約されています。以下のチュートリアルで、サブルーチンを追加します。

■サブルーチンの追加手順

1. ENDラベルをクリックします。
2. 挿入メニューから、サブルーチンを選択します。
   [サブルーチンを挿入します]ダイアログボックスが表示されます。



3. [サブルーチンを挿入します]ダイアログボックスの名称フィールドに、「カウンタのリセット」と入力します。サブルーチン名/ラベルは、最大半角32文字、全角16文字以内で設定されます。サブルーチン名/ラベルを数字で始めないでください。サブルーチン名は、“_”以外の記号は使用できません。大文字と小文字、および全角と半角は区別しません。
   参照→1.2.1 変数リストの作成
4. [OK]をクリックします。ENDとPENDの間に、挿入されたサブルーチンが表示されます。

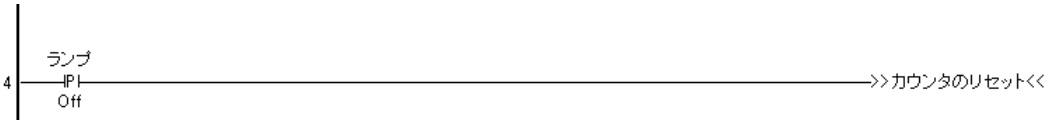
SUB STARTカウンタのリセットおよびSUB ENDカウンタのリセットというラベルが付いた2つの新しいラングが、ENDラングとPENDラングの間に挿入されました。サブルーチンを挿入する場所は、これらの2つのラングの間です。

5. SUB STARTカウンタのリセットラベルを右クリックします。
6. ショートカットメニューから[ラングの挿入]を選択します。ラングがSUB STARTラングとSUB ENDラングの間に挿入されます。
7. SUB STARTラングとSUB ENDラングの間にあるラングを右クリックします。
8. そのラングにアウト・コイル(OUT命令)を挿入します。
9. アウト・コイル(OUT命令)の両側に2つの分岐を挿入します。
10. それぞれの分岐にアウト・コイル(OUT命令)を挿入します。下の図は完成したサブルーチンです。
    これは、LTに電源を入れるたびに各カウンタをリセットするために使用されます。

各アウト・コイル(OUT命令)に変数が割り付けられます。例のようにこれらの変数をここで割り付けてください。
以上でサブルーチンが完成しました。ロジックプログラムに2つ以上のサブルーチンを追加するには、SUB ENDラングまたはENDラングを選択し、手順2)～6)を繰り返します。サブルーチンをロジックプログラムの中の特定のポイントで実行する場合、JSR命令を挿入しなければなりません。
次にこの操作を練習します。このサブルーチンは、ラング3の「ランプ」出力コイルがオンになったときに実行させます。したがって、この例では、JSR命令をラング4に置かなければなりません。

■ JSR命令の挿入手順

1. ラング3を選択します。
2. 挿入メニューから、[ラング]を選択します。
3. ラング4にPT命令を挿入します。
4. 変数「ランプ」をPT命令に割り付けます。
5. [命令の挿入]ダイアログボックスを使って、JSR命令をPT命令の右に挿入します。
6. JSR命令の命令パラメータボックスの中に、サブルーチンの名前「カウンタのリセット」を入力します。ラングは下のように表示されます。



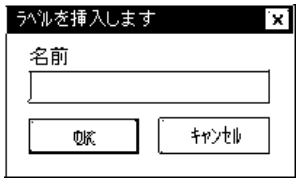
JSR命令「カウンタのリセット」が検出されたとき、サブルーチン「カウンタのリセット」にジャンプします。サブルーチンの実行が完了した後、ラング5から実行を再開します。
参考:サブルーチンを削除するには、はじめにそのサブルーチンの中の個々のラングを削除し、SUB STARTラングを削除してください。(SUB ENDラングは、 SUB STARTラングと一緒に削除されます。)

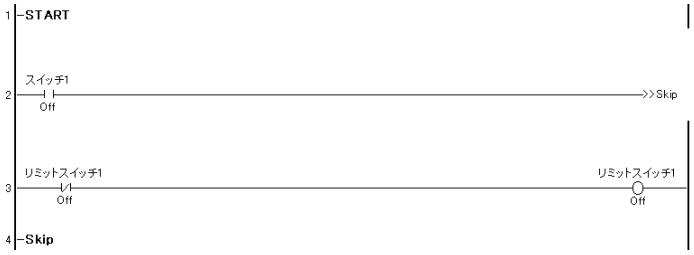
1.7.2　ラベルの挿入

JMP(ジャンプ)命令とジャンプ先のラベルをロジックプログラムの任意の部分に挿入できます。コントローラがJMP命令を実行したとき、JMP命令は、それに割り付けられている同名のラベルにジャンプし、そこからプログラムを実行します。
[オプション]ダイアログボックスの中で[ラングまたは命令をフォーカスの後に追加]が選択されているかどうかによって、ラベルは選択されているラングの上または下に挿入されます。このチュートリアルではラベルを使用しませんが、ラベルを挿入するには、以下の手順でラベルは挿入できます。

■ラベルをロジックプログラムに割り付ける手順

1. ラング上のどこかをクリックします。
2. [挿入]メニューから、[ラベル]を選択します。[ラベルを挿入します]ダイアログボックスが表示され、ラベルの名前の入力を要求してきます。





これは、ロジックプログラム中のJMP命令に割り付けられる変数と同名にしてください。JMP命令は、同名のラベルにジャンプを行います。

■ JMP 命令の挿入手順

1. ジャンプを開始するラングの最後の命令の右側をクリックし、[挿入]メニューから[命令]を選択します。
2. [命令]ダイアログボックスのJMP命令をダブルクリックします。JMP命令は必ずラングの最後の命令として挿入されます。ロジックプログラムエディタは、プログラムの中でこの命令を検出したとき、指定された同名のラベルにジャンプします。

まとめ

このレッスンでは、以下の操作を学習しました。
・サブルーチンおよびラベルを作成する
・JMP(ジャンプ)命令およびJSR(ジャンプサブルーチン)命令を挿入する。

1.8　ロジックプログラム内の移動

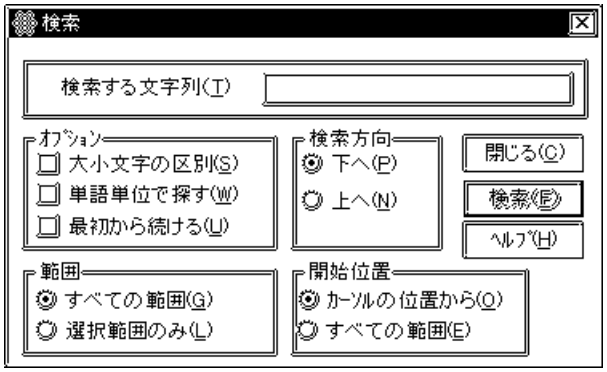
ここではロジックプログラムエディタで該当のラングや変数をすばやく検索する方法を説明します。ロジックプログラムエディタには、検索のために、[検索]、[リファレンス]、[ブックマーク]、[指定ラングへ移動]、および[指定ラベルへ移動]などのコマンドがあります。

1.8.1　[検索]コマンド

[検索]コマンドは、ロジックの中の特定のテキストを検索します。

■[検索]コマンドの使用手順

1. 開かれているウィンドウがあれば、ここでそれを閉じます。
2. [検索]メニューから、[検索]を選択します。[検索]ダイアログボックスが表示されます。



参考：[検索]ダイアログボックスは、ツールバーの中の  をクリックすることによってオープンすることもできます。

◆検索に使用する一致パターンのタイプの指定ができます。

- Fillという語を探す場合、ロジックプログラムエディタは、fillという語を含むすべての語を検出します（小文字のfillや他の語の一部になっている場合、たとえばFilletも検出されます）。
- [大小文字の区別]を選択した場合、Fillを検出しますが、fillを検出しません。また、[単語単位で探す]を選択した場合、Fillを検出しますが、Filletを検出しません。

◆検索の範囲および方向の指定

- [選択範囲のみ]を選択している場合、検索の範囲は、プログラムの中の反転表示されている部分だけです。
- [すべての範囲]を選択すると、検索の範囲は、プログラム全体を含みます。[すべての範囲]を選択することによって、選択範囲の最上段から検索を開始できます。また、[カーソルの位置から]を選択することによって、指定した位置から検索を開始できます。この例では、検索をプログラムの最初から開始します。

3. プログラムの中のSTARTラベルを選択します。
4. [検索]ダイアログボックスの[検索する文字列]フィールドをクリックします。
5. 「ソーダ」と入力します。

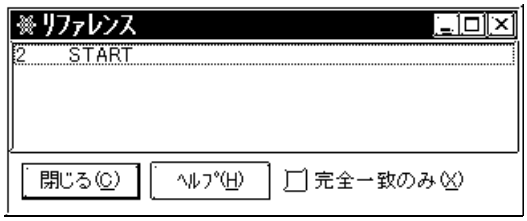
6. [すべての範囲]、[下へ]、または[カーソルの位置から]を選択します。
7. [検索]ボタンをクリックします。フォーカスは、検出した最初の一致、すなわち「ソーダ注入時間」変数の最初の部分に移ります。
8. [検索]ボタンをもう一度をクリックします。フォーカスは検出した2番目に一致する「ソーダ注入時間」に移動します。一致するエントリがなくなった時、ビープ音が鳴ります。
   参考：最初の[検索]コマンドの実行の後、[検索]メニューから[次を検索]を選択すれば、同じ条件で検索を続け、次に一致する文字列を検出できます。

1.8.2　[リファレンス]コマンド

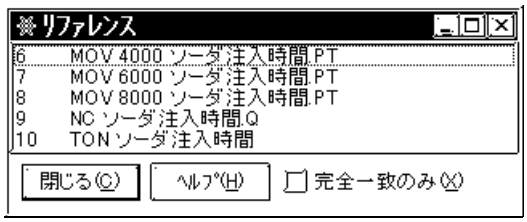
[リファレンス]コマンドは、ロジックプログラムの中の指定した変数が複数以上使用されている場合、使用されているすべての位置を検出します。このコマンドは、ラング番号とその変数が含まれる命令をリストします。このチュートリアルでは、START ラベルを選択します。[リファレンス]コマンドは、プログラムの中のどこからでも実行できます。

■[リファレンス]コマンドの使用手順

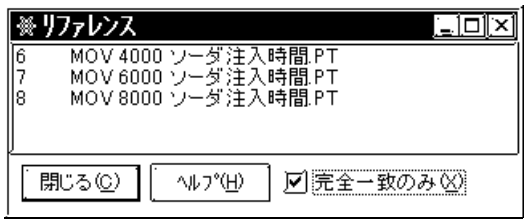
1. START ラベルをクリックします。
2. [検索]メニューから、[リファレンス]を選択します。[リファレンス]ダイアログボックスが表示されます。



3. [リファレンス]ダイアログボックスのサイズを変更し、画面の右下隅に移動します。
4. ラング6の「ソーダ注入時間 .PT」をクリックします。下のように[リファレンス]ダイアログボックスが表示されます。



5. [完全一致のみ]を選択します。[リファレンス]ダイアログボックスは下のように表示されます。



[リファレンス]ダイアログボックスには、下記の情報が表示されます。

- 行の左の数値は、変数が検出されたラングの番号を示します。この画面は、「ソーダ注入時間」を含む変数がラング6、7、8、9、および10にあることを知らせています。[完全一致のみ]を選択したとき、画面は、「ソーダ注入時間.PT」がラング6、7、および8にあることを示します。
- 次の項目は、命令タイプです。ラング上で変数が割り付けられている命令です。この画面は、「ソーダ注入時間」変数が3つのMOV命令と、1つのb接点(NC命令)、および1つのオンディレータイマ(TON命令)に使用されていることを知らせます。
- 3番目の項目は、命令に割り付けられているパラメータをリストしています。(最初に参照した変数を含む)。例の6、7、8ラングでは、それぞれMOV命令のINには整数4000、6000、8000がOUTには、「ソーダ注入時間.PT」が割り付けられていることがわかります。

[リファレンス]ダイアログボックスに表示されたリストを選択するごとに、選択した変数の位置にロジックプログラムの表示が追従します。プログラム修正や選択箇所のモニタの際に一瞬で関連プログラムに移動できます。

参考:[リファレンス]ダイアログボックスに対応する情報を表示するには、命令ではなくパラメータをクリックしなければなりません。

1.8.3　[リファレンス]ダイアログボックスと他のウィンドウの併用

[リファレンス]ダイアログボックスを使うときには、探している変数の少なくとも1つの所在がわかっていないと非常に面倒です。[検索]コマンドを使って見つけることもできますが、もっと早い方法があります。[リファレンス]ダイアログボックスと[変数リスト]ウィンドウおよび[コメントリスト]ウィンドウを合わせて使用するという方法です。

■[リファレンス]ダイアログボックスを他のウィンドウと共に使用する手順

1. [変数リスト]ウィンドウ、[コメントリスト]ウィンドウ、[リファレンス]ダイアログボックスを開きます。
2. これらのウィンドウを移動し、下の図のようにサイズを変更します。

3.[変数リスト]ウィンドウの中の変数「ソーダ注入時間」をクリックします。
参考:[コメントリスト]および[リファレンス]ダイアログボックスの表示は、選択に従って変化します。[リファレンス]ダイアログボックスには、変数「ソーダ注入時間」に含まれるすべての項目が表示されています。また、これらのウィンドウの表示が変化しても、ロジックプログラムの表示位置は変化しません。[リファレンス]ダイアログボックスで任意の変数行を選択すると、ロジックプログラムの中の変数に対応するポイントが表示されます。

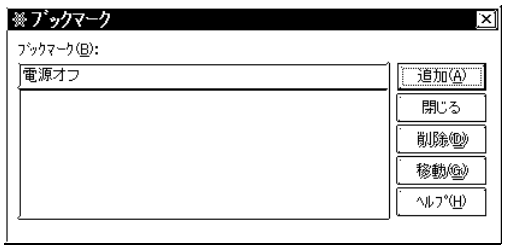
4. [リファレンス]ダイアログボックスの最初の行をクリックします。ロジックプログラムは、ラング上でユーザーが指定した命令および変数を反転表示しています。

1.8.4　ブックマークの使用

ロジックプログラムの中の特定のポイントを繰り返し参照し、そのたびにそこへスクロールするのが面倒である場合、ブックマークを使用します。ブックマークを設定するには、その設定位置を正確に指定しなければなりません。選択または反転表示できる項目はどれでもブックマークとして指定できます。この例では、ラング3のNC命令をブックマークとして設定します。

■ブックマークの設定手順

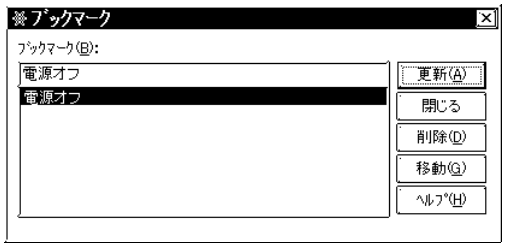
1. ラング3のNC命令をクリックします。
2. [検索]メニューから、[ブックマーク]を選択します。[ブックマーク]ダイアログボックスが表示されます。



3. [ブックマーク]フィールドに「電源オフ」と入力し、[追加]をクリックします。
これでブックマークが設定されました。「電源オフ」を選択し[移動]を入力してブックマークに戻ると、プログラムのどこからでも必ずラング3のNC命令に戻ります。新しいブックマークを設定するときは、ロジックプログラムの新しいポイントを選択し、手順1)～3)を繰り返します。ロジックプログラムエディタでは、複数のブックマークが利用できます。

■ブックマークに飛ぶ手順

1. [検索]メニューから、[ブックマーク]を選択します。[ブックマーク]ダイアログボックスが表示されます。



2. [ブックマーク]ダイアログがリストから任意のブックマークを選択し、[移動]をクリックします。
ロジックプログラムが表示されているときは、自動的にブックマークの場所に戻ります。
参考：CTRL+Mを使って[ブックマーク]ダイアログボックスをオープンできます。

■ブックマークの位置の変更手順

1. ロジックプログラムの中の新しい位置を選択します。
2. 位置を変更したい任意のブックマークリストから選択します。
3. [ブックマーク]ダイアログボックスの[更新]をクリックします。

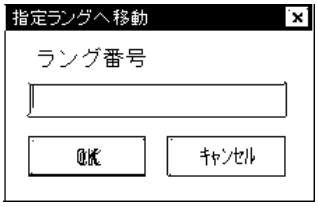
以上の手順で、ブックマークの変更ができます。

1.8.5　[指定ラングへ移動]コマンドの使用

[指定ラングへ移動]コマンドを使って、選択位置をロジックプログラムの指定したラングに移動できます。

■[指定ラングへ移動]コマンドの使用手順

1. [検索]メニューから、[指定ラングへ移動]を選択します。[指定ラングへ移動]ダイアログボックスが表示されます。



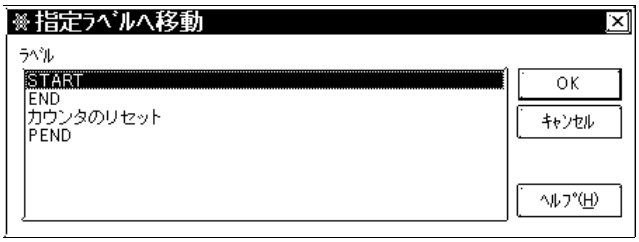
2. ラング番号を入力します。
3. [OK]をクリックすると、指定したラングに移動します。

1.8.6　[指定ラベルへ移動]コマンドの使用

[指定ラベルへ移動]コマンドを使って、ロジックプログラムの特定のラベルにジャンプできます。

■[指定ラベルへ移動]コマンドの使用手順

1. [検索]メニューから、[指定ラベルへ移動]を選択します。[指定ラベルへ移動]ダイアログボックスが表示されます。



2. 移動したいラベルを選択します。
3. [OK]をクリックすると、指定したラベルに移動します。

まとめ

このレッスンでは、以下の操作を学習しました。

・[検索]、[参照]、[ブックマーク]、[指定ラングへ移動]、および[指定ラベルへ移動]の各コマンドの使用方法

1.9　I/0の割り付け

ロジックプログラムの作成が完了した後、選択した変数に実I/0を割り付けることで、実際のI/0制御が可能です。このチュートリアルでは、使用方法を順序良く論理的に説明するため最初に変数を作成し、ロジックプログラムが完成した後に実I/0を割り付ける流れを取りましたが、プログラミングを開始する前に実I/0の配置が決まっていれば最初にI/0割り付けを行い、その後でプログラム上に実I/0と共に変数を割り付けることができます。以下のレッスンでは両方の方法を説明します。

1.9.1　実I/0への変数の割り付け

ロジックプログラムの変数を作成した後、いくつかの方法でそれをI/0に割り付けることができます。割り付ける変数は以下の通りです。
「氷供給ボタン」、「カップ大ボタン」、「カップ中ボタン」、「カップ小ボタン」は、LTからのタッチパネル入力とするので割り付けません。

変数名	端子タイプ	端子番号
電源オンボタン	入力	I0
氷供給のカップセット	入力	I2
電源オフボタン	入力	I6
ランプ	出力	Q0
氷供給	出力	Q1
ソーダバルブ	出力	Q2

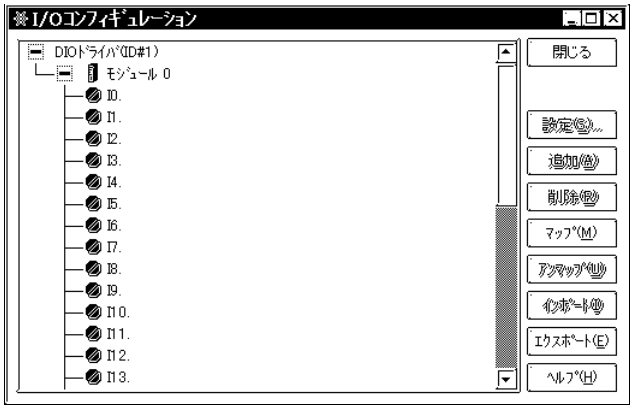
LT Type AとLT Type B、LT Type B+、LT Type CではI/0ドライバの違いにより「I/0コンフィギュレーションのオープン手順」と「ドライバのセットアップ手順」が異なります。
　Type Aをご使用の場合は「DIOドライバのセットアップ手順」
　Type B、Type B+、Type Cをご使用の場合は「Flex Networkドライバのセットアップ手順」をご覧ください。
LT Type Hについては、「LT Type Hシリーズ I/0設定ユーザーズマニュアル」（別売）を参照してください。

また、これまでのレッスンはLT Type Aをモデル環境として説明してきましたが、「Flex Networkドライバのセットアップ手順」の項目のみLT Type B+をモデル環境として説明します。

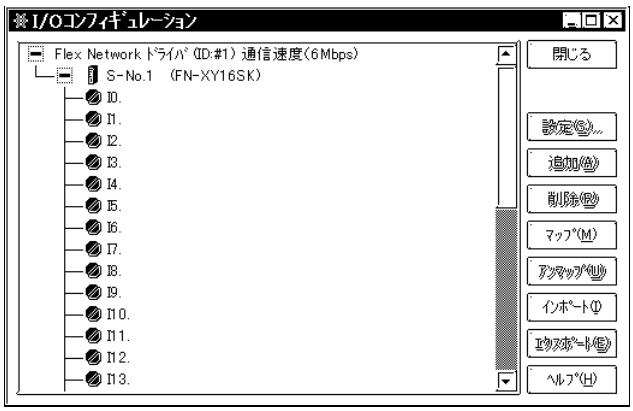
■I/Oコンフィギュレーションのオープン手順

[データ]メニューから、[I/0コンフィギュレーション]を選択します。[I/0コンフィギュレーション]ウィンドウが表示されます。

◆DIOドライバの場合



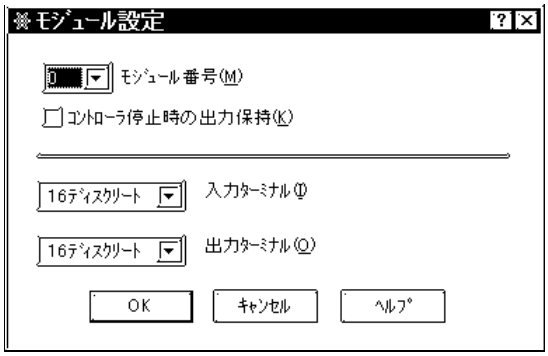
◆Flex Networkドライバの場合



参考:ツールバーの  をクリックするか、[変数タイプ]ダイアログボックスの[I/0(F)...]をクリックしても、[I/0コンフィギュレーション]ウィンドウをオープンできます。

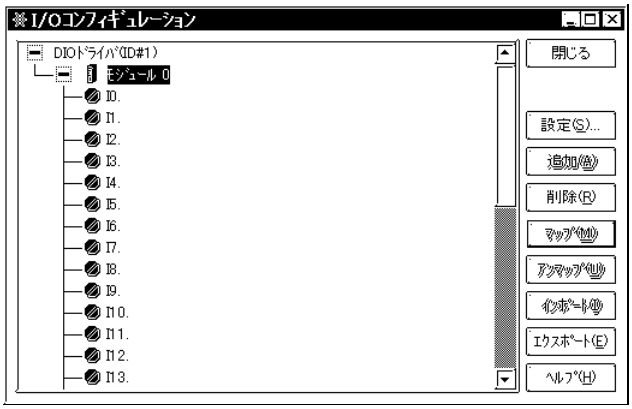
■DIOドライバのセットアップ手順

1. モジュール0を選択します。
2. [設定]をクリックします。設定ウィンドウが表示されます。



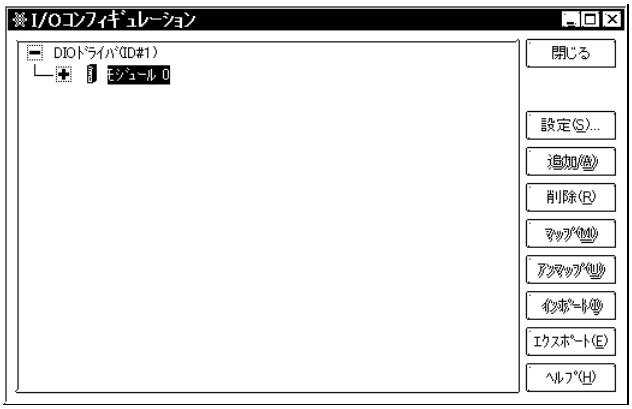
※モジュール番号は“0”のみ有効です。

3. デフォルトで入力/出力は16ディスクリート(ビット型)に設定されています。
4. [OK]をクリックします。下のように、[I/Oコンフィギュレーション]ウィンドウが表示されます。



モジュール0の下には、DIOモジュールに関連付けられている16個の入力端子と16個の出力端子が割り付けられています。このレッスンの後半で、それらに変数を割り付けます。

5. モジュール0の隣の⊟をクリックします。端子が非表示にされ、⊟の代わりに⊞が表示されます。



■Flex Networkドライバのセットアップ手順

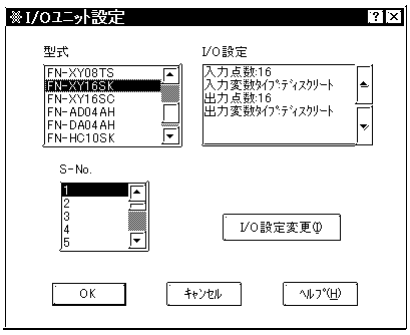
LT Type B+の本体に内蔵されているI/Oへの割り付けは、下記のドライバを設定した後、I/Oを割り付けます。（LT Type B+に内蔵されているDIOは、Flex Networkの1局として扱われます。）

型式　　：FN-XY16SK

S-No.　：他の接続機器と重複しない番号を設定してください。

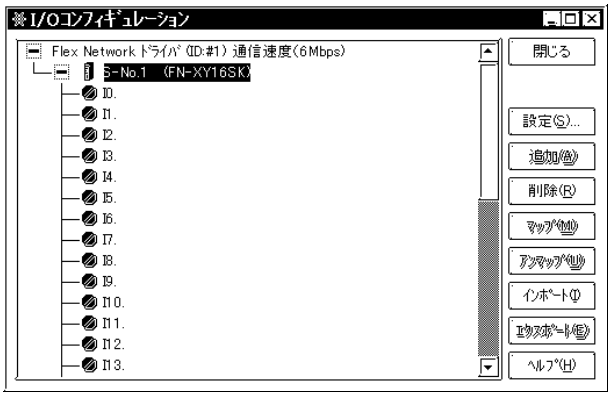
本レッスンでは「1」に設定します。

1.[I/Oコンフィギュレーション]ウィンドウで[S-No.1 (FN-XY16SK)]を選択して、[設定]をクリックします。下の[I/Oユニット設定]ウィンドウが表示されます。



2.「型式」を「FN-X16TS」から「FN-XY16SK」に変更します。

3.[OK]をクリックします。下のように、[I/Oコンフィギュレーション]ウィンドウが表示されます。



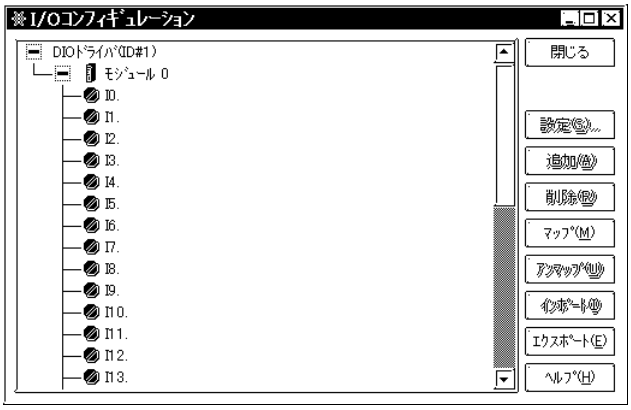
S-No.1 (FN-XY16SK)の下には、Flex Networkドライバに関連付けられている16個の入力端子と16個の出力端子が割り付けられています。このレッスンの後半で、それらに変数を割り付けます。

4. S-No.1 (FN-XY16SK)の隣の⊟をクリックします。端子が非表示にされ、⊟の代わりに⊞が表示されます。

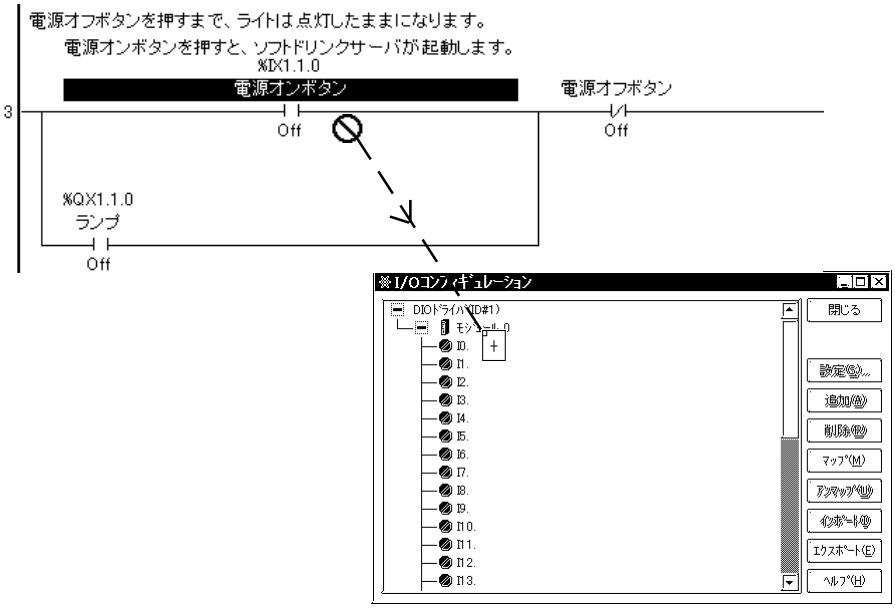
5. Flex Networkドライバは、最大63ユニット(2回線使用時)まで接続できます。2ユニット目の設定も、1ユニット目と同様の設定です。

■ロジックプログラムの変数をクリックし、I/O端子に割り付ける手順

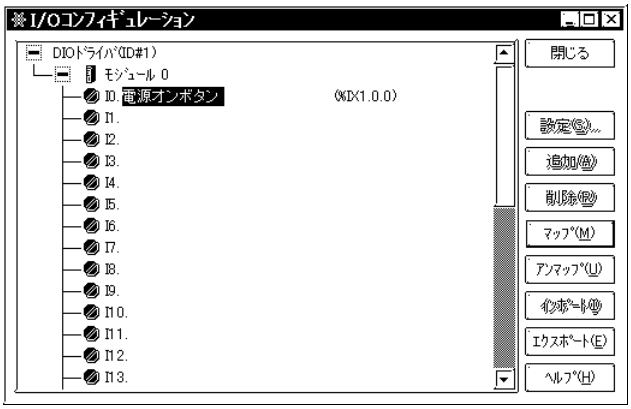
1.「モジュール 0」の隣の⊞をクリックします。下のように、[I/Oコンフィギュレーション]ウィンドウが表示されます。「モジュール 0」では、最初の16個の端子をディスクリート(ビット型)入力のために使用します。



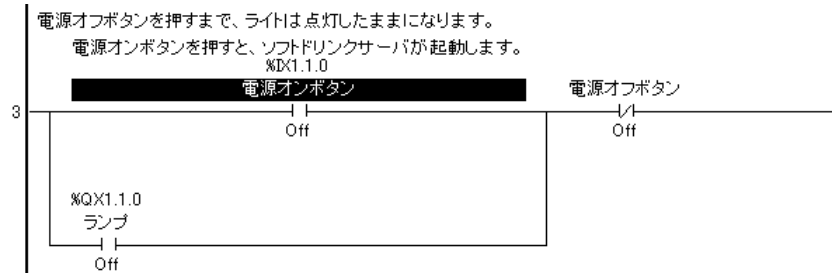
2. ラング3のa接点(NO命令)に設定された変数「電源オンボタン」を選択します。
3.「電源オンボタン」をクリックして、端子I0.にドラッグします。分岐を挿入する場合と同様に、カーソルは最初に⃠になります。カーソルがこの状態のときは、I/O端子に変数を割り付けることはできません。



4. カーソルを端子I0.の上にドラッグし、マウスを離します。変数「電源オンボタン」が端子I0.に割り付けられます。

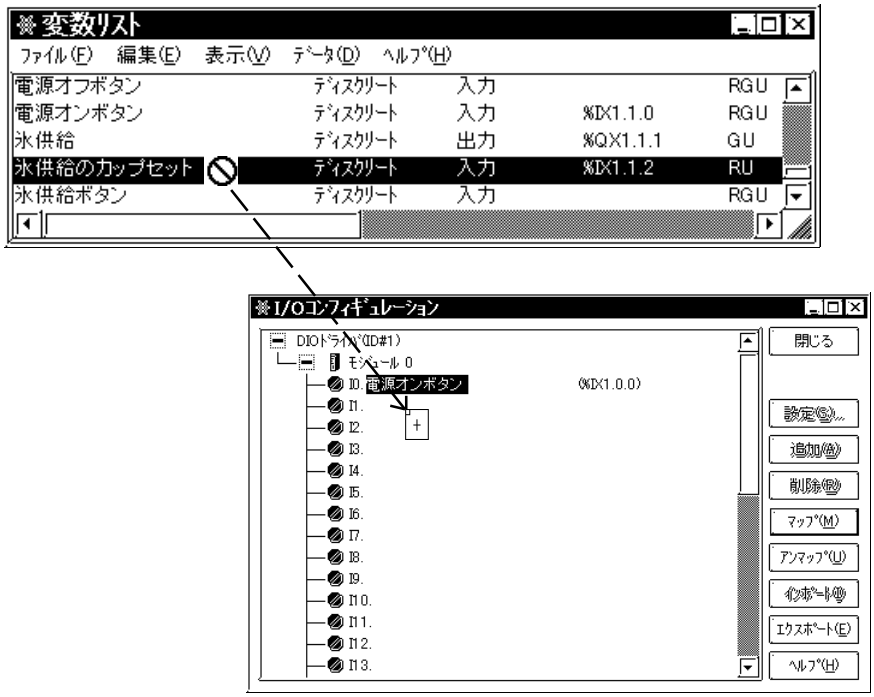


ラング3のNO命令の上の変数「電源オンボタン」の上に、一連の数字および文字が表示されています。これは、変数の実 I/0アドレスです。I/0アドレスはIEC規格に合致したフォーマットで表示されています。



■[変数リスト]ウィンドウから変数をI/0端子に割り付ける手順

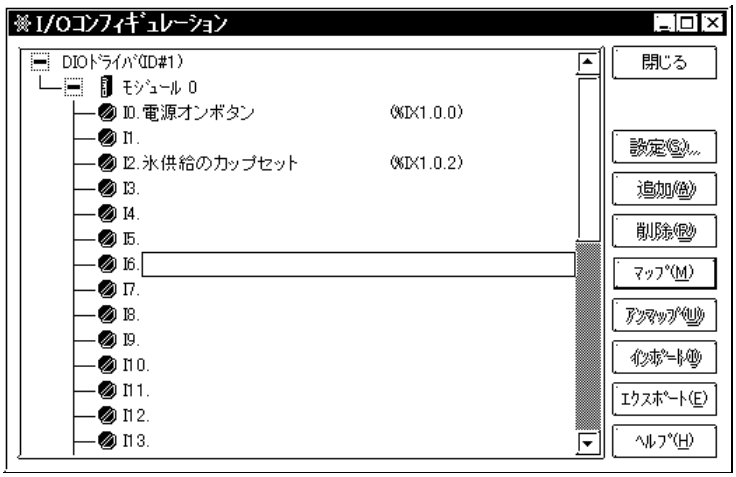
1. [I/0コンフィギュレーション]ウィンドウは、オープンしたまま、[変数リスト]ウィンドウを開きます。
2. 両方のウィンドウが見えるように、2つのウィンドウを配置します。
3. [変数リスト]ウィンドウから、変数「氷供給のカップセット」をクリックし、[I/0コンフィギュレーション]ウィンドウの端末I2にドラッグします。
4. マウスを離します。変数「氷供給のカップセット」が入力端子2に割り付けられました。
   参考：上記の手順を使って、[コメントリスト]ウィンドウからの変数をI/0に割り付けることもできます。



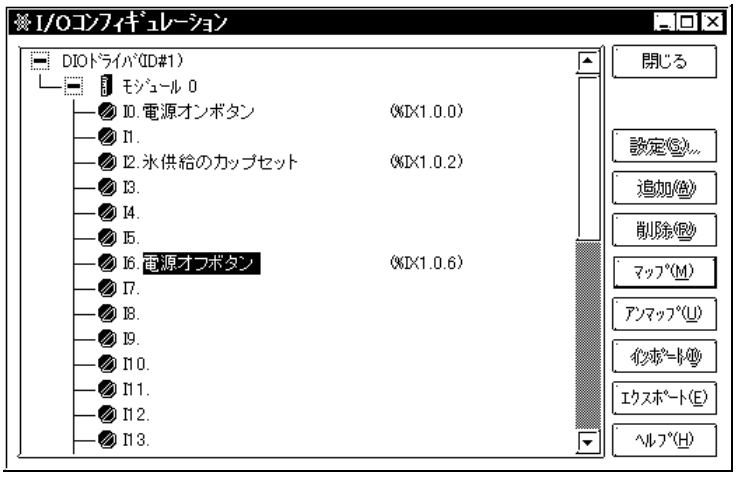
[注意]　[変数リスト]、[コメントリスト]ウィンドウから[I/0コンフィグレーション]ウィンドウに変数をドラッグしてI/0割り付けを行った場合、実I/0の属性が優先され、変数属性が入力 / 出力などに変更されます。

■テキスト入力による割り付け手順

1. 端子I6をクリックします。
2. [Enter]を押します。端子テキストフィールドが起動します。



3.「電源オフボタン」と入力します。
4. [Enter]を押します。「電源オフボタン」が入力端子I6に割り付けられます。

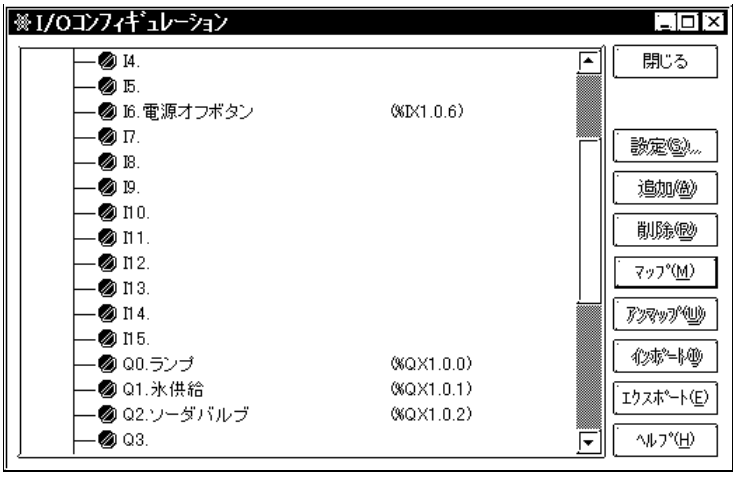


・ テキスト入力によってI/O割り付けを行うと自動的に変数リストにも登録されます。

出力端子に変数を割り付ける方法は、入力端子への変数の割り付けと同じです。今までの手順を使って、以下の表の変数を入力および出力端子に割り付けてください。

変数名	端子タイプ	端子番号
ランプ	出力	Q0
氷供給	出力	Q1
ソーダバルブ	出力	Q2

下のように入力および出力ユニットが、[I/0コンフィギレーション]ウィンドウに表示されます。



1.9.2　[I/0 コンフィギュレーション]ウィンドウの割り付け解除

■[I/0 コンフィギュレーション]ウィンドウの割り付け解除手順

1. [I/0 コンフィギュレーション]ウィンドウの端子 I0 をクリックします。
2. ［アンマップ］をクリックします。「電源オンボタン」の端子 I0. への割り付けが解除されます。その後、この変数を任意の端子にも割り付けることができます。この練習では、この変数を端子 I0 に再び割り付けます。

1.9.3　I/0 に割り付けた変数を命令で使用する方法

新しいプログラムのためにI/0を設定する最も簡単な方法は、変数を直接に端子上に入力する方法です。それらの変数は自動的に作成され、設定され、正しいI/0ポイントに割り付けられます。先にI/0を設定してから、ロジックプログラムを作成する場合、下記の方法で、I/0ポイントを作成します。

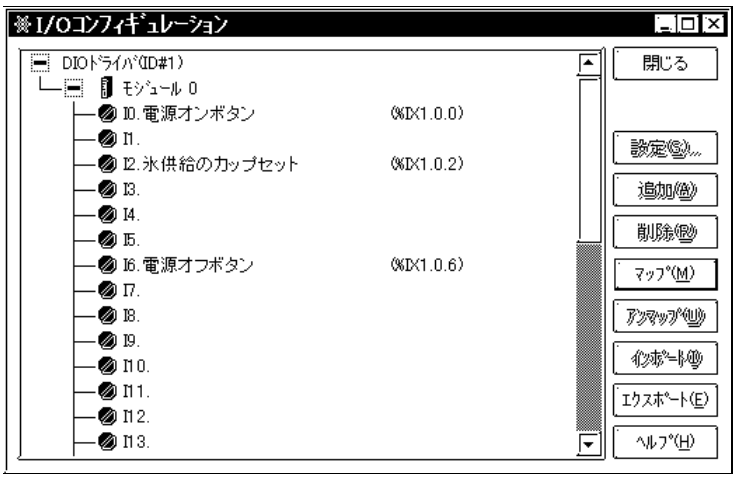
■ I/0 に割り付けた変数を命令で使用する手順

1. 変数をクリックし、I/0端子にドラッグする手順で示した方法を使って、変数をドライバの入力および出力端子に割り付けます。
2. ロジックプログラムを作成します。
3. [I/0 コンフィギュレーション]ウィンドウから変数をクリックし、I/0 を使用する命令にドラッグします。

1.9.4　I/O コンフィギュレーションのコンバート

LT Type A の I/O に割り付けた変数を Flex Network へ自動的にコンバートします。FN-X32TS、FN-XY16SK および FN-XY16SC の入出力 32 点ユニットと、LT Type B+ の本体に内蔵されている I/O（FN-XY16SK）にコンバートできます。

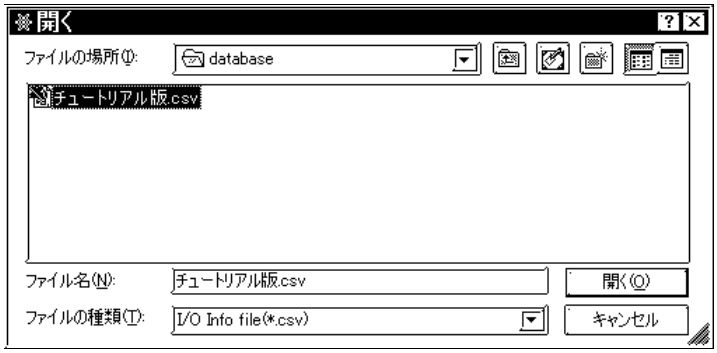
ここでは、LT Type A の I/O から LT Type B+ の本体に内蔵されている I/O（FN-XY16SK）へのコンバートを例にして説明します。



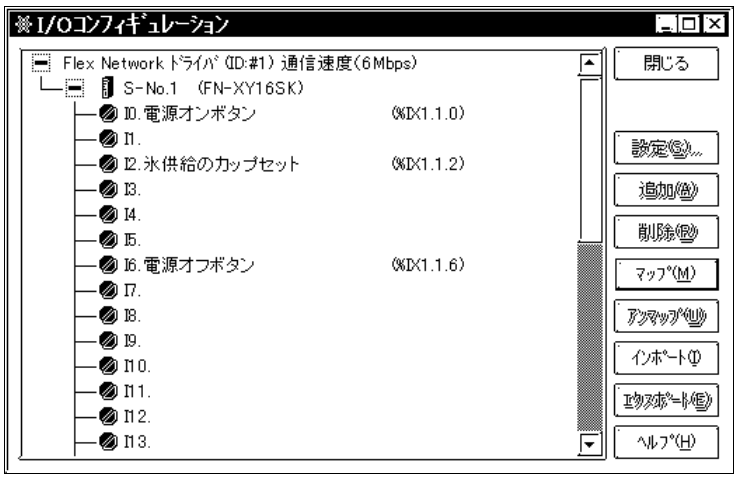
1. [I/O コンフィギュレーション]ウインドウで[エクスポート]ボタンをクリックします。LT Type A の I/O に割り付けられている変数を CSV ファイルに書き出して保存します。

2. プロジェクトマネージャの「新規作成 /LT タイプ」で Type A から Type B+ へ機種変更します。**参照→「1.1 起動のしかた」**

3. ロジックプログラムエディタから[I/O コンフィギュレーション]ウインドウを開きます。[S-No.1(FN-XY16SK)]を選択して、[インポート]ボタンをクリックします。保存したCSVファイルを選択して[開く]ボタンをクリックします。



4. CSV ファイルから変数が取り込まれ、LT Type B+ の本体に内蔵されている I/O（FN-XY16SK）へ割り付けられます。



重要

- **DIO ドライバでの「モジュール 0」の変数は Flex Network ドライバの「S-No.1」へ、「モジュール 1」は「S-No.2」へそれぞれ自動的に割り付けられます。**
  DIO ドライバと Flex Network ドライバの I/O 設定が異なる場合、インポート/エクスポート機能を正常に行うことができません。同じ I/O 設定であることを確認してから使用してください。

まとめ

このレッスンでは、以下の操作を学習しました。

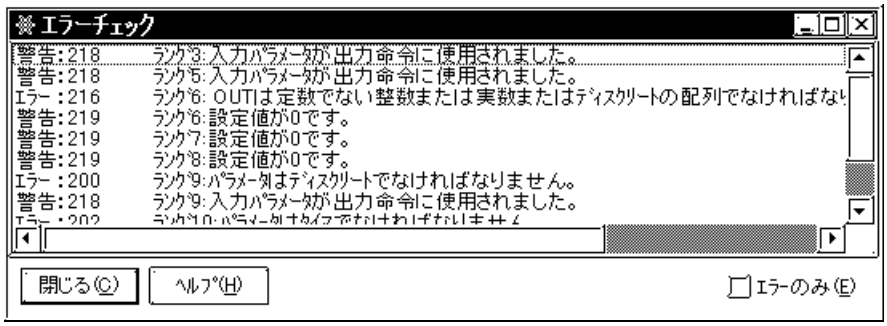
・DIO ドライバの設定
・Flex Network ドライバの設定
・変数の I/O への割り付け

1.10　プログラムエラーチェック

ロジックプログラムをLTに転送して実行する前に、プログラムにエラーがないことを確認しておかなければなりません。そのためにエラーチェックを行います。
エラーの詳細については、**参照→「第4章 エラーと警告」**

■エラーチェック実行手順

[ファイル]メニューから、[エラーチェック]を選択します。[エラーチェック]ウィンドウが表示



[エラーチェック]ウィンドウは、ロジックプログラムエディタがプログラムで検出できるすべてのエラーおよび考えられる問題箇所をリストします。問題箇所は、警告としてリストされます。ウィンドウの右下隅には、[エラーのみ]というマークが付いているチェックボックスがあります。このボックスをチェックすると、ロジックプログラムエディタがプログラムの中で検出したエラーだけが表示され、警告は表示されません。
コントローラは警告を含むプログラムをLTに書き込むことはできますが、エラーを含むプログラムはLTに書き込むことはできません。これらのエラーを修正しなければなりません。

参考：エラーチェックは、ツールバーの  をクリックすることによって実行することもできます。

[エラーチェック]ウィンドウは、エラーおよび警告を、ロジックプログラムで検出された順序に表示します。つまり、ラング1のエラーが最初に表示され、次にラング2以下のエラーが順に表示されます。[エラーチェック]ウィンドウのエラーまたは警告をダブルクリックすると、問題の行を直接表示します。
・ロジックの問題である場合、プログラムのその部分が表示されます。
・I/0割り付けに関する問題である場合、[I/0コンフィギュレーション]ウィンドウが表示されます。

前に説明したように、[エラーチェック]ウィンドウには種々のエラータイプが表示されます。エラーチェックでは、下のようなエラーが示されています。

エラー 200　ラング 9 パラメータはディスクリートでなければなりません

■エラーの修正手順

1. [エラーチェック]ウィンドウのエラー行をダブルクリックします。ラング9の命令のパラメータボックスが反転表示され、それに割り付けられる変数がないことを知らせます。

2. 変数として「ソーダバルブ」と入力します。
   それぞれのエラーおよび警告に関する詳細については、オンラインヘルプまたは「付録A：エラーと警告」を参照してください。
   [エラーチェック]ウィンドウにリストされているエラーの修正が終わったら、もう一度エラーチェックを実行します。まだ残っているエラーが表示されます。エラーをすべて修正した後、プログラムをコントローラに書き込むことができます。

まとめ

このレッスンでは、以下の操作を学習しました。

・ ロジックプログラムのエラーチェックの方法

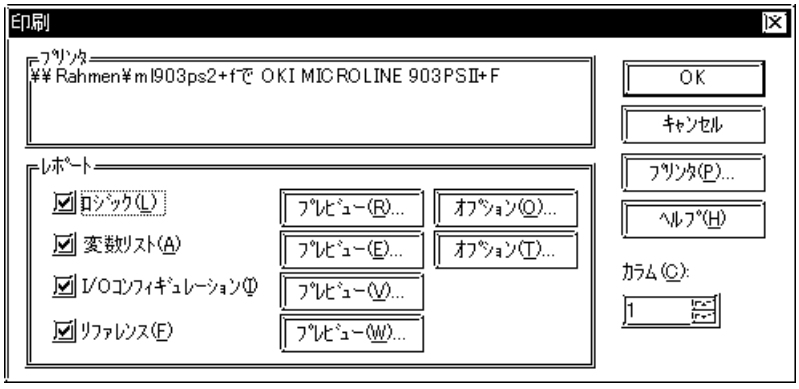
これでプログラムをLTに転送して実行する準備が完了しました。このあとの手順は、「2.1 コントローラの設定」で詳しく説明します。

1.11　ロジックプログラムの印刷

ロジックプログラムエディタでは、ロジックプログラムを印刷することができます。

■ロジックプログラムの印刷手順

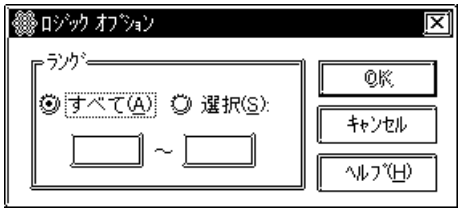
1. [ファイル]メニューから、[印刷]を選択します。[印刷オプション]ウィンドウが表示されます。プレビュー機能で印刷イメージを確認することができます。



レポートを印刷する際のカラムの数(1～4)を選択できます。[レポート]セクションの下に、[ロジック]、[変数リスト]、[I/0コンフィギュレーション]、および[リファレンス]というラベルが付いた4つのチェックボックスがあります。これらのチェックボックスは、ロジックプログラムを印刷するときに、以下のオプションを提供します。

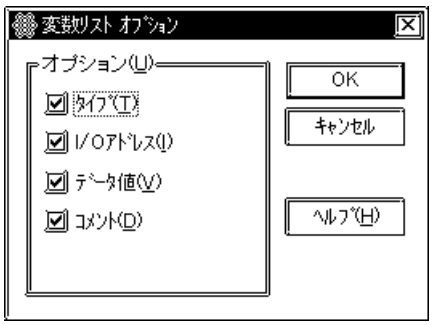
ロジック：

この項目では、ロジックプログラムのラングを印刷できます。その隣の[オプション]をクリックすると、下のようなウィンドウが表示されます。



プログラムのすべてのラングを印刷するには[すべて]を選択し、プログラムの特定の範囲のラングを印刷する場合には[選択]を選択し、その範囲を入力します。[表示]メニューの拡大、縮小で印刷するロジックの大きさを調整できます。

変数リスト：
この項目では、変数リストのオプションが設定できます。目的のチェックボックスをクリックして、変数リストに含める項目を選択します。



変数リストオプション：
このダイアログでは、変数の設定を印刷できます。

オプション	説明
タイプ	変数タイプを表示します。
I/Oアドレス	割り付けられた変数すべてのI/Oアドレスを表示します。
データ値	すべての変数のデータ値を表示します。
属性	保持、グローバルの設定を表示します。
コメント	変数に付加されたコメントを表示します。

参照：このダイアログでは、全変数の全項目を表示するリファレンスを印刷できます。

参考：ツールバーのをクリックすることによっても、プログラムを印刷できます。

まとめ

このレッスンでは、以下の操作を学習しました。
・ロジックプログラムを印刷する

1.12 ロジックプログラムのインポート / エクスポート

ロジックプログラムエディタでは、ロジックプログラムのみをエクスポートし、GLC用ロジックプログラムファイルとして保存することができます。
また逆に、ロジックプログラムファイルをインポートして別のLT用プロジェクトファイル(*.lte) のロジックプログラムとして使用できます。

インポート/エクスポートできるロジックプログラムは、プロジェクトで作成したすべてのロジックプログラムはもちろん、ロジックプログラムの一部分をエクスポートする「部分エクスポート」、ロジックプログラムの一部分をインポートする「挿入インポート」などがあります。

1.12.1 エクスポート

エクスポートできるロジックプログラムは、以下の3種類があります。

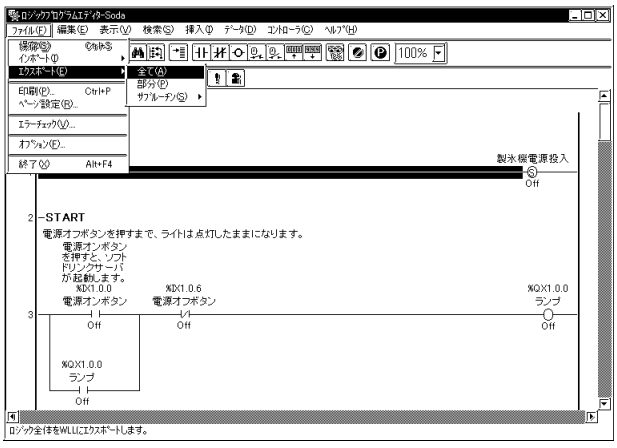
- サブルーチンを含む、すべてのロジックプログラム (*.wll)
- 選択した一部のロジックプログラム (*.wlp)
- サブルーチン部分のロジックプログラム (*.wlf)

■エクスポート手順

上記3種類のエクスポート手順を説明します。

◆サブルーチンを含む、すべてのロジックプログラム

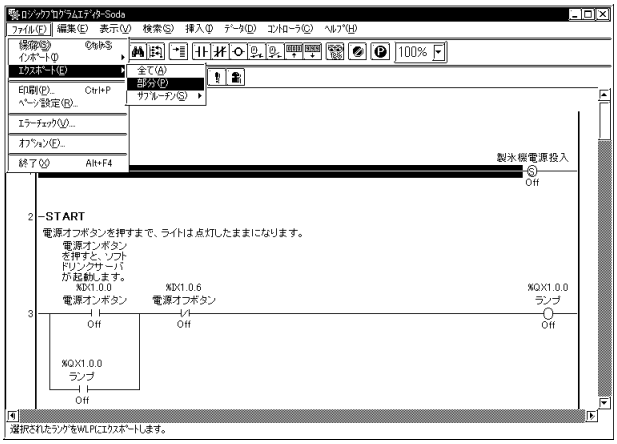
1. [ファイル]メニューから、[エクスポート / 全て]を選択します。



2. [名前を付けて保存]ウィンドウでファイル名を入力します。
3. [保存] をクリックします。
   ロジックプログラムがWLLファイルとして保存されます。

◆選択した一部分のロジックプログラム

1. [ファイル]メニューから、[エクスポート / 部分]を選択します。

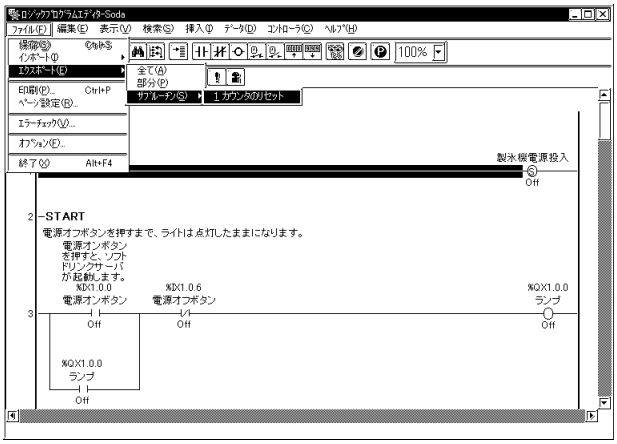


2. エクスポートするラングの範囲を設定し、[OK]ボタンをクリックします。

3. [名前を付けて保存]ウィンドウでファイル名を入力します。
4. ［保存］をクリックします。
   ロジックプログラムがWLPファイルとして保存されます。

◆サブルーチン部分のロジックプログラム

1. [ファイル]メニューから、[エクスポート/サブルーチン]を選択します。[サブルーチン]を選択すると、ロジックプログラムで作成されたサブルーチンの一覧が表示されます。一覧からエクスポートするサブルーチンを選択してください。



2. [名前を付けて保存]ウィンドウでファイル名を入力します。
3. ［保存］をクリックします。
   ロジックプログラムがWLFファイルとして保存されます。

1.12.2　インポート

エクスポートされたロジックプログラムをインポートしてくる方法として、以下の3種類があります。

・サブルーチンを含む、すべてのロジックプログラムをインポートする「上書き」

・選択した一部分のロジックプログラムをインポートする「挿入」

・サブルーチン部分をインポートする「サブルーチン」

サブルーチンを含む、すべてのロジックプログラムをインポートする場合、現プロジェクトのロジックプログラムに”上書き”されますのでご注意ください。

ラングをインポートする位置は、「ファイル / オプション / 編集」で設定することができます。

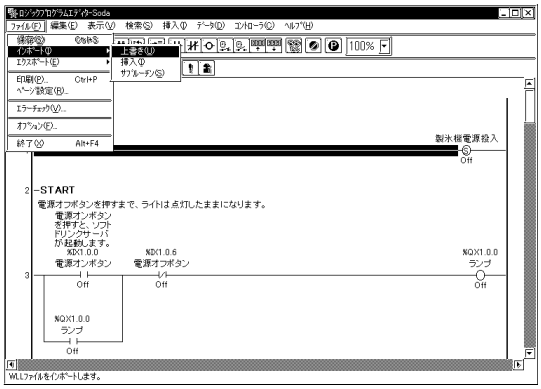
参照→「第1章 ロジックプログラムを作成する前にオプション画面で設定を行う」

■インポート手順

上記3種類のインポート手順を説明します。

◆サブルーチンを含む、すべてのロジックプログラム

1. [ファイル]メニューから、[インポート / 上書き]を選択します。

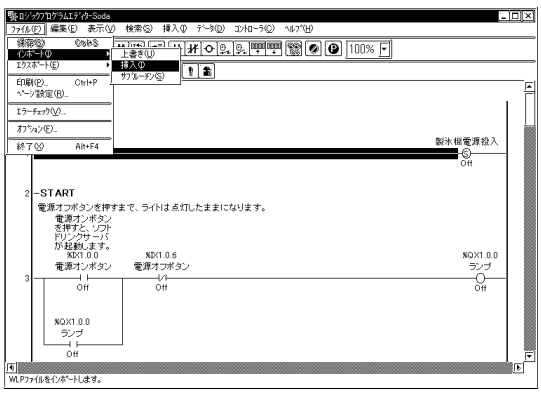


2. [開く]ウィンドウでインポートしたいWLLファイルを選択します。
3. ［開く］をクリックします。
   ロジックプログラムがインポートされ、インポートしたロジックプログラムで使用さている変数が変数リストに登録されます。
4. ロジックプログラムを保存すると、グローバル変数がロジックシンボルとしてシンボルエディタに登録されます。

参照→「オペレーションマニュアル 作画編 4.2.5 シンボルエディタ」

◆選択した一部分のロジックプログラム

1. [ファイル]メニューから、[インポート / 挿入]を選択します。



2. ロジックプログラムを挿入する場所（ラング番号）を指定します。

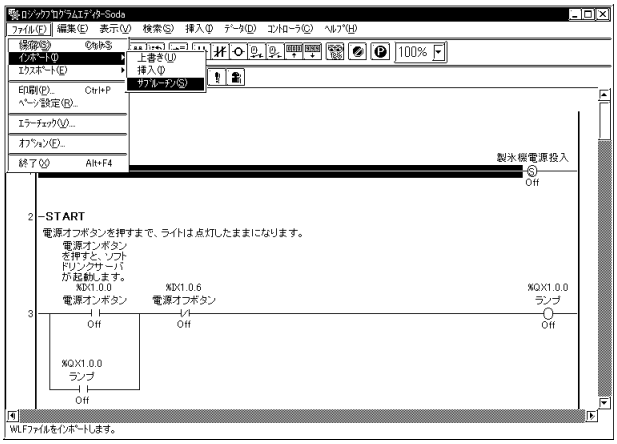
3. [開く]ウィンドウでインポートしたいWLPファイルを選択します。
4. ［開く］をクリックします。
   ロジックプログラムがインポートされ、インポートしたロジックプログラムで使用さている変数が変数リストに登録されます。
5. ロジックプログラムを保存すると、グローバル変数がロジックシンボルとしてシンボルエディタに登録されます。

参照→「オペレーションマニュアル 作画編 4.2.5 シンボルエディタ」

・ インポートするロジックプログラムとインポートされるロジックプログラムに同名の変数が存在する場合、変数タイプはインポートされる側の変数タイプに変更されます。

◆サブルーチン部分のロジックプログラム

1. [ファイル]メニューから、[インポート / サブルーチン]を選択します。



2. [開く]ウィンドウでインポートしたいWLFファイルを選択します。
3. ［開く］をクリックします。
   ロジックプログラムがインポートされ、インポートしたロジックプログラムで使用さている変数が変数リストに登録されます。
4. ロジックプログラムを保存すると、グローバル変数がロジックシンボルとしてシンボルエディタに登録されます。

参照→「オペレーションマニュアル 作画編 4.2.5 シンボルエディタ」

・ インポートするロジックプログラムとインポートされるロジックプログラムに同名の変数が存在する場合、変数タイプはインポートされる側の変数タイプに変更されます。

まとめ

このレッスンでは、以下の操作を学習しました。
・ロジックプログラムをインポート / エクスポートする

1.13　画面プログラムの開発

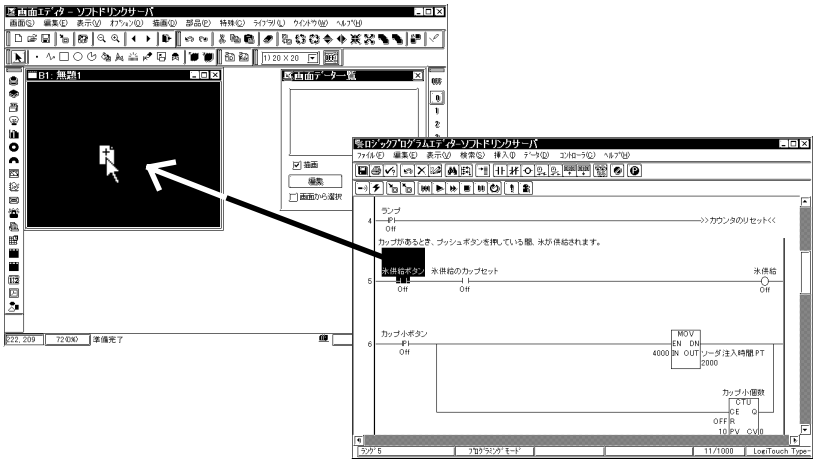
画面エディタで、「氷供給ボタン」、「カップ大ボタン」、「カップ中ボタン」、「カップ小ボタン」を作成します。下図は画面の完成例です。

■画面エディタの起動

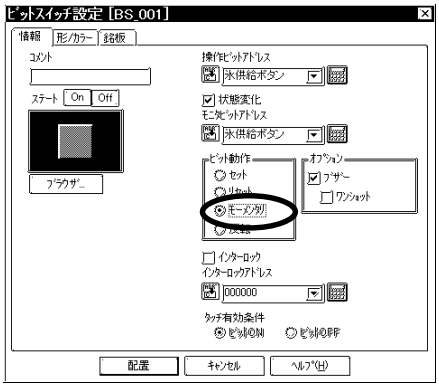
1. プロジェクトマネージャから[作画 / 画面]をクリックし、画面エディタを起動します。
2. メニューバーの[画面/新規作成]をクリックし、[ベース画面]が選択されていることを確認して[OK]ボタンを押します。

■ドラッグ & ドロップによる作画手順

1. ロジックプログラム中の「氷供給ボタン」を選択し、画面エディタへドラッグします。



2. [ビットスイッチ設定]ダイアログボックスが表示されます。[ビット動作]を[モーメンタリ]に設定してください。また、操作ビットアドレスが「氷供給ボタン」になっていることを確認して[配置]してください。



3. 「氷供給ボタン」スイッチの完成です。同様にして「カップ大ボタン」、「カップ中ボタン」、「カップ小ボタン」を作成してください。

MEMO

このページは、空白です。
ご自由にお使いください。

第2章 ロジックプログラムを実行する

エラーのないロジックプログラムが完成したら、LTにダウンロードしてコントローラ(ランタイム)上で実行することができます。
本章では、コントローラの設定方法、プログラムの書き込み方、モニタリングモードでの実行方法について説明します。

2.1 コントローラの設定

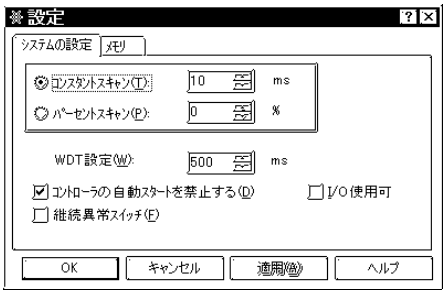
ロジックプログラムをLTに書き込む前に、コントローラが正しく設定されていることを確認してください。LTでプログラムを実行する場合、コントローラの設定にはシステムの設定、メモリという2つの項目があります。

■コントローラの設定手順

[コントローラ]メニューで[設定]を選択します。

◆システムの設定

[システムの設定]タグを選択します。



[システム設定]に設定した値は、コントローラでロジックプログラムを実行する際の設定値になります。設定後にプログラムを実行すると、手動で変更しない限り、コントローラでは常にこの設定が適用されます。設定内容は個別の実行プログラム毎に設定されます。
[システム設定]タブの各項目の内容は次ページの表の通りです。

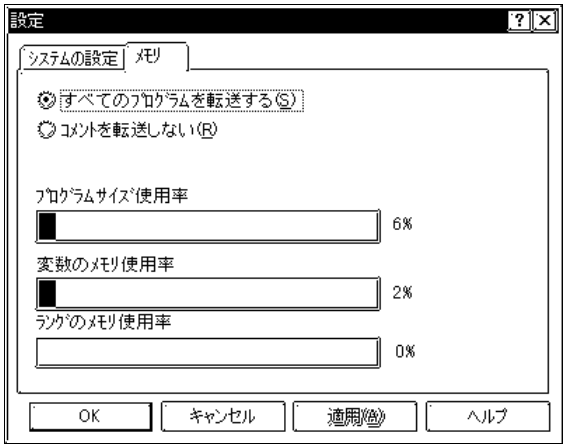
項目	内容
コンスタントスキャン	システム変数 #TargetScanに、スキャンタイムをms単位で指定します。 注：コンスタントスキャンモードでも、ロジックタイムがスキャンタイムの50%を超える場合はコンスタントなスキャンを保証できません。10ms単位で設定します。
パーセントスキャン	システム変数 #PercentAllocに、適切なスキャンタイムを、総処理時間に占める比率で指定します。求められたスキャンタイムの1msの位は切り上げられます。
WDT設定値	ロジックプログラムの異常でスキャンタイムが延び、ここで設定した値を超えてしまうと「メジャー異常」でお知らせします。
コントローラの自動スタートを禁止する	LTのオフラインメニューで”電源ON時の動作モード”を[DEFAULT]に設定した場合のみ本設定は有効になります。*1 電断後にコントローラが起動するとき、コントローラがロジックプログラムを自動的に実行しないことを設定します。システム変数 #DisableAutoStartでも同機能の設定が行えます。 参照→「8.2.22 #DisableAutoStart」
継続異常スイッチ	コントローラにマイナー異常が発生したときに、ロジックプログラムの実行を終了するかどうかを設定します。システム変数 #FaultOnMinorでも同機能の設定が行えます。 参照→「8.2.24 #FaultOnMinor」
I/O使用可	LTの本体やI/Oユニットの外部I/Oへの入出力を可能にする動作です。通常ロジックプログラムのダウンロードをおこなった後、LTを運転状態にしただけでは外部I/Oの入出力をおこなうことができません。これは、操作やロジックプログラムのミスなどで、機会が突然動作することを防止するための安全を優先する考え方の機能です。

・ コンスタントスキャン、パーセントスキャンについては、「第6章 コントローラ機能」を参照してください。
・ システム変数の詳細については、「第8章 システム変数」を参照してください。
・ コントローラのRUN/STOP操作においても「I/O使用可」が選択できます。詳細は、「2.2 コントローラのRUN/STOP」を参照してください。

*1「電源ON時の動作モード」は、「動作環境」→「コントローラメニュー」→「コントローラ設定1」で設定します。ここで、[RUN]または[STOP]に設定した場合、LT Editorの設定は無視され、オフラインでの設定が優先されます。

◆メモリ

[メモリ]タブをクリックします。



- **すべてのプログラムを転送する**
  コメントを含むすべてのロジックプログラムを転送します。
  LTより読み込むとロジックプログラムのコメントを読み込めます。
- **コメントを転送しない**
  LTに書き込むファイルのサイズを小さくする事ができます。コメントを転送しなかった場合は、LTより読み込んでもコメントはありません。
- **プログラムサイズ使用率**
  現在編集中のロジックプログラムの容量が、LTで使用可能なメモリのうち何パーセントを占めているかを示します。
- **変数のメモリ使用率**
  現在登録されている変数の容量が、LTで使用可能なメモリのうち何パーセントを占めているかを示します。
- **ラングのメモリ使用率**
  現在使用中の命令およびラングの容量が、LTで使用可能なメモリのうち何パーセントを占めるかを示します。

2.1.1　コントローラへの書き込み

ロジックプログラムエディタを使ってロジックプログラムを作成しエラーがない状態になったら、コントローラへ書き込み、LTでの実行が可能になります。
LTへのロジックプログラムの書き込みには以下の方法があります。

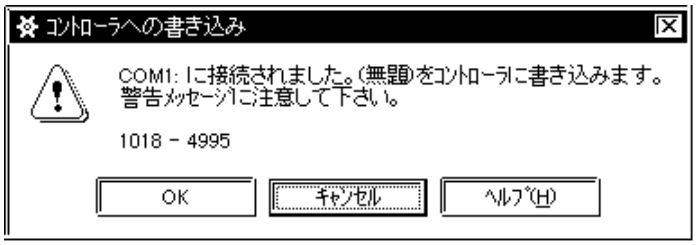
・LT Editorの[転送]ウィンドウで画面データとロジックプログラムを転送する。
・LT Editorの[転送]ウィンドウでロジックプログラムのみを転送する。

ただし、LTへロジックプログラムを書き込む前に、LTをセットアップしておく必要があります。LT Editorの[転送]ウィンドウで、プロジェクトファイルと共にシステムを転送することによりセットアップできます。転送についての詳細は、「オペレーションマニュアル 作画編 第7章 データ転送」を参照してください。

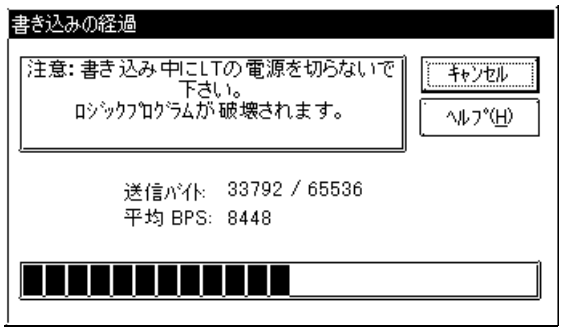
ここではLT Editorでロジックプログラムのみを転送する方法について説明します。

■コントローラへの書き込み手順

1. [コントローラ]メニューで[コントローラへの書き込み]を選択します。[コントローラへの書き込み]ウィンドウが表示されます。コントローラへ書き込むようプロンプトで要求されます。
   コントローラにプログラムが書き込まれる前に、LT Editorは自動的にエラーのチェックを実行します。プログラムにエラーがあると、コントローラへの書き込みはできません。



2. [OK]をクリックします。[ダウンロードの進行状況]ウィンドウが表示されます。LTへのファイルの転送を開始し、進行状況が表示されます。



参考：LTEファイルをコントローラに書き込むと、Flex Networkドライバなどの必要なI/Oドライバがダウンロードされます。ドライバに変更がない場合は、ドライバのダウンロードはスキップされます。

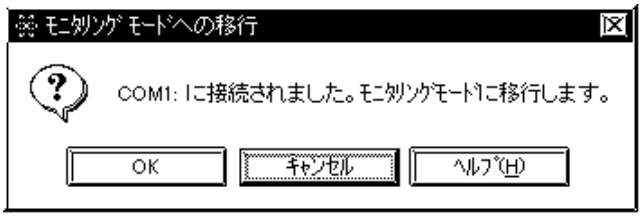
参考：転送する前にコメントを削除すれば、ダウンロードしたファイルのサイズを縮小することができます。**参照→2.1 コントローラの設定**

【注意】**ロジックプログラムの書き込みのあと、LTはリセットされます。**

2.1.2　モニタリングモードへの移行

■モニタリングモードへの移行手順

1. [コントローラ]メニューで、[モニタリングモードへの移行]を選択します。モニタリングモードへ移行するかどうかの確認のダイアログボックスが表示されます。



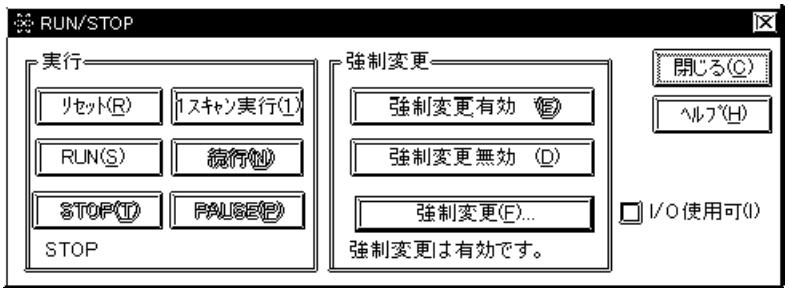
2. [OK]をクリックします。モニタリングモードで接続されたLT上のプログラムのRUN/STOPができます。

2.2　コントローラの RUN/STOP

モニタリングモードの場合、LT Editor側からコントローラのRUN/STOPを切り替えることができます。この時点でコントローラのモニタを実行し始めます。
前述のように、コントローラへモニタリングモードで接続してから、RUN/STOPやモニタリングモードの機能を実行してください。

■コントローラの RUN／STOP 手順

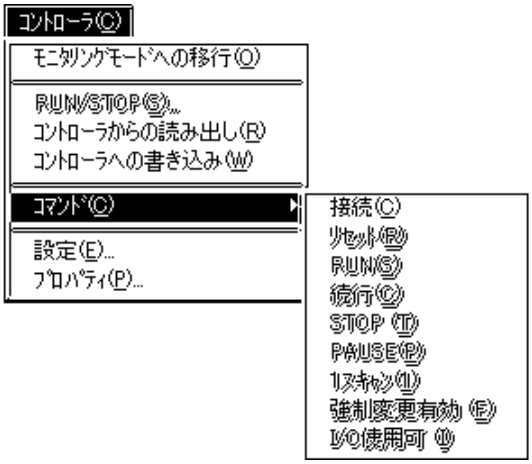
1. [コントローラ]メニューで、[RUN/STOP]を選択します。プログラミングモードでは、この項目は使用できません。[RUN/STOP]ダイアログボックスが表示されます。



項目	内容
RUN	[RUN]ボタンをクリックするとコントローラによるプログラム実行が開始されます。最初にプログラムのスキャンが始まり、すべてのラダープログラムが順に実行されます。最初のスキャンで初期設定プログラムが実行されます。
STOP	[STOP]ボタンをクリックするとコントローラが終了します。
リセット	[リセット]ボタンをクリックすると、コントローラが実行ファイルを再ロードし、I/Oおよび変数を初期化します。
１スキャン実行	[1スキャン実行]ボタンをクリックすると、プログラムのスキャンが1回行われます。この機能はアプリケーションのトラブルシューティングやデバックに有効です。
PAUSE	[PAUSE]ボタンをクリックすると、コントローラはプログラムをスキャンするのを一時停止しますが、I/Oリフレッシュは実行されます。
続行	続行は[PAUSE]ボタンをクリックするとコントローラは、現在のデータ値を使用してプログラムの実行を再開します。
強制変更有効	強制変更を有効にします。
強制変更無効	強制変更を無効にします。
強制変更	ロジックプログラムで強制変更に登録したすべての変数をリストアップします。
I/O使用可	LTの本体やI/Oユニットの外部I/Oへの入出力を可能にする動作です。通常ロジックプログラムのダウンロードをおこなった後、LTを運転状態にしただけでは外部I/Oの入出力をおこなうことができません。これは、操作やロジックプログラムのミスなどで、機会が突然動作することを防止するための安全を優先する考え方の機能です。

MEMO
- RUN/STOP の移行時に内部的にI/O 使用可のチェックをおこなっています。よってRUN状態で「I/O使用可」を変更しても反映されません。一端、必ずSTOPにして「I/O使用可」の変更を行い、RUNに戻してください。

[ｺﾝﾄﾛｰﾗ(C)]メニューおよびツールバーより選択することもできます。



	**接続**
	**モニタリングモードへの移行**
	**コントローラへの書き込み**
	**コントローラからの読み出し**
	**リセット**
	**RUN**
	**続行**
	**STOP**
	**PAUSE**
	**1ｽｷｬﾝ**
	**強制変更有効**
	**I/O使用可**

参考：リセットにより、保持型の変数はリセットされます。特別な初期設定が必要な場合は、ロジックプログラム上でMOV命令などを使って初期値を設定してください。

2.3 システム変数によるプログラムのトラブルシューティング

コントローラの起動後にアプリケーションがうまく動作しない場合、システム変数を使用してアプリケーションのトラブルシューティングを実行することができます。システム変数#Fault、#IOFault、#IOStatus、#ScanCountは、コントローラかI/Oに発生するトラブルの検出に最も有効です。

項目	内容
#FaultCode	最新のエラー状況を示します。立ち上がり一回目のスキャン実行時に、「0」にリセットされます。
#Faultrung	エラーが発生したラングの番号が表示されます。
#IOFault	I/Oドライバでエラーが検出されるとONになります。
#IOStatus	I/O固有のエラーを表示する配列です。これらのエラーには、ドライバごとに異なるエラーコードがつけられます。エラーの詳細については、ドライバのオンラインヘルプを参照してください。#IOFaultがONになった場合にのみ、エラーが#IOStatusで表示されます。
#ScanCount	コントローラが起動してから実行したスキャン回数を示します。この変数は、モニターを開始してから常に増加しています。増加していない場合は、コントローラが動作していません。

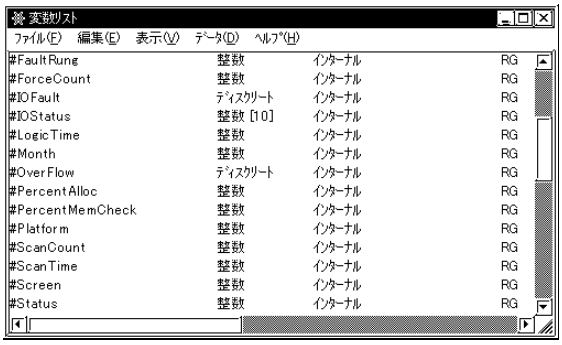
システム変数の詳細については、「第8章 システム変数」を参照してください。

2.4 システム変数の表示

システム変数を表示させて、I/Oの状態、スキャン時間、コントローラの状態に関する情報を見ることができます。**参照→「第8章 システム変数」**

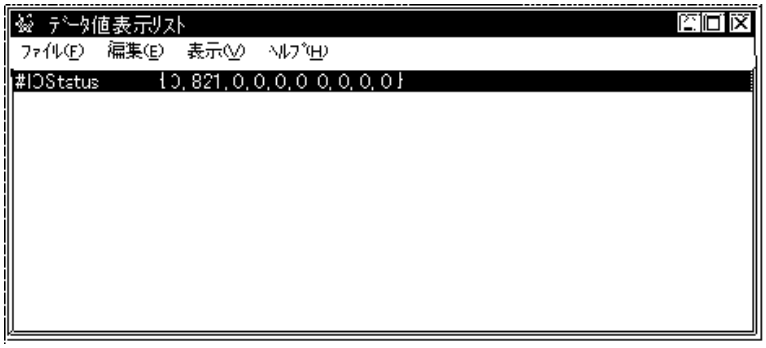
■システム変数の表示手順

1. [データ]メニューで[変数リスト]を選択します。[変数リスト]ウィンドウが表示されます。このウィンドウにはLT Editorのシステム変数（#で始まる変数）がすべて表示されます。表示されない場合は、[表示]メニューで[システム]を選択してください。



2. [データ]メニューで[データ値表示リスト]を選択すると、[データ値表示リスト]ウィンドウが表示されます。
3. モニタするシステム変数をクリックして、[変数リスト]ウィンドウから[データ値表示リスト]ウィンドウへドラッグします。

ロジックプログラムのスキャン中にエラーが発生した場合は、モニターした変数が当該のエラーを表示します。以下は、ドライバ1でI/Oエラー821が発生した例です。



2.5 コントローラからの読み出し

LT本体にあるロジックプログラムの編集や保存を行いたい場合はコントローラから読み出します。
LTからロジックプログラムを読み出すには以下の方法があります。

・LT Editorの[転送]ウィンドウで画面データとロジックプログラムを受信する。
・LT Editorの[転送]ウィンドウでロジックプログラムのみを受信する。
・ロジックプログラムエディタでロジックプログラムのみを受信する。

ここではLT Editorでロジックプログラムのみを転送する方法について説明します。
LT Editorの[転送]ウィンドウで受信する方法については、「オペレーションマニュアル 作画編 第7章 データ転送」を参照してください。

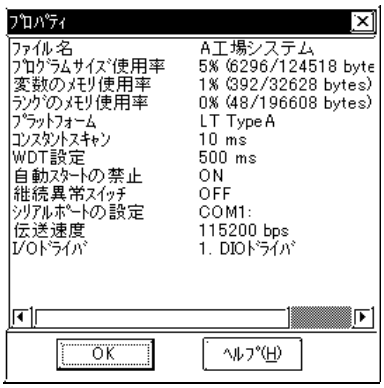
■コントローラからの読み込み手順

1. コントローラがモニタリングモードで接続されている場合は、[コントローラ]メニューで[プログラミングモードへの移行]を選択します。
   【重要】
   プログラムに初期化していない値が含まれている場合、プログラムから読み出す前に、コントローラを終了してください。
2. [コントローラ]メニューで、[コントローラからの読み出し]を選択します。コントローラに書き込まれたプログラムがLT Editorで開かれます。これでプログラムに変更を加えたり、「*.lte」ファイルとして保存できるようになります。

2.6 プロパティ

コントローラメニューの[プロパティ]を選択するとLTのプログラム情報がリストボックスで表示されます。

以下はプログラム情報のプロパティボックスです。



MEMO

このページは、空白です。
ご自由にお使いください。

第3章 モニタリングモードでの動作確認

コントローラで実行しているプログラムにモニタリングモードで変更を加え、その変更内容をすぐに実施することができます。
本章での解説や例示では、C:\Program Files\Pro-face\LT\SAMPLEにあるSoda.lteを使用します。このファイルでは、ラダーのカラーやオプションをシステムのデフォルト値に設定しているものとします。

3.1 編集を始める前に

■ロジックプログラムの実行手順

1. Soda.lteを開きます。このファイルはサンプルプログラムとして、LT Editorに添付されています。
   デフォルト設定のパスはC:\Program Files\Pro-face\LT\SAMPLEです。
2. コントローラにプログラムを書き込みます。
3. コントローラをモニタリングモード接続します。
4. コントローラをRUNします。コントローラの操作の詳細は、参照→**第2章 ロジックプログラムグラムを実行する**

■モニタリングモードでLTに加えられるプログラムの変更

コントローラにモニタリングモードで接続中、以下の変更をプログラムに加えることができます。

・変数の状態の変更
・変数の値の変更

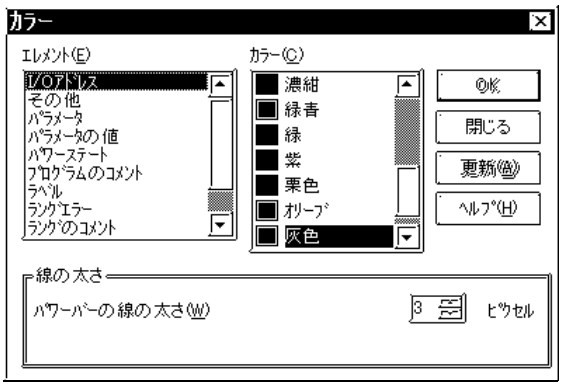
3.2 モニタリングモード編集にカラーを使用する

ロジックプログラムエディタでは、デフォルト設定のカラーを使用して、特定の状態を示したり、モニタリングモードの動作中にロジックプログラムに変更を加えることができます。
デフォルト設定のカラーは以下のとおりです：

項目	内容
緑	回路が導通しています。
赤	ラングにエラーが発生していることを示します。
紫	オンラインエディットしていることを示します。

■ロジックプログラムエディタのカラーのデフォルト設定の変更手順

1. [表示]メニューで[カラー]を選択します。[カラー]ダイアログボックスが表示されます。



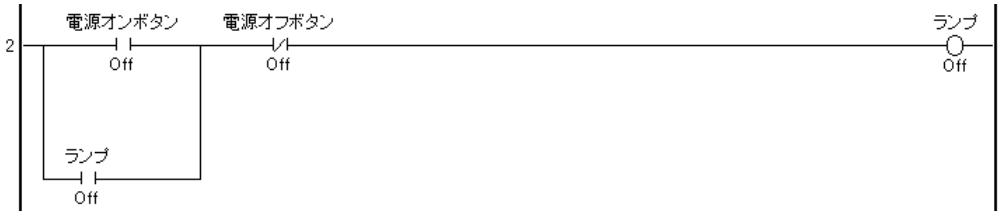
2. 変更したい[エレメント]と[カラー]を選択して、[更新]をクリックします。

3.3 ディスクリート変数を ON/OFF する

ロジックプログラムの実行中は、ディスクリート変数を手動でON／OFFすることができます。強制ONにした場合はON状態が保持されますが、手動でディスクリートをONにした場合スキャンが行われると、ディスクリート変数の状態がプログラムの影響を受けるため、プログラムに依存します。

■ディスクリート変数の ON ／ OFF 方法

1. ラング2の出力コイルに割り付けた変数「ランプ」をマウス右ボタンでクリックします。
2. ショートカットメニューで[ONにする]を選択します。変数「ランプ」がONになり、ラングが導通します。



3. ラング2の出力コイルに割り付けた変数「ランプ」をマウス右ボタンでクリックします。
4. ショートカットメニューで[OFFにする]を選択します。「ランプ」変数がOFFになり、導通状態はクリアされます。

参考：[オプション]ダイアログボックスの[モニタ]タブで「パワーフロー」が選択されていないと、導通状態はロジックプログラムに表示されません。**参照→「第1章 ロジックプログラムを作成する前にオプション画面で設定を行う」**

3.4 ディスクリート変数を強制的にON/OFFする

コントローラにモニタリングモードで接続している間は、ディスクリート変数を強制的にON/OFFすることができます。ディスクリート変数を強制的にON/OFFすると、強制ON/OFFを手動で解除するまでは状態が保持されます。整数変数、文字型変数のビット指定の状態は変更できません。3.3のディスクリート変数のON/OFFの場合は、プログラムの演算結果に依存しましたが、強制ON/OFFの場合は、プログラムの演算結果に依存しません。

■ディスクリート変数の強制的なON／OFF方法

1. ラング9の出力コイルの変数「ソーダバルブ」を右クリックします。
2. ショートカットメニューで[強制ON]を選択します。
3. [強制変更]ウィンドウで[OK]をクリックします。

変数がONとなり、プログラムではOFFになりません。

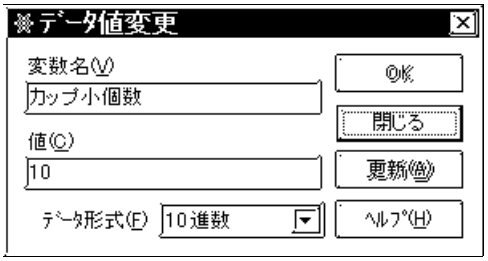
参考：強制ON/OFFを指定したのにロジックプログラムに反映されない場合は、強制変更無効の設定になっています。有効にするには、[RUN/STOP]ダイアログボックスで[強制変更有効]ボタンをクリックするか[コントローラ]メニューおよびツールバーより切り替えることができます。デフォルトは、強制変更有効となっています。

3.5 変数値の変更

コントローラにモニタリングモードで接続している間は、ロジックプログラムに含まれる変数の値を設定することができます。

■変数値の変更手順

1. [データ]メニューで[データ値変更]を選択すると、[データ値変更]ダイアログボックスが表示されます。
2. ロジックプログラムの変数「カップ小個数」をクリックします。下のような[データ値変更]ダイアログボックスが表示されます。



3. [値]フィールドを選択してから、5を入力してください。
4. [更新]をクリックします。「カップ小個数」の値が5になりました。他の値を変更するか、または[閉じる]をクリックして[データ値変更]ウィンドウを閉じます。

参考　・ データ値は、10進数、16進数、8進数、2進数のいずれかのフォーマットで入力できます。[データ表示リスト]で選択することもできます。

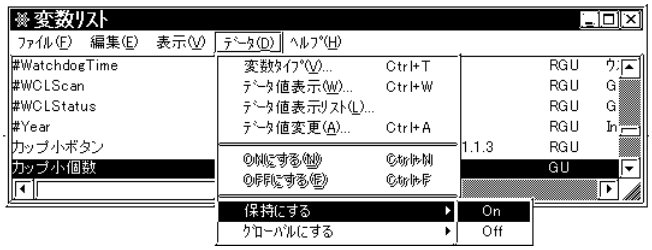
・ [データ値変更]ダイアログボックスとともに、[変数リスト]ウィンドウか[データ値表示リスト]ウィンドウを使用すれば、変数をすばやく検索して設定することができます。

3.6 変数の属性変更

変数リストの[データ]メニューに変数の保持､グローバルの属性を変更するメニューがあります。このメニューはプログラミングモードのみ有効となります。

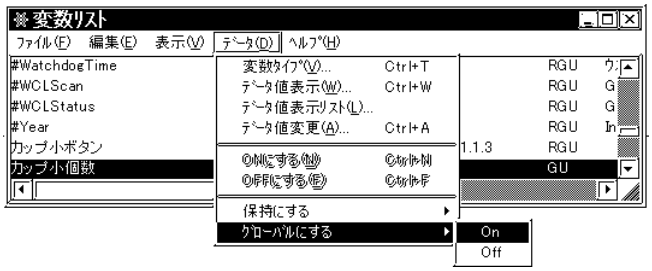
■変数(保持)の属性変更手順

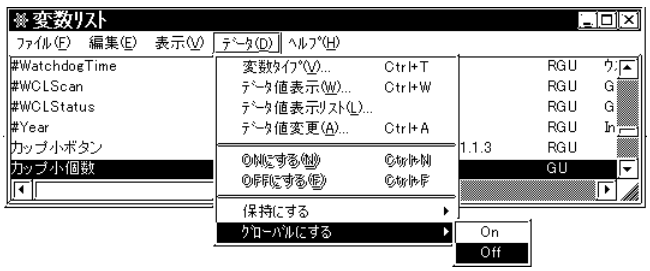
1. [データ]メニューで[変数リスト]を選択すると､[変数リスト]ウィンドウが表示されます。変更したい変数を選択し、属性を変更します。システム変数の保持は変更できません。



■変数(グローバル)の属性変更手順

1. [データ]メニューで[変数リスト]を選択すると､[変数リスト]ウィンドウが表示されます。変更したい変数を選択し、属性を変更します。

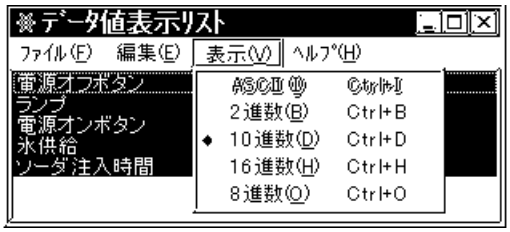




3.7 データ値表示リスト

■表示モード一括変更

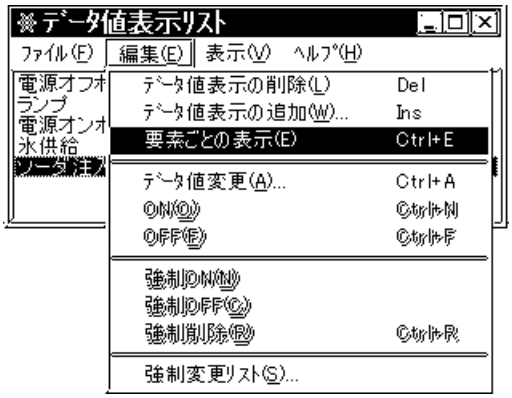
[データ]メニューから[データ値表示リスト]を選択します。選択した変数の表示モードを一括で変更が可能です。



■配列の要素ごとの表示

データ値表示リストで配列の場合は、[要素ごとの表示]で配列タイマ、カウンタの値を要素ごとに表示します。

1. [データ]メニューで[変数リスト]を選択し、[データ]メニューから[データ値表示リスト]を選択します。
2. [データ値表示リスト]メニューから[要素ごとの表示]を選択します。



第4章 エラーと警告

プログラムでエラーチェックが行われると、[エラーチェック]ウィンドウにエラーや警告が表示されます。これらのエラーや警告は、プログラムのロジック、変数、I/0などに起こる問題に関するものです。エラーの原因ごとにエラーコードが割り付けられています。このコードの値を参考にすると、エラーや警告の発生原因が特定できます。

200-299：ロジックのエラーと警告

ロジックプログラムの命令に関する情報については、メインウィンドウで選択してから[ヘルプ]メニューで[関係トピックヘルプ]を選択して[F1]キーを押してください。

◆エラー 200
パラメータはディスクリートでなければなりません。
命令にディスクリートのオペランドが必要です。以下のいずれかです：
ディスクリート値
ディスクリート配列の要素
整数値のディスクリート要素

◆エラー 201
パラメータはカウンタでなければなりません。命令にカウンタの値が必要です。

◆エラー 202
パラメータはタイマでなければなりません。命令にタイマの値が必要です。

◆エラー 203
・・・は整数または実数でなければなりません。命令に、変数または定数として整数または実数が必要です。

◆エラー 204
・・・は定数でない整数または実数でなければなりません。命令に整数または実数の変数が必要です。命令を定数にすることはできません。

◆エラー 205
・・・は整数でなければなりません。命令にデータ値が整数である変数または整数の定数が必要です。

◆エラー206

・・・は整数でなければなりません。しかし配列は許されません。命令にデータ値が整数である変数または整数の定数が必要です。命令を配列にすることはできません。

◆エラー207

・・・は定数でない整数でなければなりません。命令に整数の変数が必要です。命令を定数にすることはできません。

◆エラー208

パラメータがラベルでなければなりません。命令にラベル名が必要です。そのラベル名が存在しなければなりません。

◆エラー209

パラメータがサブルーチンでなければなりません。命令にサブルーチン名が必要です。

◆エラー210

ラベルが範囲外のところで使用されています。指定したラベルは存在しますが、有効範囲外です。

◆エラー211

サブルーチンは自分自身をコールできません。サブルーチンの中でJSR命令で同じサブルーチンをコールします。

◆エラー212

・・・は・・・と同じタイプでなければなりません。2個のパラメータが同じタイプ（整数、実数など）でなければなりません。

◆エラー213

・・・は・・・と同じサイズでなければなりません。2個のパラメータが同じサイズでなければなりません。つまり2個のパラメータはどちらも、「同じ数の要素をもつ配列」または「非配列」でなければなりません。

◆エラー214

XはYと同じサイズまたは、整数でなければなりません。2個のパラメータが同じサイズであるか、または2番目のパラメータの方が大きいサイズのように見なされる整数でなければなりません。

◆エラー215

Xは整数または実数またはディスクリートの配列でなければなりません。命令に変数、定数、または配列として、整数、実数、またはディスクリートが必要です。

◆エラー216

・・・は定数でない整数または実数、またはディスクリート配列でなければなりません。命令に変数または完全な配列として、整数、実数、またはディスクリートが必要です。定数にすることはできません。

◆警告217

両方のパラメータは定数です。命令が2個の定数を比較しています。

◆警告218

入力パラメータが出力命令に使用されました。変数は入力にマークをしているのに（[変数タイプ]ウィンドウを参照）、出力命令で使用されています。I/O割り付けを確認してください。

◆警告219

設定値が”０”です。

◆警告220

設定値が”０”です。

◆警告 224

パラメータに保持型のデータを使用できません。命令のパラメータに割り付けた変数に保持型データを使用できません。

◆警告 225

・・・は整数配列でなければなりません。命令のパラメータに割り付ける変数は、整数配列でなければなりません。

◆エラー 250

重複ラベルは使用できません。同じラベルが、2回以上定義されています。プログラムの別の部分であっても重複はできません。

◆警告 251

空サブルーチン実行上意味がありません。サブルーチンにラングがありません。

◆警告 252

空ラングは実行上意味がありません。ラングに命令がありません。空のラングの修正を選択しないと、プログラムに効果が生じません。

◆警告 253

空分岐は実行上意味がありません。分岐に命令がありません。空の分岐の修正を選択しないと、プログラムに効果が生じません。

◆エラー 254

出力、演算などの制御命令がラングの最後になければなりません。命令の右側に何もありません。

◆警告 255

・・・が複数のタイマで使用されています。同一のタイマ変数が複数のタイマ命令で使用されています。結果は不定となります。[リファレンス]ダイアログボックスを使用して他のタイマ命令を検索し変数名と変更してください。

◆エラー 256

・・・が複数のカウンタで使用されています。同一のカウンタ変数が複数のカウンタ命令で使用されています。結果は不定となります[リファレンス]ダイアログボックスを使用して他の命令を検索し変数名を変更してください。

◆エラー 257

ラング上の最後の命令は出力命令でなければなりません。命令が出力命令になっていません（したがってパラメータの値が変わりません）。

◆エラー 258

複数の出力命令を使用できません。出力命令は右側に他の命令をもつことはできません。

◆エラー 259

分岐の最後の命令は出力命令でなければなりません。出力命令は右側に他の命令をもつことはできません。

◆エラー 260

ネスティングのレベルが最大を超えました。ラングの分岐レベルが多すぎます（レベルの最大数は25です）。ラングを小さくいくつかに分割してみてください。

◆エラー 262

プログラムが大きすぎます。LTのフラッシュメモリの使用可能な容量を超えています。

◆警告 263

・・・が複数のコイルで使用されています。同一変数が複数のコイルで使用されています。ロジックプログラムの処理順で最後に変数が割り付けられた命令の結果が有効になります。

◆エラー 269

ラングのメモリ使用率が・・・%超えています。

◆エラー 270

ラベルの最大数を超えました。最大数は 2048 です。

◆エラー 271

変数の最大数を超えました。最大数は 8192 です。

◆エラー 272

定数の最大数を超えました。

◆エラー 273

NT 命令および PT 命令の最大数を超えました。最大数は 2048 です。

◆エラー 274

PID 命令の最大数を超えました。最大数は 100 です。

300-399：変数のエラーと警告

◆警告 300

変数タイプは入力または出力ですがI/0アドレスが割り付けられていません。変数は入力または出力にマークをしているのに（[変数のタイプ]ウィンドウを参照)、I/0 にマップされません。

◆エラー 301

タイプが設定されていません。変数に変数タイプが割り付けられていません。変数タイプを割り付けるには、[変数タイプ]ウィンドウを使用してください。

◆エラー 302

ラベルが見つかりません。存在しないラベルがジャンプサブルーチン(JSR)命令の飛び先として指示されています。

◆エラー 303

参照される変数はタイマまたはカウンタでなければなりません。タイマまたはカウンタの変数の要素を指定したにもかかわらず、別のタイプの変数になっています。[変数タイプ]ウィンドウを参照してください。

◆エラー 304

変数タイプは整数でなければなりません。変数を使用して配列の要素かまたは修飾語を指定しました。この変数は整数でなければなりません。[変数タイプ]ウィンドウを参照してください。

◆エラー 305

配列変数でない変数に配列参照されました。配列の要素を指定しましたが、その変数が配列として指定されていません。[変数タイプ]ウィンドウを参照してください。

◆エラー 306

配列の範囲を越えて参照しています。配列のサイズと等しいかそれ以上の定数を使用して、配列の要素を指定しました。（有効な要素には 0 からサイズ -1 の番号がついています。）サイズは[変数タイプ]ウィンドウで変更できます。

◆エラー 308

修飾参照が範囲を超えています。範囲を超えたビット、バイト、ワードの要素を指定しています。

◆エラー 309

変数の参照が正しくありません。カウンタの変数にタイマの参照を指定したり、その逆を指定しています。

◆警告 310

・・・がすでに存在します。置き換えできません。その名前の変数はすでに存在しています。[変数のインポート状態]ウィンドウで[OK]をクリックすると、元からあった変数は新しい変数に置き換えられます。

◆エラー 311

クリップボードのバッファのフォーマットが認識されません。現在のクリップボードの中身は、[変数リスト]ウィンドウへの貼り付けには適していません。

◆エラー 312

警告が多すぎます。[変数のインポート状態]ウィンドウでは、一定の数の警告しか表示しません。このメッセージが表示されたら、他にも表示されていない警告があるということです。

◆警告 313

・・・の右括弧 "] " がありません。配列には、[]で囲んだサイズが必要です。たとえば、整数[10]など。

◆警告 314

配列のサイズは・・・で無効です。サイズ1が使用されます。この変数は配列を意図しているようですが、サイズが認識されません。サイズは、整数[10]のように、[]で囲んだ整数にしなければなりません。

◆警告 315

・・・に未知の変数・・・があります。変数タイプは未定義となります。この変数はロジックプログラムエディタの変数タイプとして認識されません。以下の原因が考えられます：

・スペル間違い

・先頭か末尾にブランクがある

◆警告 316

・・・に未サポートの配列・・・があります。設定した配列を無視します。この変数では配列は設定できません。

◆エラー 317

変数名が無効です . . . 変数名として無効な文字を入力しました。

◆エラー 318

エラー数が多すぎます。

◆エラー 320

I/0変数が多すぎます。

◆エラー 321

変数が多すぎます。変数の数を減らしてください。

400-499：I/0のエラーと警告

◆エラー 400

変数名がすでにマップされています。この変数は複数のI/0ポイントにマップされています。[I/0のコンフィギュレーション]ウィンドウを参照してください。

500-549：一般的な I/O ドライバのエラー

◆エラー 500

LTE ファイルが破損しているか、LTE ファイルのダウンロード中に障害が発生した可能性があります。

◆エラー 501

I/O ターミナルに割り当てられている変数に、インターナル（内部）タイプの変数が割り当てられています。入力もしくは出力タイプに変更してください。

◆エラー 502

出力ターミナルに割り当てられている変数に、入力タイプの変数が割り当てられています。出力タイプに変更してください。

◆エラー 503

入力ターミナルに割り当てられている変数に、出力タイプの変数が割り当てられています。入力タイプに変更してください。

◆エラー 504

整数ターミナルに割り当てられている変数に、ディスクリートタイプの変数が割り当てられています。整数タイプに変更してください。

◆エラー 505

ディスクリートターミナルに割り当てられている変数に、整数タイプの変数が割り当てられています。ディスクリートタイプに変更してください。

◆エラー 506

ドライバがコントローラの変数を確認しないときに表示されます。

◆エラー 507

ターミナルに変数が割り当てられていないときに表示されます。

◆エラー 508

ドライバでサポートされていない LT タイプが選択されたときに表示されます。

600-799：PID 命令のエラー

◆エラー 600

コントロールブロック変数は、要素数 7 個以上の整数配列型に設定してください。

◆エラー 601

PID のパラメータは、整数型に設定してください。

800-899：特定の I/O ドライバのエラー

I/O ドライバに関係するエラーについて詳しくは、I/O ドライバのオンラインヘルプを参照してください。

900-1000：特定の I/O ドライバの警告

I/O ドライバに関係する警告について詳しくは、I/O ドライバのオンラインヘルプを参照してください。

第5章 用語集

◆16進数

16が1単位となる整数の表記法です。この値は、前に16#をつけて入力することができます。たとえば、16#FFは255になります。

◆I/O

Input（入力）／Output（出力）。コントローラは、（株）デジタル製I/Oユニットやサードパーティの提供するI/Oユニットにより現実のデバイスに接続されます。

◆I/Oアドレス

I/Oモジュールに割り付けるときのアドレス値です。I/Oアドレスのフォーマットは、割り付けるドライバによって異なります。

◆IEC61131-3

国際電気標準会議（IEC）の制定した標準で、命令リスト（IL）、ラダーロジック図（LD）、機能ブロック図（FBD）、構造化テキスト（ST）、順次機能チャート（SFC）の5つの制御言語について印刷および表示方法を定義します。

◆ウォッチドッグタイマ

一定時間内にENDラングまで実行できなかったとき、ウォッチドッグタイマがメジャー異常を検出します。ウォッチドッグタイマの設定は[設定]ダイアログボックスで設定できます。

◆エラー

エラーには、メジャー異常、マイナー異常、I/Oエラーの3つがあります。

メジャー異常が発生すると、コントローラは、すぐにロジックプログラムの実行を停止します。エディタには、「メジャー異常」と表示されます。状態をクリアするには、コントローラを[RUN/STOP]ダイアログボックスでリセットしてください。

マイナー異常は、軽微なエラーです。

I/Oエラーは、I/Oの読み込み/書き込みのエラーです。

◆強制変更(Forces)

ディスクリート変数は強制的にONまたはOFFにできます。これはロジックで実行する動作に優先するものです。たとえば、変数が強制的にOFFにされた場合、ロジックがONにしようとしても、OFFのままです。プログラムの強制リストは、[強制リスト]ウィンドウで見ることができます。

◆クリップボード(Clipboard)

Windowsの機能で、コピーや貼り付けのための一時記憶場所です。異なるアプリケーション間でも、ひとつのアプリケーション内でも使用可能な機能です。

◆コメント(Descriptions)

コメントは最大32767文字(半角)のテキストで、プログラムの一部についてコメントするものです。コメントの要約は、[コメントリスト]ウィンドウで見ることができます。

◆コントローラ

コントローラはロジックプログラムを実行し、I/Oを制御します。コントローラは表には現れず、LTの拡張タスクとして実行します。ロジックプログラムエディタはモニタリングモードでコントローラを監視します。

◆サブルーチン(Subroutine)

個々に別の名前のついたロジックのグループ。サブルーチンはENDとPENDというマーカの間に位置し、他のサブルーチン内には配置できません。[挿入]メニューで[サブルーチン]をクリックすると、[サブルーチン開始]と[サブルーチン終了]の両方のマーカが作成されます。2つのマーカの間にロジックを挿入することができます。サブルーチンはJump Subroutine（JSR）(ジャンプサブルーチン)という命令で呼び出されます。サブルーチンはどこからでも何度でも呼び出すことができ、コードを一回書き込むだけですむ点が便利です。サブルーチン名が必要です。

◆サブルーチン名

サブルーチン名には、文字・数字・下線を32文字まで使用できます。サブルーチン名を数字から始めることはできません。

◆システム変数(System Variables)

「システム変数」は事前に定義された特殊な変数で、コントローラの状態に関する情報を提供し、コントローラの操作に反映させます。自動で作成され削除不可能な点を除けば、通常の変数とよく似た働きをします。

◆実数(Real)

小数点を含む数や特定の記数法で表す数。ロジックプログラムエディタでは実数の範囲を±2.25e^{-308}から±1.79e^{-308}とします。有効数字は最大15桁です。

◆状態フロー(State Flow)

モニタリング時パラメータに基づいて個々の命令の動作を緑色で強調表示します。パラメータの状態に応じて、それぞれの接点が強調表示されます。

◆整数(Integer)

32ビットの情報の含まれる記憶単位。整数には、-2,147,483,648から2,147,483,647（16進数で16#0000000から16#FFFFFFF）までの値が割り付けられます。整数には小数点を含むことはできません。

◆接続線

命令の間の垂直線です。新規の命令を挿入する場合は、挿入したい位置の接続線にフォーカスをあわせます。

◆ディスクリート(Discrete Point)

OFFまたはONのいずれかの状態を持つ変数です。

◆[データ値表示リスト]ウィンドウ(Data Watch List Window)

登録した変数のモニタリング状況がこのウィンドウに表示されます。[オプション]ダイアログボックスで更新の基準を調整することができます。

◆定数

42(整定数)や3.14159(実定数)などの数です。

◆ドラッグ(Drag)

マウスの左ボタンを押したままマウスを動かしてから離します。マウスポインタにより、そこがドラッグできる有効な場所かどうかがわかります。

◆内部変数(Internal Variable)

I/0に割り付けられている以外の変数です。

◆バイト(Byte)

8ビットの情報をもつ記憶単位。バイトには0から255の値を割り当てます。ロジックプログラムエディタの整数は4バイトから成ります。

◆配列

(1つの名前を割り付けた)同じタイプの複数の要素です。

◆パラメータ(Parameter)

(命令に割り付けられている、変数、変数の要素、定数、ラベル名)への入力または(命令に割り付けられている、変数、変数の要素、定数、ラベル名)からの出力。

◆パワーフロー(Power Flow)

ロジックプログラムの実行する流れ。

◆ビット (Bit)

基本的な記憶単位、値は1または0。

◆フォーカス(Focus)

ロジックプログラムでの選択事項を強調表示するための黒色のカーソルのことです。

◆分岐

ラングに並列接続してロジックプログラムを実行します。

◆ブックマーク(Bookmark)

ロジックプログラムのどこにでも配置できる非表示のマーカーで、プログラムのその箇所にすぐに戻ることができます。

◆ブレークポイント

ロジックプログラム中にあり、ブレークポイントでロジックプログラムの実行を一時停止します。

◆プログラミングモード

ロジックプログラムを含む拡張子が.lteのファイルを編集します。

◆変数(Variable)

カウンタ、I/0の値などのデータ値です。理解しやすい変数名をつけてください。

通常、変数は自動的に作成されます。パラメータボックスか[I/0コンフィギュレーション]ウィンドウで新規の名前を入力すると、エディタは適切なタイプで新規の変数を自動的に作成します。[変数リスト]ウィンドウで不要な変数を削除できます。変数名は文字、数字、下線を含めて最大20文字で定義しますが、数字で始めることはできません。

◆母線

メインウィンドウの両端にある2本の垂直な線です。

◆モニタリングモード

エディタはコントローラに書き込んだロジックプログラムをモニタリングします。

たとえば：Power_Off_pushbutton、ResetButton、ALARM2などです。

◆命令(Instruction)

ロジックプログラムの基本的な要素(ディスクリート命令、ビット演算命令、データ操作命令、演算命令、タイマとカウンタ、プログラム制御命令)です。コントローラに特定の機能を実行させる命令です。ロジックプログラムエディタの命令は、IEC61131-3の仕様に基づいています。

◆要素(Element)

要素は変数全体ではなくある一部分の名前のことです。以下のような部分が該当します：

・タイマの変数やカウンタの変数の要素

・配列の要素

・整数の一部；変更子を参照してください。

たとえば：FillTimer.ET、LimitSwitches<5>、LimitSwitches<Index>、Flags.X<12>、SensorArray<Position>.X<Index>

◆ラング

1つ以上の命令で2本の母線の間に作成する回路です。

◆ラベル名(Label Name)

最大32文字から成る名前で、ロジックプログラム内の位置を識別したりラベルを付けたりするための名前です。ラベル名を数字から始めることはできません。

◆ロジックプログラム

アプリケーションプログラム中のラングの集合。ロジックプログラムエディタではラダー言語をサポートしています。

◆ワード(Word)

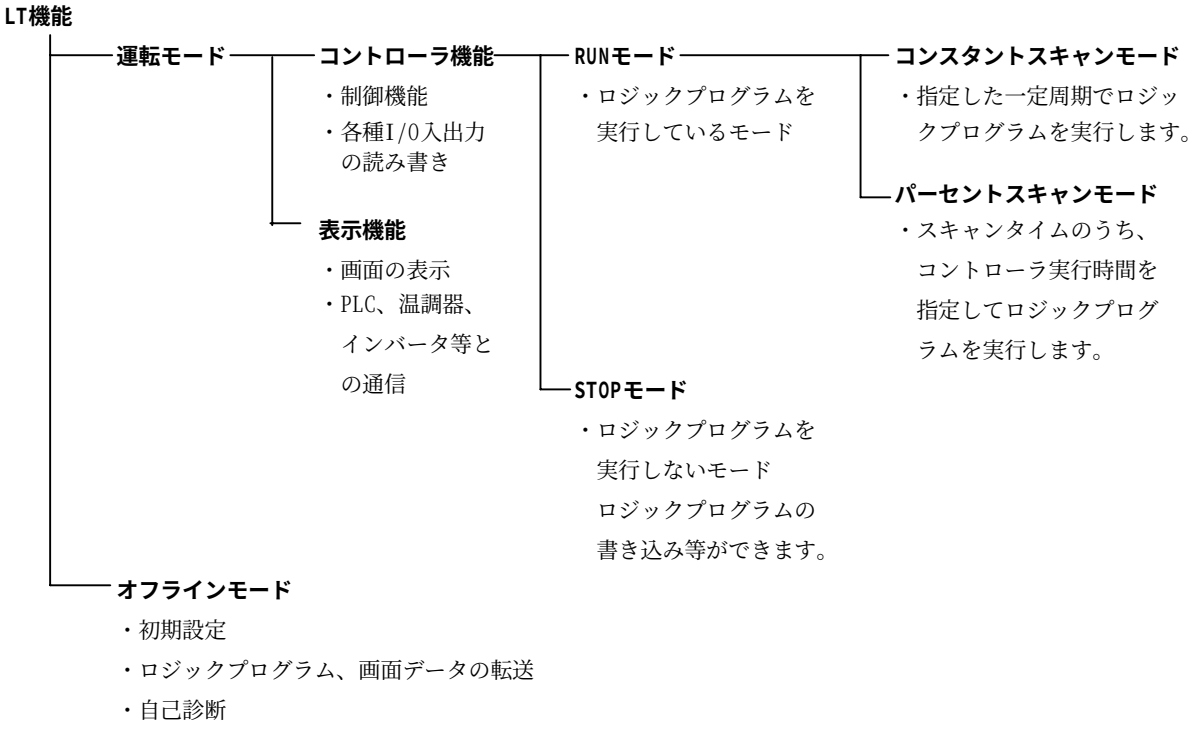
16ビットからなる記憶単位。ワードには0から65535までの値が割り付けられます。

機能編

第6章 コントローラ 機能

6.1 動作モード概要

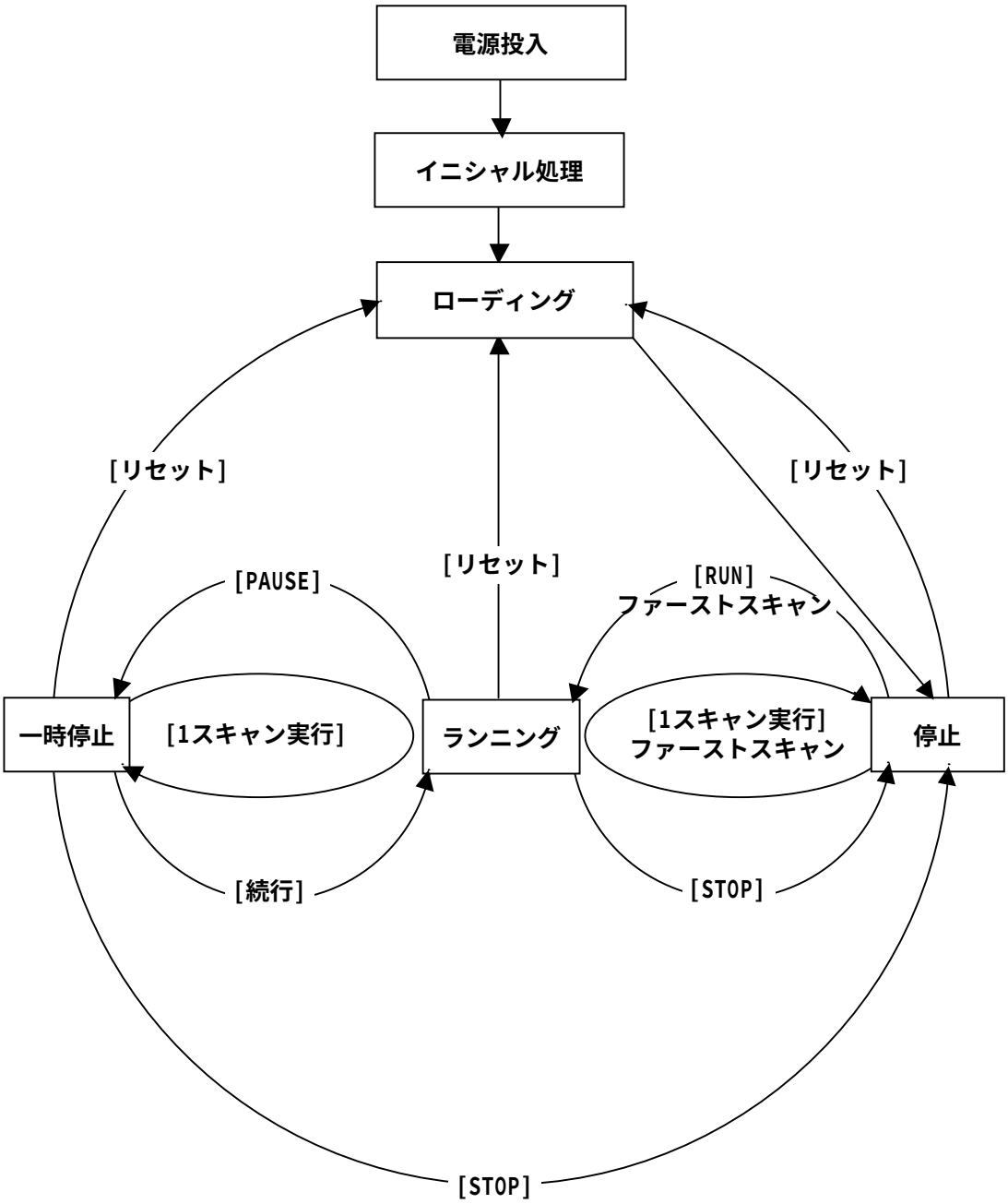
LTには、画面の表示を実行する機能と制御を実行するコントローラ機能があります。
LTの動作モードの概要を以下に示します。



重要 **・動作モードはシステム設計上重要です。この章をよくお読みになり、LTの動作をご理解の上、安全を考慮したシステム設計を行ってください。**
・オフラインモードに入るとコントローラは停止します。運転モードに戻るとLTはリセットされます。

6.1.1 コントローラ機能概略

コントローラ機能の動きは以下のようになっています。それぞれの状態については次ページ以降で説明しています。



■イニシャル処理

ロジックプログラム実行エンジンの初期状態です。ロジックプログラム実行エンジンの初期化の後、コントローラの状態は「ローディング」に移ります。

■ローディング

この状態においてメモリからロジックプログラムの読み込みを行います。ロジックプログラムが正しくロードされたかどうかのチェックを行い、正常でなければエラー処理を行います。正しくロードされれば「停止」に移ります。「電源 ON 時の動作モード」が START に設定されている場合は、[RUN]コマンドが自動実行されます。ランニングに移る時に I/O の初期化が行われます。

■停止

この状態はコントローラの停止状態です。コマンド（[リセット]、[RUN]、[1 スキャン実行]、[続行]、[PAUSE]）を受けるとそれぞれの状態に移ります。

[リセット]コマンドで「ローディング」に移ります。
このとき変数の初期化が行われます。保持型変数は電断時や LT のリセット時は直前のデータを保持しますが、モニタリングモード※1 や #Command でコントローラのリセットを行ったときは、プログラミングモード※2 で設定した値を初期値とします。非保持型変数は 0 でクリアされます。
[RUN]コマンドで「ランニング」に移ります。
[1 スキャン実行]コマンドで 1 回だけロジックプログラムを実行します。

■ファーストスキャン

I/O 読み込み、START ラベルより上に記述されたロジックプログラムの実行、I/O 書き込みを行います。

■ランニング

ロジックプログラム実行エンジンの継続実行状態です。I/O 読み込み、ロジックプログラムの実行、I/O 書き込み、システム変数（#AvgLogicTime、#AvgScanTime など）の更新を行います。

[リセット]コマンドで「ローディング」に移ります。
[STOP]コマンドで「停止」に移ります。
[PAUSE]コマンドで「一時停止」に移ります。

■一時停止

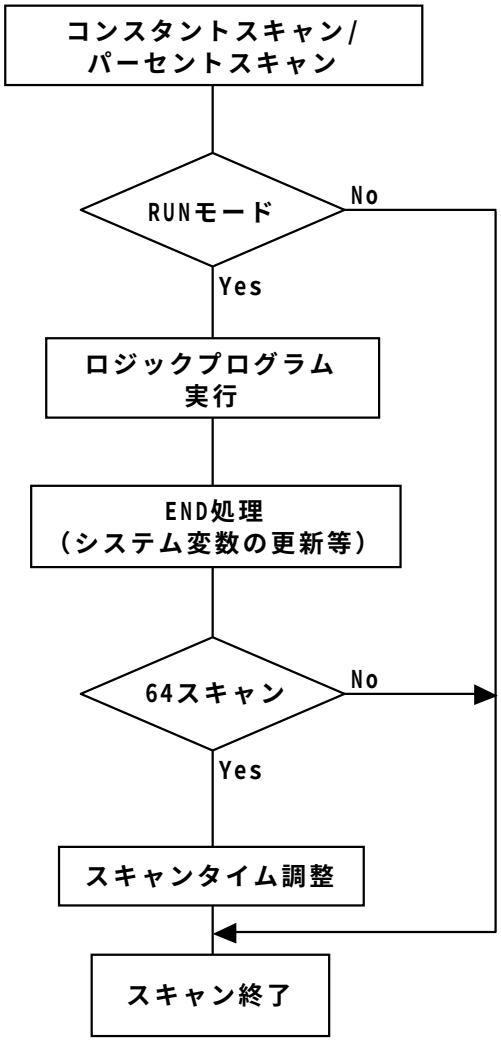
この状態はロジックプログラム実行エンジンの一時停止状態です。I/O のウォッチドッグタイムアウトを避けるため、I/O 読み込みと I/O 書き込みを実行します。しかし、ロジックプログラムを実行しないため、出力の状態は変化しません。コマンドを受けるとそれぞれの状態に移ります。

[リセット]コマンドで「ローディング」に移ります。
[1 スキャン実行]コマンドで 1 回だけロジックプログラムを実行します。
[STOP]コマンドで「停止」に移ります。
[続行]コマンドで「ランニング」に移ります。

※1 コントローラで実行しているプログラムをエディタ上で実施するモード
※2 プログラムを作成するモード

6.1.2 RUN モードの流れ

RUN モードの流れは以下のようになっています。



■スキャンタイムの調整

スキャンタイムの調整は、64 スキャンごとに行われます。コンスタントスキャンモード、パーセントスキャンモード、それぞれのスキャンタイムの調整は、以下のようになります。

◆コンスタントスキャンタイムモード

スキャンタイム = (#AvgLogicTime × 100) ÷ 50

◆パーセントスキャンモード

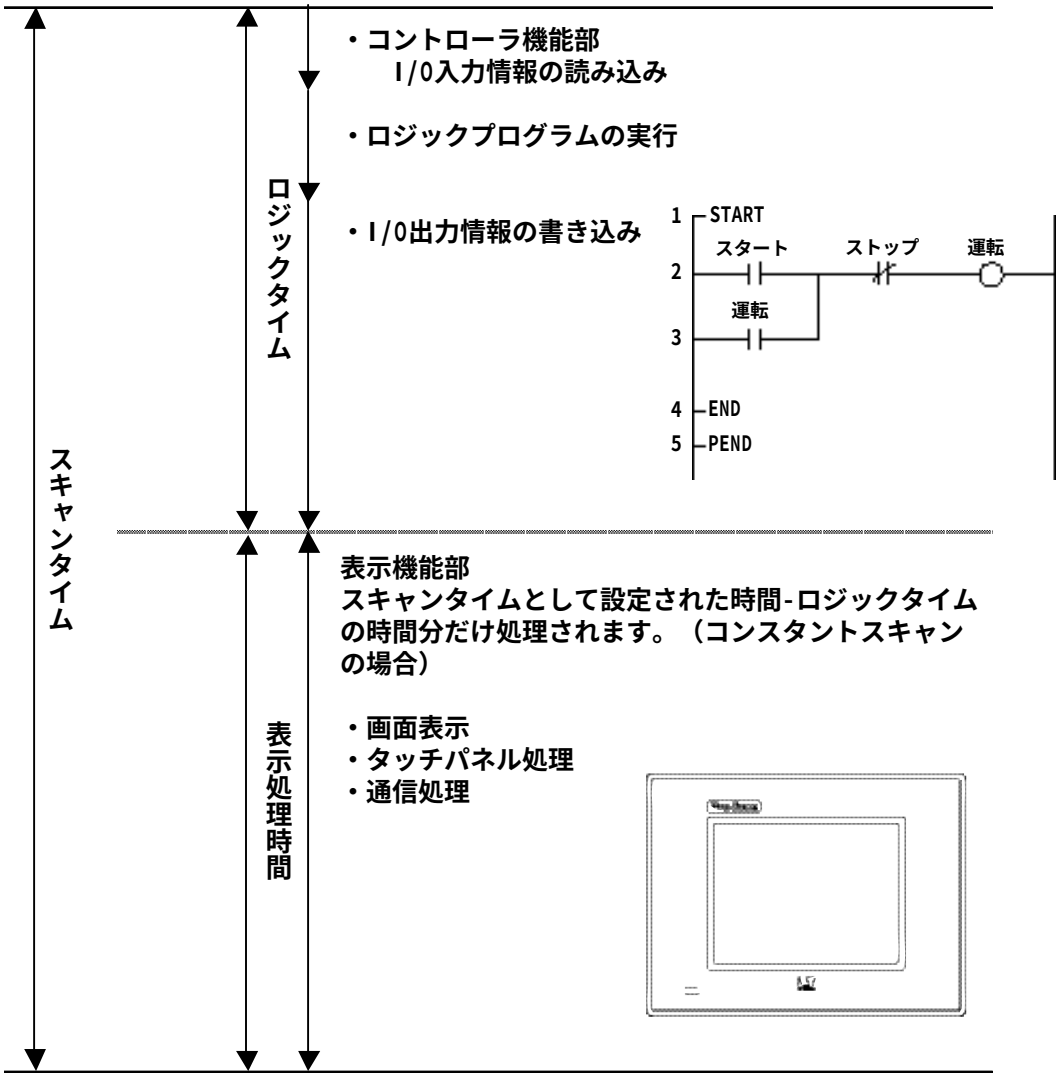
スキャンタイム = (#AvgLogicTime × 100) ÷ #PercentAlloc

#AvgLogicTime、#PercentAlloc については、**参照→ 第 8 章 システム変数**

重要 **・ LT のスキャン時間には、約 +0.3% の誤差が含まれます。**

6.1.3 LTのスキャンの概略

LTのスキャンタイムモードにはコンスタントスキャンモードとパーセントスキャンモードがあります。
概略として、LTのスキャンタイムはロジックプログラムを実行するコントローラ部と表示部（画面・タッチパネル処理、外部機器通信処理）が含まれ下図のようになっています。

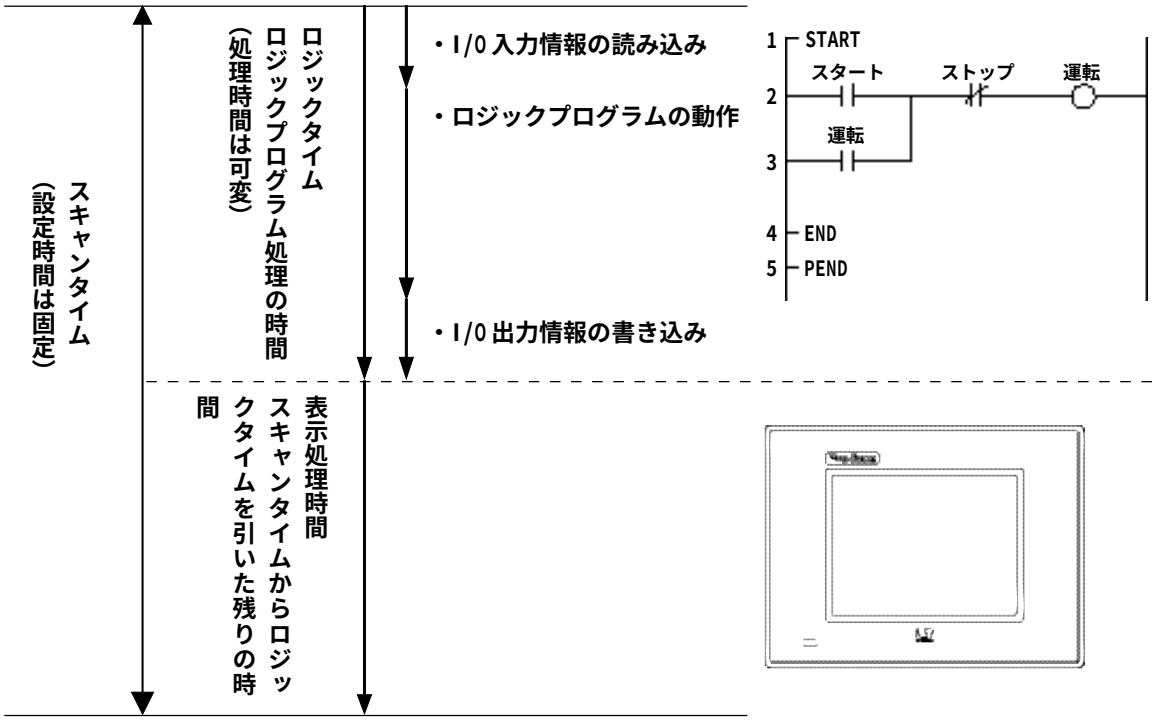


参照→6.1.2 RUN モードの流れ

■コンスタントスキャンモード

設定されたスキャンタイムを一定に保ちながら動作するモードです。

画面は監視（データ表示）がメインで操作に関しては少なく、制御（ロジックプログラム）を優先するシステムに適しています。



表示処理時間　＝　コンスタントスキャン設定値（ms）　－　ロジックタイム（可変）

例）コンスタントスキャン 50ms と設定し、ロジックタイムの実行時間が、20ms の場合
表示処理時間　＝　50ms － 20ms
　　＝　30ms

ロジックタイムが長くなれば、表示処理を行う時間は短くなります。
したがって LT 上の表示更新速度が遅くなりますが、ロジックプログラムの処理は、コンスタントに行われます。

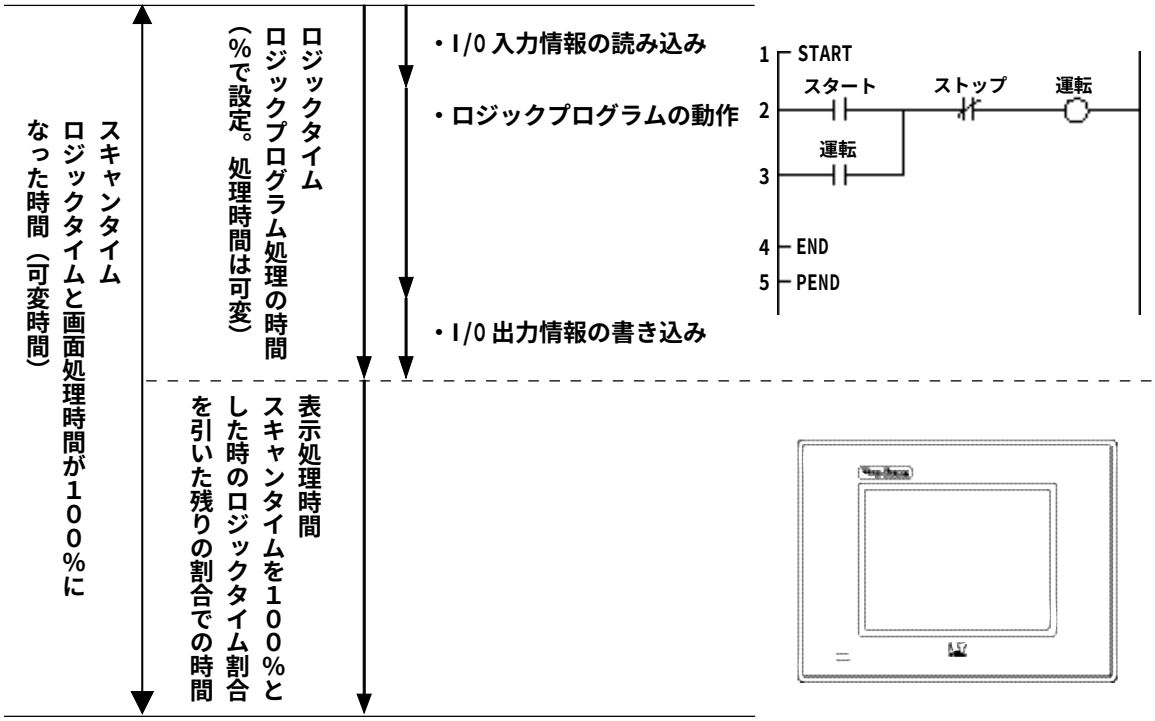
注 意　ロジックタイムがコンスタントスキャン設定値の 50%を超えた場合、スキャンタイムがロジックタイムの 2 倍になるようにスキャンタイムが自動調整されます。
例)コンスタントスキャン設定値が 50ms の場合
ロジックタイムが 30ms の時はスキャンタイムは 60ms になります。

・スキャンタイムの設定は 10ms 単位で入力してください。
・設定時間は、LT を試用運転した後、#AvgScanTime の値を参考に設定してください。
参照→ 8.2.2 #AvgScanTime

■パーセントスキャンモード

ロジックタイムを設定された割合としてスキャンタイムを可変させて動作するモードです。

ロジックプログラムより画面の操作スピードや切り替えスピードを優先したい場合に用いてください。



スキャンタイム　＝　ロジックタイム　÷　パーセントスキャン設定値（%）

例）パーセントスキャン 40%と設定し、ロジックタイムの実行時間が 20ms の場合
スキャンタイム　＝　（20　÷　40）×　100
　　＝　50ms

よって、
表示処理時間　＝　50ms　－　20ms
　　＝　30ms

ロジックタイムが長くなると、表示処理時間も長くなるので、スキャンタイムは長くなります。

したがって、ロジックタイムが長くなればなるほど、表示処理に割り当てられる時間が長くなるので、LT 上の表示更新速度は速くなりますが、ロジックプログラムの処理周期は遅くなります。

注 意
- **ロジックプログラムの一命令の処理時間には変化ありません。**
- **パーセントスキャン設定値は50%を越えて設定することはできません。**
- **パーセントスキャン設定値を 50%にした場合、表示とロジックプログラムの処理が同じ時間になるので表示処理が優先されるわけではありません。**

スキャンタイムの値が10ms 単位になるようにパーセントスキャンを設定してください。

MEMO

このページは、空白です。
ご自由にお使いください。

第7章 変数

ここでは、LT Editorで用いられる変数タイプについて説明します。

ハードウェアに依存しない変数を使用することにより、再利用性の高いプログラムを作成することができます。

7.1 変数名

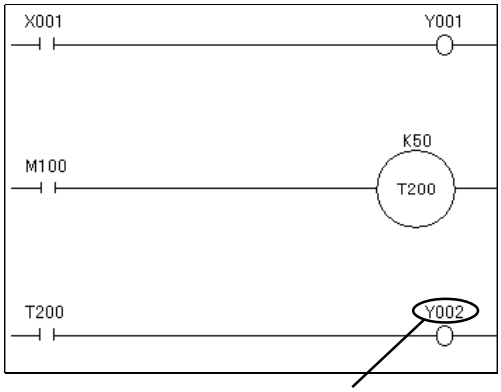
LT Editorでは、I/Oやカウンタのデータを格納するエリアとして変数を使用します。変数はユーザーで定義し、ロジックプログラムでそのままの名前で使用します。

一般のPLCでは、データを格納するエリアはデバイスアドレスいう形でPLCメーカー特有の名称で扱われます。

		外部入出力	内部リレー	タイマ	データレジスタ	
例)	M社	X001	M100	T200	D00001	
	O社	01	1001	TIM000	DM0000	
	(株)デジタル	スイッチ1	タイマスタート	タイマ	運転時間	など

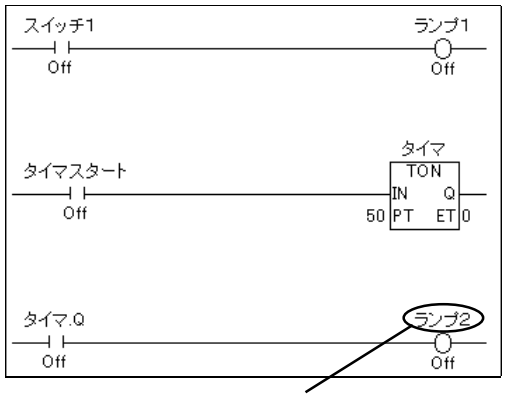
LT Editorではこのようなデバイスアドレスに任意の名前を付け、**変数**という形でロジックプログラム中で使用します。

PLC



各メーカーごとの
デバイスアドレス

LT Editor



変数名
(LT Editorでユーザーが設定した任意の名前)

・LT Editor(Ver1.03以降)では、**C:\Program Files\Pro-face\LT\SAMPLEフォルダ内の、変数サンプルI(TypeA).lte、変数サンプルII(TypeA).lte、変数サンプルIII(TypeA).lte**内にあらかじめ下記変数を登録しています。

変数サンプルI(TypeA).lte

デバイス種類	変数名	変数タイプ	変数の属性
内部リレー	M0～M127	ディスクリート	非保持・グローバル・インターナル
カウンタ	C0～C63	カウンタ	保持・グローバル・インターナル
タイマ	T0～T63	タイマ	保持・グローバル・インターナル
レジスタ	D0～D127	整数	保持・グローバル・インターナル
保持リレー	L0～L127	ディスクリート	保持・グローバル・インターナル
入力リレー	X0～X15	ディスクリート	非保持・グローバル・入力
出力リレー	Y0～Y15	ディスクリート	非保持・グローバル・出力

変数サンプルII(TypeA).lte

デバイス種類	変数名	変数タイプ	変数の属性
内部リレー	M[0]～M[255]	ディスクリート	非保持・非グローバル・インターナル
カウンタ	C0～C31	カウンタ	非保持・グローバル・インターナル
タイマ	T0～T31	タイマ	非保持・グローバル・インターナル
レジスタ	D[0]～D[255]	整数	非保持・非グローバル・インターナル
保持リレー	L[0]～L[255]	ディスクリート	保持・非グローバル・インターナル
リンクリレー	B[0]～B[127]	ディスクリート	非保持・グローバル・インターナル
リンクレジスタ	W[0]～W[127]	整数	非保持・グローバル・インターナル
入力リレー	X[0]～X[15]	ディスクリート	非保持・非グローバル・入力
出力リレー	Y[0]～Y[15]	ディスクリート	非保持・非グローバル・出力

変数サンプルIII(TypeA).lte

デバイス種類	変数名	変数タイプ	変数の属性
内部リレー	R[0]～R[255]	ディスクリート	非保持・非グローバル・インターナル
カウンタ	CNT0～CNT31	カウンタ	非保持・グローバル・インターナル
タイマ	TIM0～TIM31	タイマ	非保持・グローバル・インターナル
レジスタ	DM[0]～DM[511]	整数	保持・グローバル・インターナル
保持リレー	HR[0]～HR[255]	ディスクリート	保持・非グローバル・インターナル
リンクリレー	LR[0]～LR[127]	ディスクリート	非保持・グローバル・インターナル
入力リレー	IN[0]～IN[15]	ディスクリート	非保持・グローバル・入力
出力リレー	OUT[0]～OUT[15]	ディスクリート	非保持・グローバル・出力

変数名をつける際の制限は以下の通りです。

- 変数名は最大で半角 20 文字、全角 10 文字（20 バイト）です。
  配列の各要素およびタイマ、カウンタの各専用変数(「.PT」、「.PV」など)も文字数に含まれますので作成時には注意してください。
- 全角文字と半角文字は区別されません。先に登録した変数名が有効となります。
  例)「タンク」、「ﾀﾝｸ」の順で登録した場合、「タンク」が有効となります。
- 大文字小文字は区別されません。先に登録した変数名が有効となります。
  例)「TANK」、「tank」の順で登録した場合、「TANK」が有効となります。
- 数字で始まる変数名は全角､半角問わず使用できません。
- スペースを含めることはできません。
- 「_」以外の記号は使用できません。ただし、「_」を「__」のように重ねて使用することはできません。(○:tank_1、×:tank__1)
- 「#」で始まる変数名はシステムで予約されているため、使用できません。
- 変数名「LS」、「LSS」は、システムデータエリア、読み込みエリア､特殊リレーとしてシステムで予約されています。ユーザー定義の変数としては使用できません。
  (「LS」、「LSS」は半角文字で作成してください。)
  参照→第 10 章 LS エリアリフレッシュ

- 変数名を付ける際にシステムによって変数名をブロック分けすると、LT Editor の変数リストから変数を探す時に便利です。(このとき、ブロック名と変数名の間に「_」を入れると見やすくなります。)

  例)システムに複数のコンベア(コンベア A、B、C・・)がある場合、コンベア A に設置されたモーターやセンサの変数名を

  A_モーター
  A_センサ

  とします。
  また、ディスクリート(bit)を B、整数(integer)を I、浮動小数点(float)を F として

  AB_モーター起動スイッチ
  AI_モーター回転数
  AF_モーター力率

  という変数名にすると､接点やコイルに使用する変数と四則演算に使用する変数を区別することができます。

- PLC のデバイス名と同様に変数名を扱いたい場合は､必要量の変数を配列で取ると便利です。

例)

PLC デバイス	LT Editor 配列変数	LT Editor 変数タイプ	PLC デバイス	LT Editor 配列変数	LT Editor 変数タイプ
外部入力	X[100]	ディスクリート	内部リレー	M[100]	ディスクリート
外部出力	Y[100]	ディスクリート	データレジスタ	D[100]	整数

変数の設定については**参照→1.2 変数の作成**

あらかじめシステムで予約されているシステム変数については、**参照→第 8 章 システム変数**

7.2 変数タイプ

変数には、大きく分けてディスクリート(ビット)、整数、実数の3つのタイプがあります。

また、これらの変数タイプで構成されたタイマとカウンタもあります。

ディスクリート、整数、実数の各変数は、配列指定もできます。配列指定の詳細については、「7.3 配列変数へのアクセス」を参照してください。

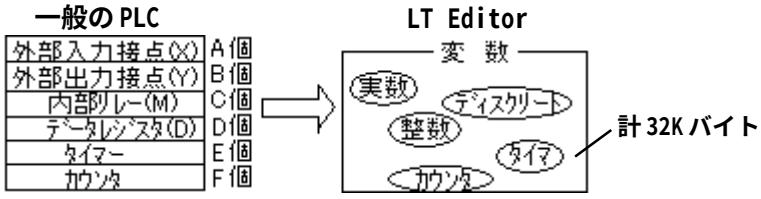
配列変数のサイズ（要素数）は最大で65535まで設定できます。ただし、LTの変数格納エリアは約32K バイトです。全変数のメモリ使用量がこれを越えないように設定してください。

下表にそれぞれの変数が使用するメモリ量を示します。

変数のタイプ	使用するメモリ量(単位:バイト)
ディスクリート	12
ディスクリート配列	20+(要素数×12)
整数	8
整数配列	20+(要素数×8)
実数	16
実数配列	20+(要素数×16)
タイマ	48
カウンタ	80

また、サンプルプログラムを含めたメモリの使用量については、付録を参照してください。

PLCでは各デバイスごとに使用数に制限がありますが、LTの変数は変数格納エリアの32Kバイト以内であれば変数タイプに関係なくいくつでも登録できます。



PLCのデバイスとLTの変数との比較

■ディスクリート変数

ディスクリート変数はON/OFFを表す1ビットの長さの変数で、0か1の値を持ちます。

■整数変数

整数変数は32ビットの長さの変数で、-2147483648～2147483647の整数値を持ちます。

■実数変数

実数変数は64ビットの長さの変数で、±2.225e-308～±1.79e+308の浮動小数点と0の値を持ちます。小数点以下15桁まで使用できます。

■タイマ・カウンタ

タイマとカウンタは複数の専用変数で構成されます。

専用変数の変数タイプはそれぞれ独立しています。

◆タイマ

タイマ命令に使用する変数には、以下の4つの専用変数があります。
詳しくは、**9.2 命令詳細**を参照してください。

専用変数	内容	変数タイプ
変数.PT	設定値	整数
変数.ET	現在値	整数
変数.Q	タイマ出力ビット	ディスクリート
変数.TI	タイマ計測ビット	ディスクリート

※「変数」には任意の変数名を付けることができます。

・変数**タイマ**が非保持の場合でも、専用変数**タイマ.PT**は保持されたままです。
保持については、**■変数の属性**を参照してください。

◆カウンタ

カウンタ命令に使用する変数には、以下の7つの専用変数があります。
詳しくは、**9.2 命令詳細**を参照してください。

専用変数	内容	変数タイプ
変数.PV	設定値	整数
変数.CV	現在値	整数
変数.R	カウンタリセット	ディスクリート
変数.UP	アップカウンタ	ディスクリート
変数.QU	アップカウンタ出力	ディスクリート
変数.QD	ダウンカウンタ出力	ディスクリート
変数.Q	カウンタ出力	ディスクリート

※「変数」には任意の変数名を付けることができます。

・変数**カウンタ**が非保持に指定されていても、専用変数**カウンタ.PV**は保持されたままです。
・カウンタをリセットしたスキャンでは、カウンタの更新は行いません。カウンタのリセットのために1スキャン必要となります。
保持については、**■変数の属性**を参照してください。

■変数の属性

変数には、変数タイプの他に、以下のような属性があります。

ここではそれぞれの属性について説明します。

◆インターナル

LT内部で使用します。外部入出力には使用できません。PLCでは、内部リレー(内部レジスタ)に相当します。

◆入力 / 出力

外部入出力を使用できます。I/Oコンフィギュレーションで I/O に割り付けます。PLCでは入力 / 出力リレーに相当します。

I/Oコンフィギュレーションについては、**参照→1.9 I/Oの割り付け**

◆保持

保持型変数はSRAMで管理されるため、電源断時もデータ値の保持が可能です。また、保持型変数は、プログラミングモードで設定した値を初期値として持っています。電断時やLT本体のリセットを行った時は、直前のデータを保持しますが、モニタリングモードや#Commandでコントローラのリセットを行った時、またはロジックプログラムをダウンロードした時は、プログラミングモードで設定した値で初期化されます。また、LTの lte ファイルを読み込むことで実行結果をEditorで保存することができます。ただし、保持型変数を初期値として使う場合、ロジックの実行中に変化するような設計にすると、Editorで読み込んだときに設定した初期値が失われるので、設計上の注意が必要です。非保持型のデータは0クリアまたはOFFになります。

◆グローバル

グローバルは、グローバルと非グローバルがあります。画面エディタで部品などの表示機能に使用する変数はグローバルに設定してください。グローバル変数はロジックプログラムを保存することにより、自動的にシンボルエディタにLTシンボルとして登録され、画面エディタの表示機能でも共有できるようになります。変数リストで複数個選択することでグローバル / 非グローバルの一括変換が可能です。グローバル変数は2048個まで設定できます。

参照→ 3.6 変数の属性変更

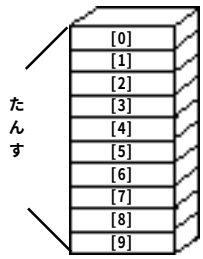
・あらかじめ用意されているシステム変数は初期設定でグローバルに設定されています。

7.3 配列変数へのアクセス

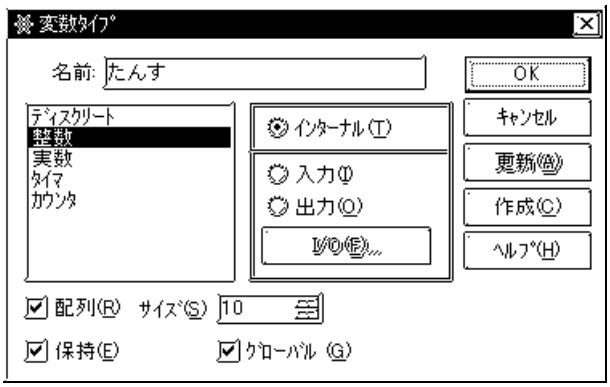
LT Editorにて、配列変数の要素・ビット・バイト・ワード単位でアクセスする方法を説明します。

配列とは、一つの変数名で複数の要素を宣言して扱う方法です。これにより同じタイプの変数をまとめて登録できます。

例えば、机やたんすの引出しを想像してください。



要素数10の配列変数**たんす**には、たんすという名前の引出しが[0]から[9]まで10個用意されているということになります。このたんすの各引き出しを**たんす[0]**、**たんす[1]**、…、**たんす[9]**と呼びます。これら一つ一つの引き出しがPLCでいうところのデータレジスタ一つ一つになります。よってたんすメモリを10個使用する場合は、たんすという変数名でサイズ(要素数）10の配列と宣言します。変数タイプの設定は以下のようになります。



■ディスクリート配列へのアクセス

ディスクリート配列では、変数名に修飾語**[n]**をつけると配列の要素単位でアクセスできます。**n**にはアクセスする要素番号を指定します。ただし、配列の1番目は要素番号0になります。

例）ディスクリート配列**モーター設定**は10要素のディスクリート配列です。

７番目の要素は出力コイル**扇風機**を制御します。

７番目の要素をONにすると、出力コイルがONになります。

モーター設定の７番目の要素にアクセスする場合**モーター設定[6]**とします。

■整数配列へのアクセス

整数配列へは、配列の要素・ビット・バイト・ワード単位でのアクセスができます。

変数名の直後に**[n]**をつけることで配列の要素単位でアクセスできます。

ビット･バイト･ワード単位でアクセスするには変数名に下表の修飾語をつけてアクセスします。mにはアクセス単位で何番目にアクセスするかを指定します。

アクセス単位	修飾語
ビット	.X[m]
バイト	.B[m]
ワード	.W[m]

◆整数配列で要素単位のアクセスをする場合

例えば、整数配列を使用して数値計算、反復情報のトラッキング、データのロギングができます。

例) 整数配列**飲料水売り上げ**が１ヶ月間に売ったソーダの数を記録するときは、以下のようなデータ構造になります。

配列は31の整数型要素で構成され、１ヶ月(31日)の各日に対応しています。

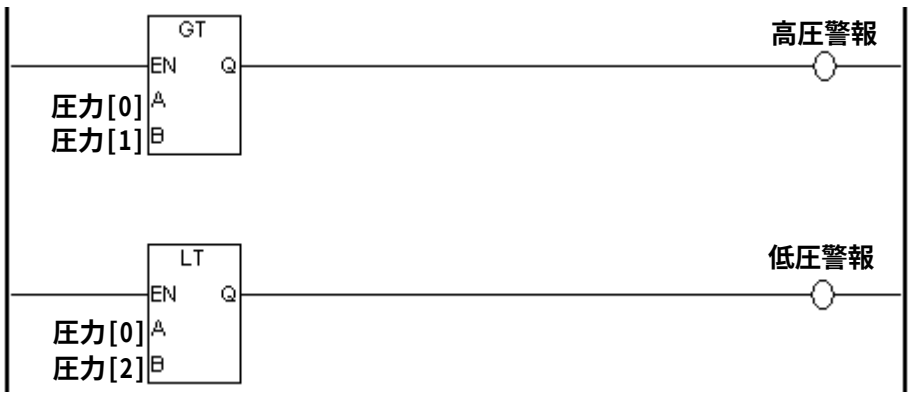
1日目	飲料水売り上げ[0]
2日目	飲料水売り上げ[1]
3日目	飲料水売り上げ[2]
4日目	飲料水売り上げ[3]
	･ ･ ･
28日目	飲料水売り上げ[27]
29日目	飲料水売り上げ[28]
30日目	飲料水売り上げ[29]
31日目	飲料水売り上げ[30]

以下の例では、整数配列**圧力**には3つの要素があります。

圧力[0]はボイラの現在の圧力、**圧力[1]**は圧力上限値、**圧力[2]**は圧力下限値を表します。

圧力が高圧限界より上がったり、低圧限界より下がったりすると、警報がONになります。

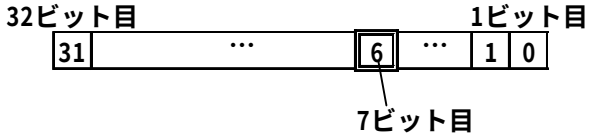
現在の圧力	圧力[0]
圧力上限値	圧力[1]
圧力下限値	圧力[2]



◆整数配列でビット単位のアクセスをする場合

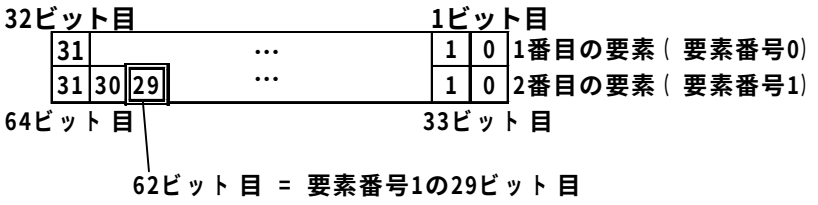
また、整数配列はディスクリート配列変数と同様に、ビット・バイト・ワード単位でのアクセスと組み合わせてアクセスすることもできます。整数配列変数**飲料水売り上げ**の **n+1**番目の要素の **m+1**ビット目にアクセスするには**飲料水売り上げ[n].X[m]**のようになります。

例）・ 整数変数**アラーム**の7ビット目にアクセスする場合
アラーム .X[6]



・ 整数配列変数**飲料水売り上げ**の62ビット目にアクセスする場合
飲料水売り上げ .X[61]

または、**飲料水売り上げ[1].X[29]**



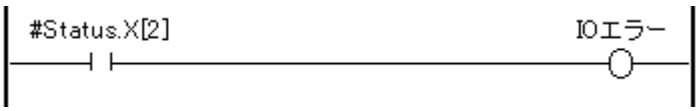
したがって、**飲料水売り上げ .X[61] = 飲料水売り上げ[1].X[29]**となり、どちらも整数配列**飲料水売り上げ**の62ビット目へのアクセスになります。

・ 整数配列変数**飲料水売り上げ**の6バイト目にアクセスする場合
飲料水売り上げ .B[5]　または　**飲料水売り上げ[1].B[1]**

・ 整数配列変数**飲料水売り上げ**の5ワード目にアクセスする場合
飲料水売り上げ .W[4]　または　**飲料水売り上げ[2].W[0]**

・ 飲料水売り上げ.X[61]と飲料水売り上げ[0].X[61]は、同じ意味です。

以下の例では、システム変数#Statusの3番目のビットをNO命令の変数として使用します。#Statusの3番目のビットは、LTにI/Oエラーがあるかどうかを通知します。したがって、3番目のビットがONになると、出力コイル**IOエラー**がONになり、I/Oエラーが発生したことを知らせます。



■実数配列へのアクセス

実数配列へは、配列の要素単位でアクセスできます。
変数名に修飾語**[n]**をつけてアクセスします。

nにはアクセスする要素番号を指定します。ただし、配列の１番目は要素番号０になります。

例）・ 実数配列**溶液温度**の５番目の要素にアクセスする場合
溶液温度[4]

注： 画面エディタで扱える変数の数は2048個です。配列の要素も１つの変数になります。例えば、要素数５の配列を変数として扱った数は５になります。

実数配列を使用して数値計算、反復情報のトラッキング、データのロギングができます。

例） 実数配列**溶液温度**が１時間毎に24時間分の溶液の温度を記録するときは、以下のようなデータ構造になります。

配列は24の実数型要素で構成され、１日のうちの１時間にそれぞれ対応しています。

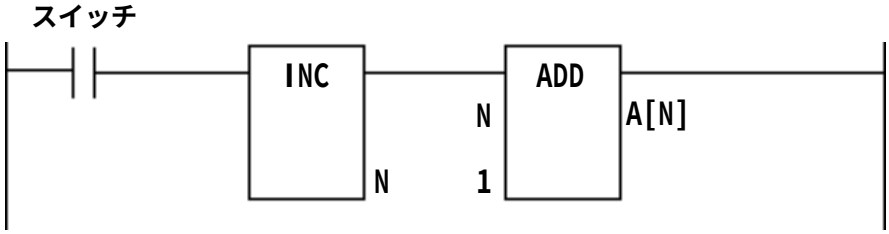
溶液温度[0]は、0:00の温度データに対応します。

溶液温度[0]
溶液温度[1]
溶液温度[2]
溶液温度[3]
-
-
-
溶液温度[20]
溶液温度[21]
溶液温度[22]
溶液温度[23]

■配列への間接アクセス

配列の要素[n]を整数変数で間接的にアクセスすることができます。また、 .X[m]、B[m]、W[m]といった修飾語の[]かっこ内の番号も間接的にアクセスすることができます。

例えば、下の回路のようにスイッチを押すと、INC命令でNはスキャンする度に１ずつ増加し、ADD命令でNと１を加算した値がA[N]に代入されます。５回スキャンしたとすれば、A[0]に1、A[1]に2、A[2]に3、A[3]に4、A[4]に５が代入されます。ただし、Nの初期値は０とします。



第8章 システム変数

ここでは、コントローラであらかじめ定義されているシステム変数について説明します。

8.1 システム変数一覧

システム変数はコントローラの状態を表し、動作に影響します。通常の変数と同じように変数タイプを持ち同じ動きをします。システム変数はあらかじめ定義されているため、名称の変更や削除はできません。

区分	システム変数	説明	初期値	変数タイプ	
情報	#AvgLogicTime	64スキャンごとの平均ロジックタイム(読み込み、実行、書き込み)を示します。(単位:ms)	0	整数	読み込み専用
	#AvgScanTime	64スキャンごとの平均スキャンタイム(読み込み、実行、書き込み、表示処理)を示します。(単位:ms)	0	整数	
	#Clock100ms	0.1sのクロックを発生します。	-	ディスクリート	
	#EditCount	LTでは現在は使用されていません。	-	整数	
	#ForceCount	強制変更された変数の延べ数を示します。	0	整数	
	#IOStatus	I/Oドライバの状態を示します。	-	整数[10]	
	#LogicTime	最新のロジックタイム(読み込み、実行、書き込み)を示します。(単位:ms)	0	整数	
	#Platform	コントローラのプラットフォームを示します。	-	整数	
	#ScanCount	実行されたスキャン数を示します。現在のスキャンは含みません。	0	整数	
	#ScanTime	最新のスキャンタイム(読み込み、実行、書き込み、表示処理)を示します。(単位:ms)	0	整数	
	#Status	コントローラの状態を示します。	-	整数	
	#StopPending	LTでは現在は使用されていません。	-	ディスクリート	
	#Version	コントローラのバージョンを示します。	-	整数	
	#WCLScan	LTでは現在は使用されていません。	-	整数	
	#WCLStatus	LTでは現在は使用されていません。	-	整数	
	#Year	「年」をBCD2桁で格納しています。	-	整数	
	#Month	「月」をBCD2桁で格納しています。	-	整数	
	#Day	「日」をBCD2桁で格納しています。	-	整数	
	#Time	「時分」をBCD4桁で格納しています。	-	整数	
	#WeekDay	「曜日」を0～6の値で格納しています。	-	整数	
	#WatchdogTime	エディタまたはオフラインにて設定された値を示します。(単位:ms)	-	整数	

区分	システム変数	説明	初期値	変数タイプ	
エラー	#FaultCode	最新のエラーコードを示します。	-	整数	読み込み専用
	#FaultRung	エラーが発生したラング番号を示します。	-	整数	
	#IOFault	エラーが発生したときONにします。	-	ディスクリート	
	#Overflow	算術命令または実数から整数への変換でオーバーフローが発生したときONにします。	0	ディスクリート	
設定	#Command	コントローラの動作モードを変更します。	0	整数	書き込み可能
	#DisableAutoStart	電源ON時の動作モードの設定	-	ディスクリート	
	#Fault	ErrorHandlerサブルーチン内で実行を停止するために使用します。	0	ディスクリート	
	#FaultOnMinor	マイナー異常が検出されたときロジックの実行を終了するかどうかを設定します。	0	ディスクリート	
	#PercentAlloc	パーセントスキャンを設定します。(単位:%)	0	整数	
	#PercentMemCheck	LTでは現在は使用されていません。	-	整数	
	#Screen	LTの「画面切替」画面番号を書き込みます。(BIN／BCD)	0	整数	
	#StopScans	LTでは現在は使用されていません。	-	整数	
	#TargetScan	コンスタントスキャンを設定します。(単位:ms)	-	整数	

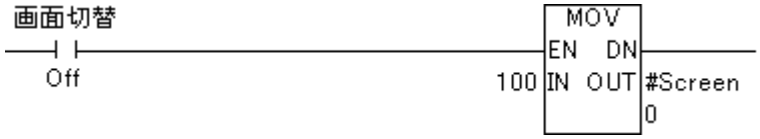
・ #Year、#Month、#Day はLTの時計データを格納しています。時計データの変更はLT本体の初期設定またはシステムデータエリアへの書き込みで行います。

参照→「LTシリーズユーザーズマニュアル」(別売)、「機器接続マニュアル」

8.1.1 システム変数使用例

システム変数の使用方法を#Screenを例にあげて説明します。

以下のロジックプログラムは、画面番号100のベース画面(B100)に切り替えるためのプログラムです。スイッチを押すと、#Screenに100が代入されることにより、画面が切り替わります。



8.2 システム変数詳細

ここでは各システム変数の詳細を説明します。

8.2.1 #AvgLogicTime

#AvgLogicTimeは、平均ロジックタイムをms単位で格納します。

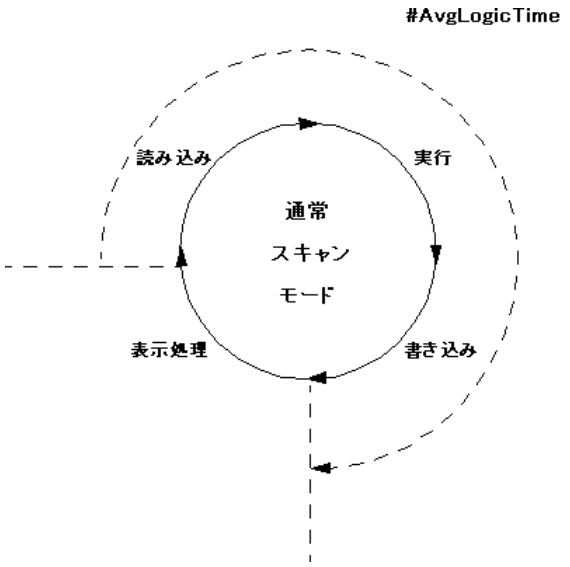
平均ロジックタイムとは、1回のスキャンでI/Oの読み込み、ロジックプログラムの実行、I/Oの書き込みまでに必要な時間の平均です。

#AvgLogicTimeは、64回スキャンするごとに更新されます。

変数タイプ: 整数

設定: コントローラ

読み込み専用



8.2.2 #AvgScanTime

#AvgScanTimeは、平均スキャンタイムをms単位で格納します。

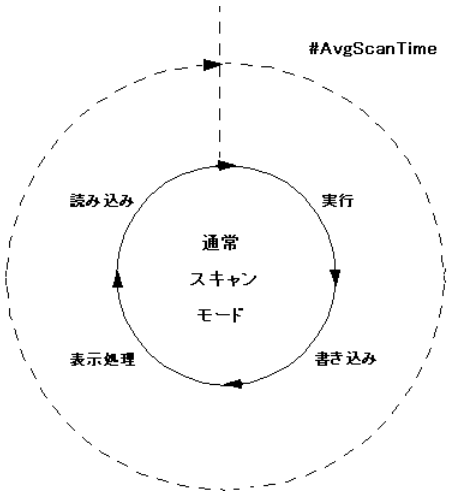
平均スキャンタイムとは、1回のスキャンでI/Oの読み込み、ロジックプログラムの実行、I/Oの書き込み、表示処理までに必要な時間の平均です。

#AvgScanTimeは、64回スキャンするごとに更新されます。

変数タイプ: 整数

設定: コントローラ

読み込み専用



8.2.3 #Clock100ms

＃Clock100msは100msのクロックを発生させます。読み込み専用ですのでクロック値は変更しないでください。

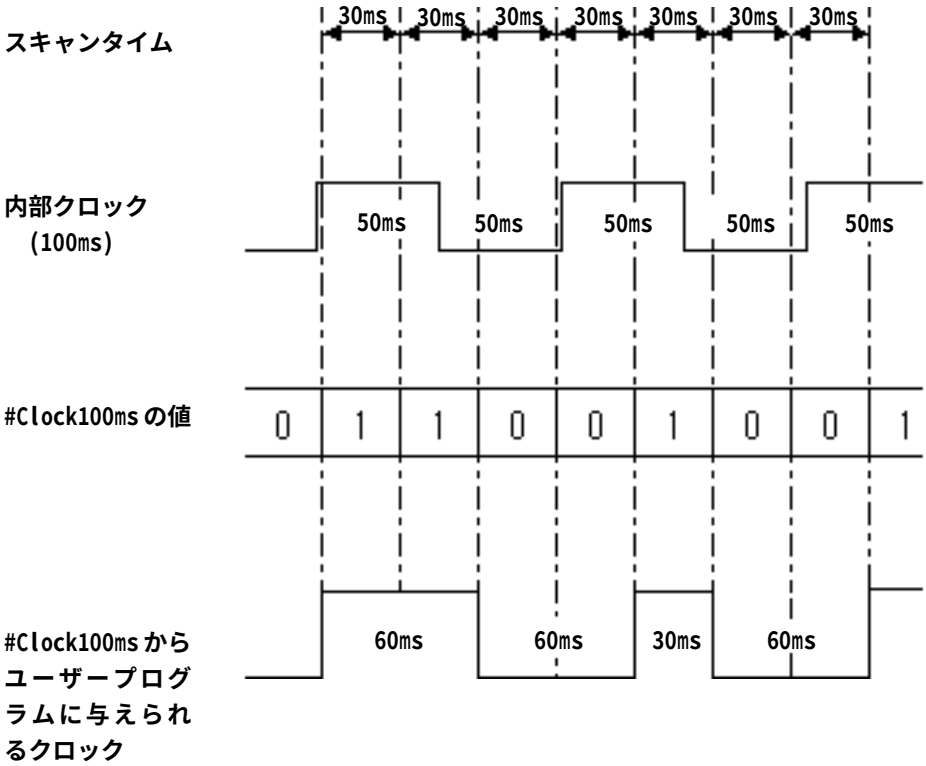
変数タイプ: ディスクリート

設定: コントローラ

読み込み専用

・LTのスキャンタイムが50msを超える場合、#Clock100msのクロックは保障されません。
・#Clock100msのクロックは、LTの各スキャンの初めに内部クロックの100msクロックを読み込んでいるため、誤差が生じます。

◆スキャンタイムが30msの場合



・ #Clock100msには、スキャンタイムと同等の誤差を含みます。

8.2.4 #Day

#Day はコントローラに設定されている「日」を BCD2 桁で示します。

変数タイプ：整数

設定：コントローラ

読み込み専用

年、月、日、時分は以下のシステム変数で表されます。

例)2001 年 7 月 14 日 6 時 19 分の場合

	年	月	日	時分
システム変数	#Year	#Month	#Day	#Time
値	1	7	14	619

8.2.5 #ForceCount

#ForceCount は、現在のロジックプログラムに強制変更された数がいくつあるかを格納します。強制変更については、**2.2 コントローラの RUN/STOP** を参照してください。

変数タイプ: 整数

設定: コントローラ

読み込み専用

[データ値表示リスト]ウィンドウでは、ロジックプログラムに 5 つの強制変更された変数があることが表示されています。

8.2.6 #IOStatus

#IOStatus は I/O ドライバによりセットされます。I/O ドライバの現在の状態を格納します。

I/O ドライバの状態は #IOStatus[1]に格納されます。

#IOStatus の値が 0 のときは、I/O ドライバは正常です。0 以外の値のときは、I/O ドライバによって表示されている内容が異なります。

変数タイプ: 整数[10]

設定: コントローラ

読み込み専用

[データ値表示リスト]ウィンドウでは、I/O ドライバ 1 にエラー 802 が発生したことが表示されています。

I/O ドライバのエラーコードの内容については、
「第 11 章 I/O ドライバ」をご覧ください。

8.2.7 #LogicTime

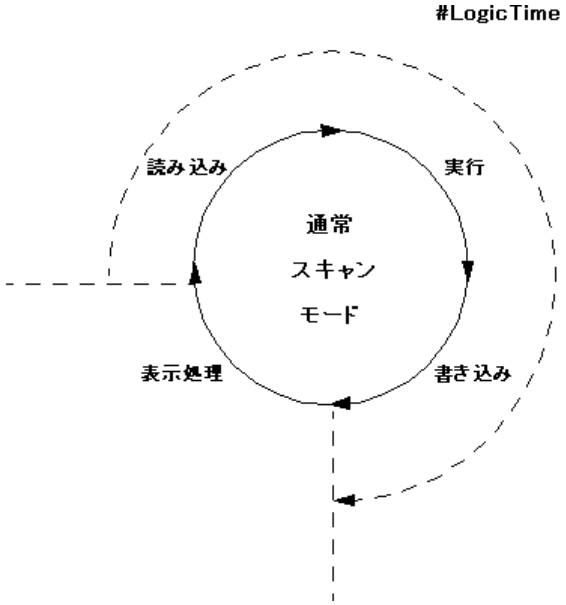
#LogicTime は、前回のスキャンのロジックタイムを表します。

ロジックタイムとは、1 回の スキャンで I/O の読み込み、ロジックプログラムの実行、I/O の書き込みまでに必要な時間です。表示処理を実行している時間は含まれません。

変数タイプ: 整数

設定: コントローラ

読み込み専用



8.2.8 #Month

#Month はコントローラに設定されている「月」を BCD2 桁で示します。

変数タイプ：整数

設定：コントローラ

読み込み専用

年、月、日、時分は以下のシステム変数で表されます。

例)2001 年 7 月 14 日 6 時 19 分の場合

	年	月	日	時分
システム変数	#Year	#Month	#Day	#Time
値	1	7	14	619

8.2.9 #Platform

#Platform は現在コントローラが RUN しているプラットフォームを表示します。

変数タイプ: 整数

設定: コントローラ

初期値: 1

読み込み専用

値	プラットフォーム
16#54	LT Type A
16#64	LT Type B/Type B+
16#74	LT Type C
16#114	LT Type H-AD
16#144	LT Type H-ADP
16#154	LT Type H-ADT

8.2.10 #ScanCount

#ScanCount はカウンタで、ロジックプログラムのスキャンが 1 回終わるごとにインクリメントされます。

#ScanCount の値の範囲は 0 ～ 16#FFFFFFFF です。最大値(16#FFFFFFFF)を超えた場合、#ScanCount の値は 0 になります。

変数タイプ: 整数

設定: コントローラ

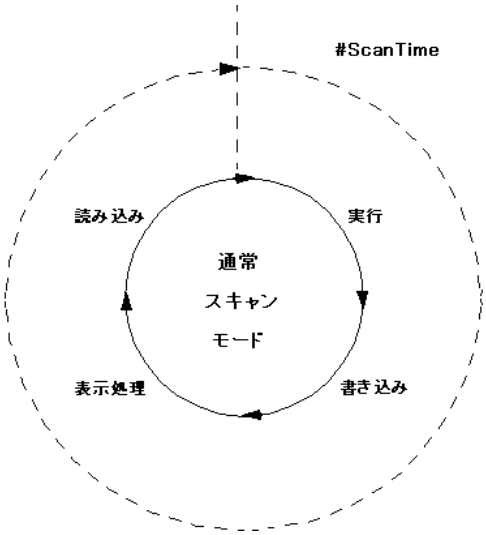
読み込み専用

#ScanCount を確認すると、ロジックプログラムが実行されているかを、容易に知ることができます。

8.2.11 #ScanTime

#ScanTime は、最新のスキャンタイムを ms 単位で格納します。

スキャンタイムとは、I/O の読み込み、ロジックプログラムの実行、I/O の出力、表示処理までに必要な時間です。



変数タイプ: 整数

設定: コントローラ

読み込み専用

8.2.12 #Status

#Status は、コントローラの状態を表示します。バイトとビットを、以下のように定義します。

バイト 0 には、コントローラの現在のエラー状態が表示されます。

バイト 1 には、エラー状態の履歴が表示されます。コントローラをリセットしたときのみ、0 にリセットされます。

バイト 2 には、現在の動作状態が表示されます。

バイト 3 は予約エリアです。

変数タイプ: 整数

設定: コントローラ

読み込み専用

ラッチエラーフラグを使用すると、断続するエラーの検出が容易にできます。#Status は、16 進数で表示してください。

以下のエラーフラグが1ならば、以下の状態であることを示します。

		エラーフラグ
バイト0	ビット0	メジャー異常
	ビット1	マイナー異常
	ビット2	I/Oエラー
	ビット3	予約
	ビット4	読み込みエラー
	ビット5	予約
	ビット6	スキャンエラー
	ビット7	予約

		ラッチエラーフラグ
バイト1	ビット8	メジャー異常
	ビット9	マイナー異常
	ビット10	I/Oエラー
	ビット11	予約
	ビット12	読み込みエラー
	ビット13	予約
	ビット14	スキャンエラー
	ビット15	予約

		コントローラの状態
バイト2	ビット16	RUN中
	ビット17	I/O使用可/使用不可
	ビット18	強制変更有効/無効
	ビット19	PAUSE
	ビット20	予約
	ビット21～23	予約

バイト3	予約

8.2.13 #Time

#Timeはコントローラに設定されている「時分」をBCD4桁で示します。

変数タイプ: 整数

設定: コントローラ

読み込み専用

年、月、日、時分は以下のシステム変数で表されます。

例)2001年7月14日6時19分の場合

	年	月	日	時分
システム変数	#Year	#Month	#Day	#Time
値	1	7	14	619

8.2.14 #Version

#Version は、コントローラのバージョン番号を表します。#Version は、16 進数で表示されます。

変数タイプ: 整数

設定: コントローラ

読み込み専用

バイト番号	内容	V.1.0.0の場合
バイト3	メジャーバージョン	01
バイト2	マイナーバージョン	00
バイト1	予約	——
バイト0	予約	——

8.2.15 #Year

#Year はコントローラに設定されている「年」を BCD2 桁で示します。

変数タイプ: 整数

設定: コントローラ

読み込み専用

年、月、日、時分は以下のシステム変数で表されます。

例)2001 年 7 月 14 日 6 時 19 分の場合

	年	月	日	時分
システム変数	#Year	#Month	#Day	#Time
値	1	7	14	619

8.2.16 #Weekday

#Weekday は、現在の曜日を 0 ～ 6 の値で表示します。

変数タイプ: 整数

設定: コントローラ

読み込み専用

#Weekday は LS2054 の値を反映しています。LS2054 は日が変わる（23 時 59 分→ 00 時 00 分）タイミングで 0 ～ 6 の値を循環インクリメント（・・→ 5 → 6 → 0 → 1 →・・）しています。

#Weekday の値は 0 ～ 6 の範囲ですが、#Weekday の値と実際の曜日との対応付けは、ユーザー側で任意に決めてください。対応付けを行うには必ず表示器の画面上から LS2062（書き込み専用）に対し、0 ～ 6 の値を設定値表示器などで行ってください。

8.2.17 #FaultCode

#FaultCodeでは、最新のエラー状態が識別されます。

リセットですべて0にクリアされます。

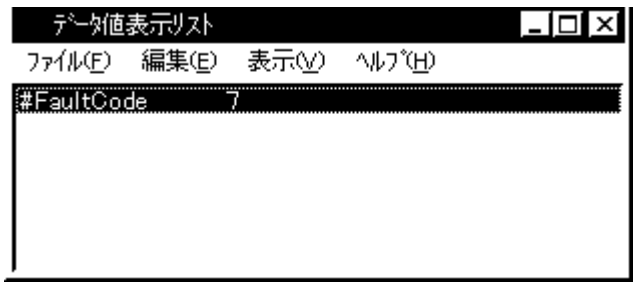
参照→12.2 エラーコード

変数タイプ: 整数

設定: コントローラ

読み込み専用

コード	程度	原因
0	正常	エラーはありません。
1	マイナー	算術命令の結果、または実数から整数への変換結果がオーバーフローしました。
2	メジャー	配列の領域を超えて参照されました。
3	メジャー	整数(32ビット)の範囲を超えたビットが参照されました。
4	メジャー	スタックがオーバーフローしました。
5	メジャー	不正な命令コードを使用しています。
6		システムで予約
7	メジャー	スキャンタイムがウォッチドックタイムを超えました。
8		システムで予約
9	メジャー	ソフトウェアのエラーです。場合によっては、システムをリブートし、回復する必要があります。
10		システムで予約
11		システムで予約
12	マイナー	BCD/BIN変換エラー
13	マイナー	ENCO/DECO変換エラー
14		システムで予約



[データ値表示リスト]ウィンドウに、#FaultCode 7が表示されています。

スキャンタイムがウォッチドッグタイムを超えたことを表しています。

8.2.18 #FaultRung

#FaultRungは、エラーが発生したラングの番号が格納されます。

エラーがないときは、0が格納されています。

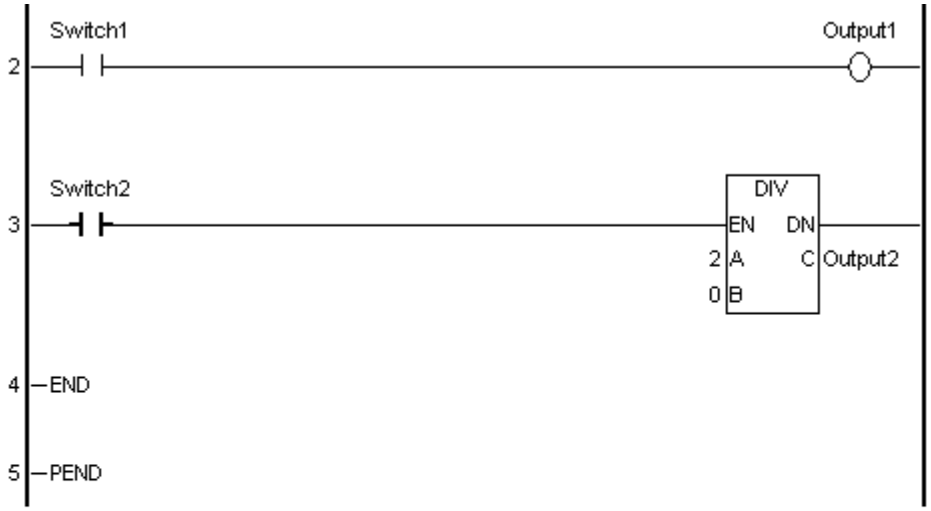
以下の例では、ラング3でエラーが発生したことが示されています。

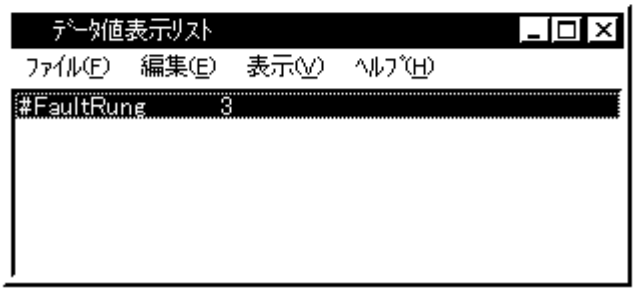
このエラーは、DIV命令を実行したときに、整数を0で除算したことが原因です。次のエラーが発生するか、またはコントローラをリセットするまで残ります。

変数タイプ: 整数

設定: コントローラ

読み込み専用





8.2.19 #IOFault

I/O ドライバで I/O エラーが発生したときに、#IOFault が ON になります。

このエラーは次のエラーが発生するか、またはコントローラをリセットするまで残ります。

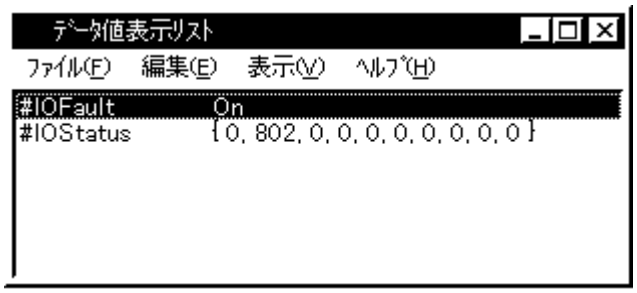
#IOStatus を確認すると、I/O ドライバの詳細な状態を知ることができます。

#IOFault が ON になると、[データ値表示リスト]ウィンドウに #IOFault が表示されます。

変数タイプ: ディスクリート

設定: コントローラ

読み込み専用



I/O ドライバのエラーコードの内容については、
「第 11 章 I/O ドライバ」をご覧ください。

8.2.20 #Overflow

#Overflow は、演算エラーが発生すると ON になり、次に算術命令または変換をするまで ON のままです。

演算エラーには、演算命令時のオーバーフロー、実数から整数への変換時のオーバーフロー、0 での除算などがあります。

演算エラーが発生すると、マイナー異常が発生します。**参照→12.2 エラーコード**

"ErrorHandler" サブルーチンがあれば実行されます。"ErrorHandler" サブルーチンはエラー処理サブルーチンです。"ErrorHandler" という名前であらかじめ作成しておく必要があります。

コントローラは #Fault をもとにして、実行を停止させることができます。

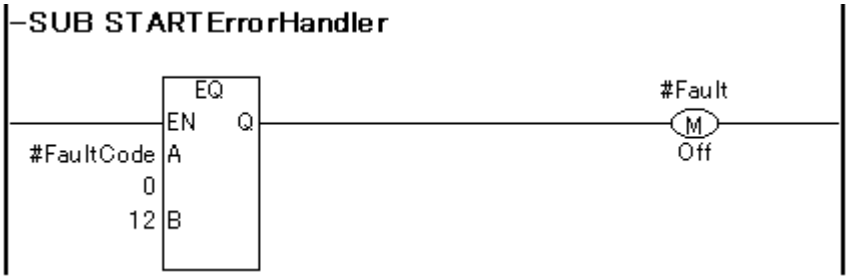
参照→ 8.2.23 #Fault

変数タイプ: ディスクリート

設定: コントローラ

読み込み専用

例)以下の "ErrorHandler" サブルーチンは BCD/BIN 変換エラーを検出して、ロジックプログラムの実行を停止させます。



実数から整数への変換でもオーバーフローが発生しない場合は、#Overflow は ON になりません。

8.2.21 #Command

#Commandはコントローラに対する制御コマンドとして使用される整数変数です。

コントローラは#Command を認識した後、0 にリセットします。複数のビットがON になっている場合、最下位ビットが優先されます。

変数タイプ: 整数

設定: ユーザー

初期値: OFF(全ビット)

書き込み可能

ビット0(=1)	コントローラのSTOP
ビット1(=2)	コントローラのRUN
ビット2(=4)	コントローラのリセット
ビット3(=8)	1スキャン実行
ビット4(=16)	続行
ビット5(=32)	PAUSE
ビット7(=128)	I/O使用可

8.2.22 #DisableAutoStart

ON にして電源を入れると、コントローラはSTOP モードで立ち上がります。

OFF にして電源を入れると、コントローラは前回の電断時の動作モード（RUN/STOP）で立ち上がります。

ただし、LT の初期設定で、コントローラ設定の「電源ON 時の動作モード」を「DEFAULT」に設定した場合のみ上記の設定が有効となります。

変数タイプ: ディスクリート

設定: ユーザー

初期値: OFF

書き込み可能

8.2.23 #Fault

#Fault は、"ErrorHandler" サブルーチンの終了時に、コントローラがロジックプログラムの実行を停止するか継続するかを判断する際に参照されます。#Fault をON にすることで、コントローラのロジックプログラムの実行を停止することが可能です。"ErrorHandler" サブルーチンについては、8.2.20 #Overflowを参照してください。

変数タイプ: ディスクリート

設定: ユーザー

初期値: OFF

書き込み可能

"ErrorHandler"サブルーチンがないときは、#Faultは意味を持ちません。

8.2.24 #FaultOnMinor

#FaultOnMinorは、"ErrorHandler"サブルーチンが存在しないロジックプログラム実行時にマイナー異常が発生した場合、コントローラがロジックプログラムの実行を停止するか継続するかを判断する際に参照されます。#FaultOnMinorをONにすることで、コントローラのロジックプログラムの実行を停止することが可能です。**参照→12.2 エラーコード**
"ErrorHandler"サブルーチンについては、8.2.18 #Overflowを参照してください。

変数タイプ: ディスクリート
設定: ユーザー
初期値: OFF
書き込み可能

8.2.25 #PercentAlloc

パーセントスキャンモードの場合に使用します。
#PercentAllocで、LTの総処理時間に対してコントローラが使用できる割り合いを設定します。スキャンタイムの値が10ms単位になるように設定してください。
#PercentAllocは、初期設定として設定するか、コントローラのRUN中にモニタリングモードで設定できます。通常は、[設定]ダイアログボックスで設定します。
参照→6.1.2 RUNモードの流れ

変数タイプ: 整数
設定: ユーザー
範囲： 0 ～ 50%
初期値: 50
書き込み可能

8.2.26 #Screen

#Screenで、LTの画面を切り替えます。「画面切り替え確認」機能によって#Screenの動作が以下のように異なります。

「画面切り替え確認」が有効の場合は#Screenに切り替え画面番号をセットした後、実際に画面が切り替わると0に初期化されます。**参照→「第1章 プログラムの作成」**

変数タイプ：整数
設定：ユーザー / コントローラ
初期値：0
書き込み専用

・ #Screenで設定された画面番号は表示させたい画面番号です。現在表示されている画面番号ではありませんのでご注意ください。
・ #Screenは、書き込み専用ですので画面が切り替わったことを判別するなどの用途には使用できませんのでご注意ください。

#Screenで画面切り替えを行う場合、画面切り替えはすべてロジックプログラム上の#Screenで行ってください。タッチ入力による画面切り替えも直接#Screenに書き込まず、下の例のようにロジックプログラムからの起動にて画面切り替えを行ってください。

- **電源投入直後の初期画面を#Screenにて変更する場合には約200ms以上待つか、LSS[0].x[3](LS2032のbit3)の立ち上がり(0→1)のタイミングで行ってください。**

8.2.27 #TargetScan

コンスタントスキャンモードの場合に使用します。

#TargetScanで、スキャンタイムを10ms単位で指定します。

ロジックタイムが一定の場合、#TargetScanの値を大きくすると、表示処理の処理時間を長くすることができます。また、#TargetScanの値を小さくすると、表示処理の時間が短くなります。これは、処理時間の大半をコントローラが使用するためです。

#TargetScanは、初期設定として設定するか、コントローラのRUN中にモニタリングモードで設定できます。通常は、[設定]ダイアログボックスで設定します。

参照→6.1.2 RUNモードの流れ

変数タイプ: 整数

設定: ユーザー

範囲: 10 ～ 2000ms

初期値: 10ms

書き込み可能

8.2.28 #WatchdogTime

#WatchdogTimeで、ウォッチドッグタイマの値をms単位で表示します。

#ScanTimeがこの値を超えると、メジャー異常が発生します。

参照→12.2 エラーコード

#WatchdogTimeは、起動中に設定値を変更しないでください。変更しても値は変化しますが設定値は変わりません。通常は、オフラインで設定するか、[設定]ダイアログボックスで設定します。

変数タイプ: 整数

設定: 本体起動前にユーザー設定

初期値: 500ms

読み込み専用

MEMO

このページは、空白です。
ご自由にお使いください。

第9章 命 令

ここでは、LT Editorで用いられる命令について説明します。

9.1 命令一覧

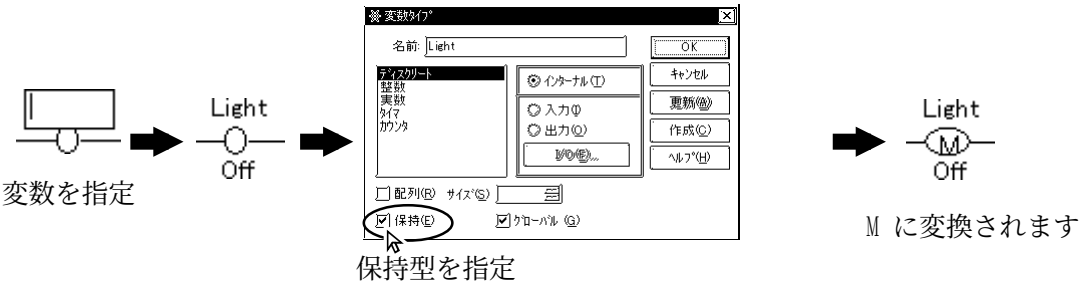
LT Editorでサポートしている命令を以下に示します。

■ディスクリート命令

命令	機能	シンボル	処理内容
NO	a接点	-| |-	論理演算開始（a接点演算開始）
NC	b接点	-|/|-	論理演算開始（b接点演算開始）
OUT/M*1	アウト・コイル/ 保持型コイル	-( )- / -(M)-	出力/保持型変数への出力
NEG/NM*1	反転コイル/ 保持型反転コイル	-(/)- / -(/M)-	反転出力/保持型変数への反転出力
SET/SM*1	セット・コイル/ 保持型セット・コイル	-(S)- / -(SM)-	セット/保持型変数へのセット
RST/RM*1	リセット・コイル/ 保持型リセットコイル	-(R)- / -(RM)-	リセット/保持型変数へのリセット
PT*2	立ち上がり接点	-|P|-	立ち上がりによる論理演算開始
NT*2	立ち下がり接点	-|N|-	立ち下がりによる論理演算開始

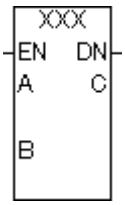
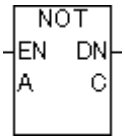
***1 これらの命令は、変数が保持型の場合に自動的に右側の命令（保持型命令）に変換されます。入力の際には左側の命令（非保持型命令）で入力してください。**

例）下図のようにOUT命令の変数を保持型に指定するとM命令に変換されます。

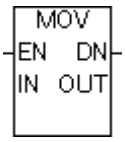
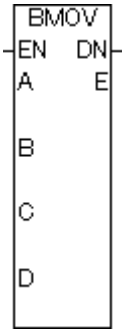
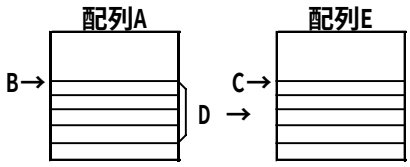
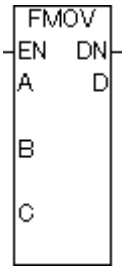
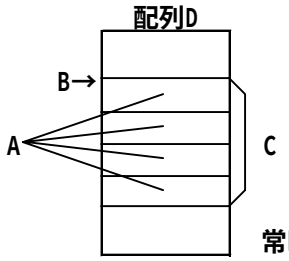


***2 PT/NT命令の最大数は2048個です。**

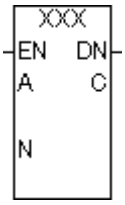
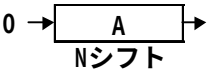
■論理演算命令

命令	機能	シンボル	処理内容
AND	論理積		A and B → C 常時導通
OR	論理和		A or B → C 常時導通
XOR	排他的論理和		A xor B → C 常時導通
NOT	ビット反転		$\overline{A}$ → C 常時導通

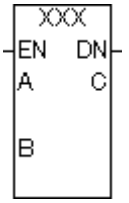
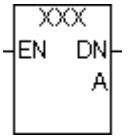
■転送命令

命令	機能	シンボル	処理内容
MOV	移動		IN → OUT 常時導通
BMOV	ブロック転送		 常時導通
FMOV	フィル転送		 常時導通

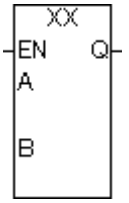
■シフト命令

命令	機能	シンボル	処理内容
ROL	左回転		 → C 常時導通
ROR	右回転		 → C 常時導通
SHL	左シフト		 → C 常時導通
SHR	右シフト		 → C 常時導通

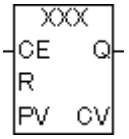
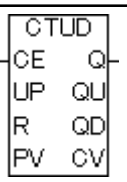
■演算命令

命令	機能	シンボル	処理内容
ADD	加算		A ＋ B → C　常時導通
SUB	減算		A － B → C　常時導通
MUL	乗算		A × B → C　常時導通
DIV	除算		A ÷ B → C　常時導通
MOD	剰余算		A % B → C　常時導通
INC	インクリメント		A ＋ 1 → A　常時導通
DEC	デクリメント		A － 1 → A　常時導通

■比較命令

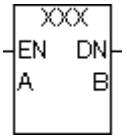
命令	機能	シンボル	処理内容
EQ	比較(=)		A ＝ B　のとき、導通
GT	比較(>)		A ＞ B　のとき、導通
LT	比較(<)		A ＜ B　のとき、導通
GE	比較(>=)		A ≧ B　のとき、導通
LE	比較(<=)		A ≦ B　のとき、導通
NE	比較(<>)		A <> B　のとき、導通

■タイマ / カウンタ命令

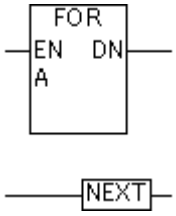
命令	機能	シンボル	処理内容
TON	オンディレータイマ		参照→9.2.34 TON(オンディレータイマ)
TOF	オフディレータイマ		参照→9.2.35 TOF(オフディレータイマ)
TP	パルスタイマ		参照→9.2.36 TP(パルスタイマ)
CTU	アップカウンタ		参照→9.2.37 CTU(アップカウンタ)
CTD	ダウンカウンタ		参照→9.2.38 CTD(ダウンカウンタ)
CTUD	アップダウンカウンタ		参照→9.2.39 CTUD(アップダウンカウンタ)

重要　・ タイマ命令には、スキャンタイムと同等の誤差を含みます。

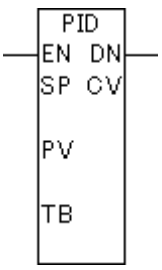
■変換命令

命令	機能	シンボル	処理内容
BCD	BCD変換		A → BCD変換 → B　　常時導通
BIN	バイナリ変換		A → バイナリ変換 → B 常時導通
ENCO	エンコード		A → エンコード変換 → B 常時導通
DECO	デコード		A → デコード変換 → B 常時導通

■プログラム制御命令

命令	機能	シンボル	処理内容
JMP	ジャンプ	->>ラベル名	ラベルの位置にジャンプ
JSR	ジャンプサブルーチン	->>サブルーチン名<<-	サブルーチンにジャンプ
RET	リターンサブルーチン	-<RETURN>-	呼び出されたJSR命令に戻る
FOR、NEXT	繰り返し		Aで指定された回数、FORとNEXTの間のロジックプログラムを繰り返し実行

■特殊命令*1

命令	機能	シンボル	処理内容
PID	PID演算		EN導通時 SPとPVをPID演算して、CVに出力 EN非導通時 TBとCVにMOV

*1 特殊命令の最大数は100個です。

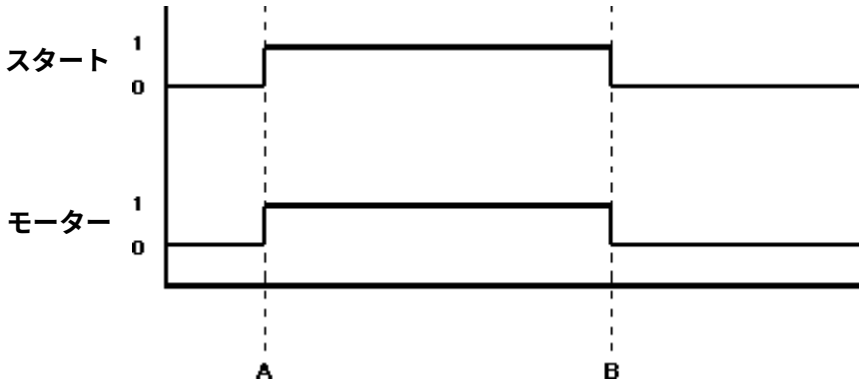
9.2 命令詳細

ここでは各命令の詳細を説明します。

9.2.1 NO(a 接点)

NO 命令を実行すると、変数が ON のときに導通します。

以下の例では、NO 命令の機能について説明しています。

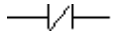


A　変数**スタート**が ON になると、変数**モーター**が ON になります。

B　変数**スタート**が OFF になると、変数**モーター**が OFF になります。

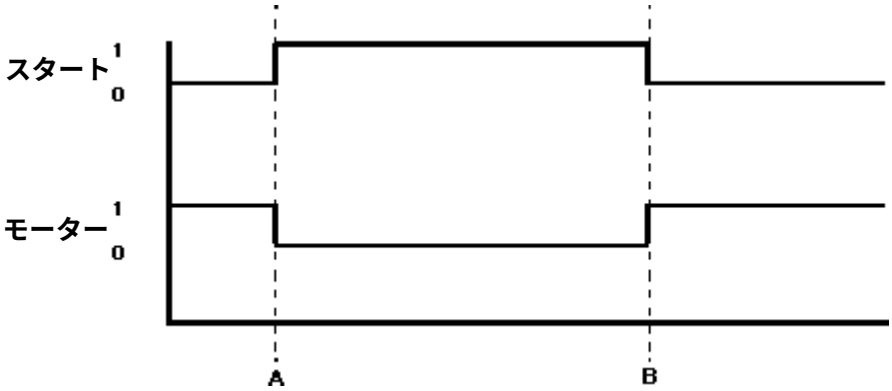
9.2.2 NC(b 接点)

変数



NC 命令を実行すると、変数が OFF のときに導通します。

以下の例では、NC 命令の機能について説明しています。

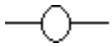


A　変数**スタート**が ON になると、変数**モーター**が OFF になります。

B　変数**スタート**が OFF になると、変数**モーター**が ON になります。

9.2.3 OUT/M(アウト・コイル)

変数

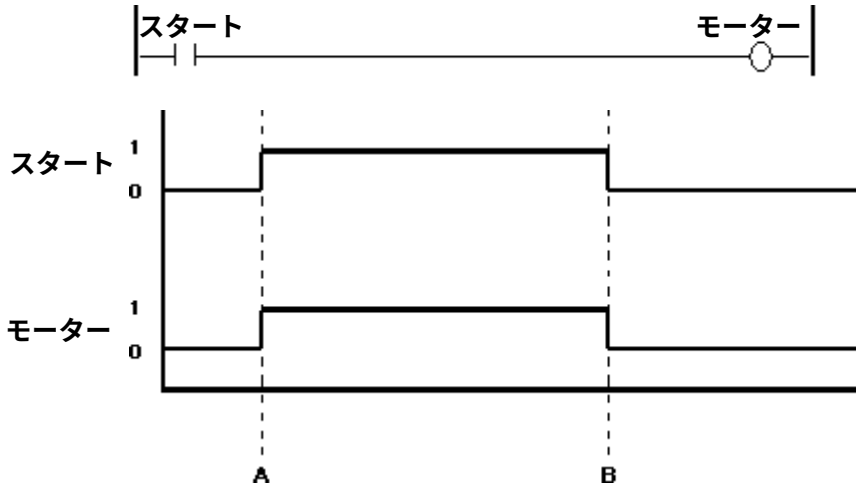


OUT 命令は、I/O に割り付けられた変数の ON/OFF や、内部メモリのディスクリート変数の ON/OFF に使用します。

この命令はコイル型出力命令なので、１ラングあたり１つだけ使用でき、出力命令の右側には他の命令を使用できません。出力命令は右側の母線のすぐ左に置きます。

OUT 命令に割り付けられた変数が、保持型であれば、下記の記号がロジックプログラムに表示されます。

以下の例では、OUT 命令の機能について説明しています。



A　変数**スタート**が ON になり、変数**モーター**が ON になります。

B　変数**スタート**が OFF になり、変数**モーター**が OFF になります。

OUT命令は非保持型変数のみ使用できます。保持型変数にはM（保持型コイル）命令を使用してください。

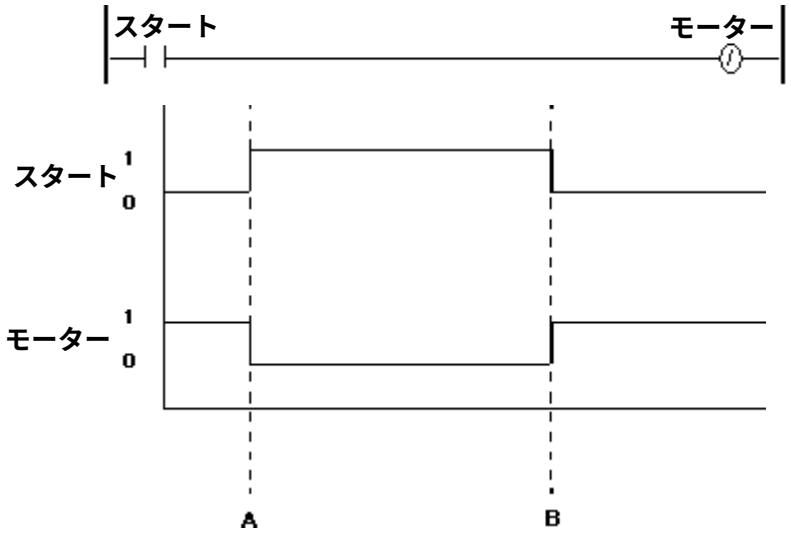
9.2.4 NEG(反転コイル)

変数

NEG 命令を実行すると、コイルに導通したとき変数が OFF になり、導通しないと ON になります。この命令はコイル型出力命令なので、１ラングあたり１つだけ使用でき、出力命令の右側には他の命令を使用できません。出力命令は右側の母線のすぐ左に置きます。

NEG 命令に割り付けられた変数が、保持型であれば、下記の記号がロジックプログラムに表示されます。

以下の例では、NEG 命令の機能について説明しています。



A　変数**スタート**が ON になり、変数**モーター**が OFF になります。

B　変数**スタート**が OFF になり、変数**モーター**が ON になります。

NEG 命令は非保持型変数のみ使用できます。

9.2.5 SET(セット・コイル)

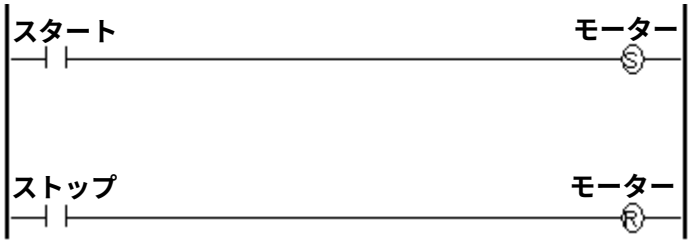
コイルに導通してからSET命令を実行すると、変数がONになります。

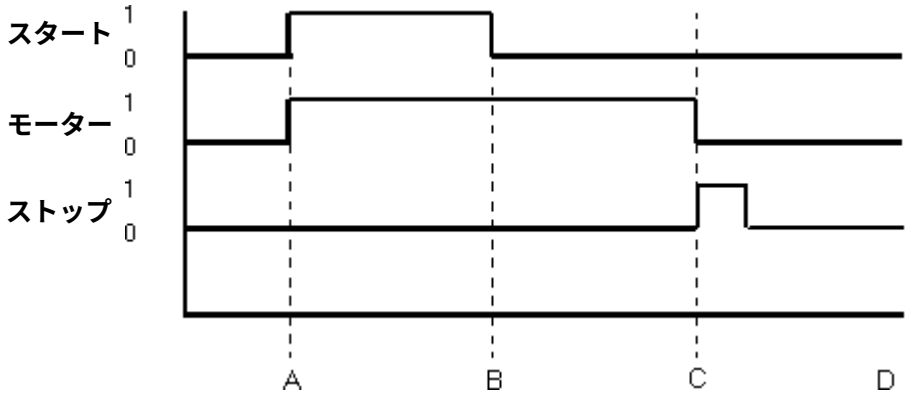
変数は、RST命令のような他の命令で確実にOFFにされるまで、ONのままです。

この命令はコイル型出力命令なので、１ラングあたり１つだけ使用でき、出力命令の右側には他の命令を使用できません。出力命令は右側の母線のすぐ左に置きます。

SET命令に割り付けられた変数が、保持型であれば、下記の記号がロジックプログラムに表示されます。

以下の例では、SET命令の機能について説明しています。





A　変数**スタート**がONになり、変数**モーター**がセットされます。

B　変数**スタート**がOFFになりますが、変数**モーター**には影響しません。

C　変数**ストップ**がONになり、変数**モーター**がリセットされます。

D　変数**スタート**がONになるまで、変数**モーター**はリセットのままです。

SET命令は非保持型変数のみ使用できます。保持型変数にはSM（保持型セット・コイル）命令を使用してください。

9.2.6 RST(リセット・コイル)

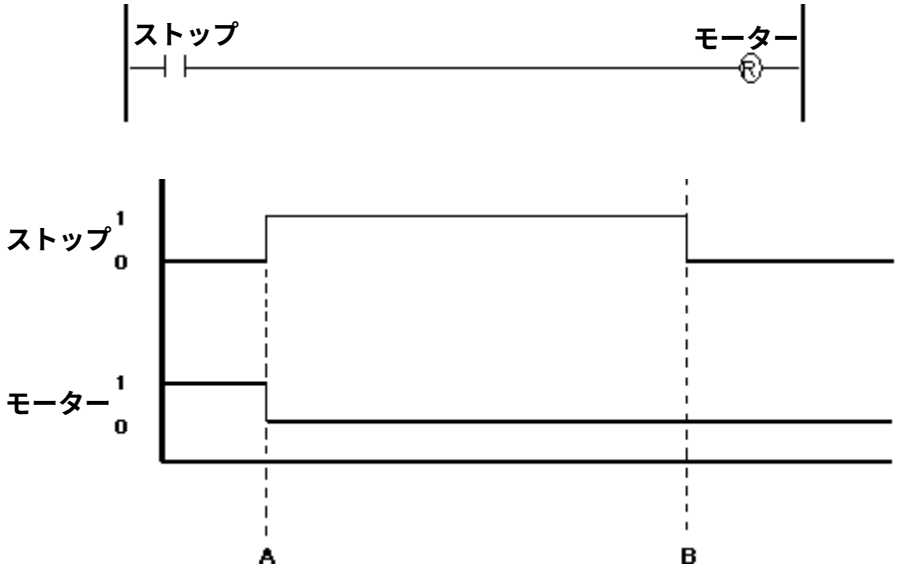
RST 命令を実行すると、SET 命令などで ON された変数が OFF になります。

変数は、SET 命令のような他の命令で確実に ON にされるまで OFF のままです。

この命令はコイル型出力命令なので、１ラングあたり１つだけ使用でき、出力命令の右側には他の命令を使用できません。出力命令は右側の母線のすぐ左に置きます。

RST 命令に割り付けられた変数が、保持型であれば、下記の記号がロジックプログラムに表示されます。

以下の例では、RST 命令の機能について説明しています。



A　変数**ストップ**が ON になると、変数**モーター**がリセットされます。

B　変数**ストップ**が OFF になっても、RST 命令でリセットされた変数**モーター**は、他の命令で ON にされるまで OFF のままです。

・RST命令は非保持型変数のみ使用できます。保持型変数にはRM(保持型リセット・コイル)命令を使用してください。
・実数、整数変数をRST命令でリセットする(0にする)ことはできません。

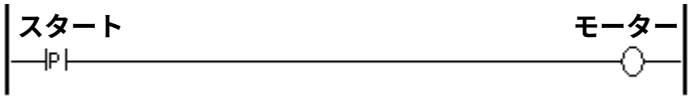
9.2.7 PT(立ち上がり接点)

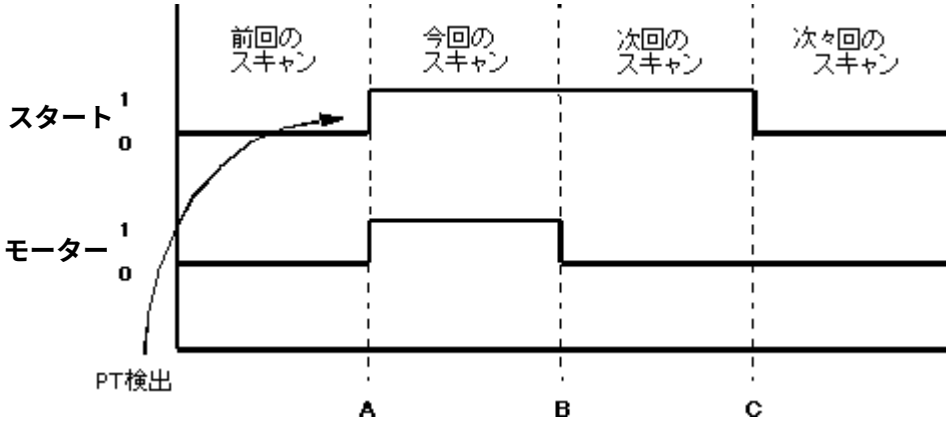
変数

PT 命令を実行すると、前回のスキャンで OFF の変数が今回のスキャンで ON になったときに、1スキャンの間だけ導通します。

スタートアップ時には、前回のスキャンでの立ち上がり接点の状態は OFF とされます。

以下の例では、PT 命令の機能について説明しています。





A　変数**スタート**が ON になり、変数**モーター**が ON になります。

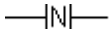
B　スキャンを 1 回すると、変数**モーター**は OFF になります。

C　変数**スタート**の立ち上がりを検出しなかったので、変数**モーター**は OFF のままです。

・ PT（立ち上がり接点）命令および NT（立ち下がり接点）命令のオペランドに対し、配列やビット指定などの各要素に変数を用いて間接アドレッシングを行う場合には注意が必要です。
前回実行時のオペランドに設定された変数と今回実行時のオペランドに設定された変数の状態を比較して実行しますので、それぞれの指定する変数値が異なる場合は状態比較の対象が異なります。

9.2.8 NT(立ち下がり接点)

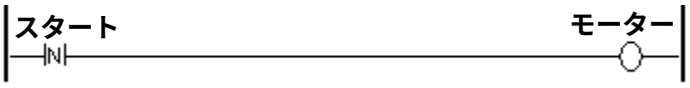
変数

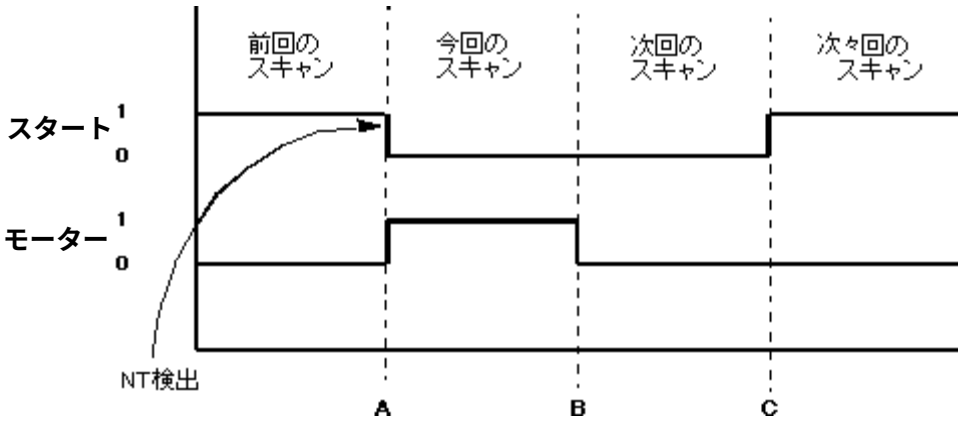


NT命令を実行すると、前回のスキャンでONの変数が今回のスキャンでOFFになったときに、1スキャンの間だけ導通します。

最初にスキャンするときは、前回のスキャンでの状態はOFFとされるので、NT命令を実行しても導通しません。

以下の例では、NT命令の機能について説明しています。





A　変数**スタート**がOFFになり、変数**モーター**がONになります。

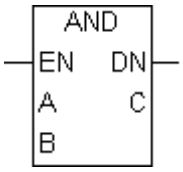
B　スキャンを1回すると、変数**モーター**がOFFになります。

C　変数**スタート**の立ち下がりを検出しなかったので、変数**モーター**はOFFのままです。

重要

- **NT（立ち下がり接点）命令およびPT（立ち上がり接点）命令のオペランドに対し、配列やビット指定などの各要素に変数を用いて間接アドレッシングを行う場合には注意が必要です。**
  前回実行時のオペランドに設定された変数と今回実行時のオペランドに設定された変数の状態を比較して実行しますので、それぞれの指定する変数値が異なる場合は状態比較の対象が異なります。

9.2.9 AND(論理積)



AND 命令を実行すると、A と B のビットが ON のときのみ、C のビットが ON になります。

これ以外のときは、C のビットは OFF になります。

A	演算子	B	C
ON	AND	ON	ON
ON		OFF	OFF
OFF		ON	OFF
OFF		OFF	OFF

整数 A　0 1 1 0 ... 1 1 0 0

整数 B　1 1 0 0 ... 0 0 0 1

整数 C　0 1 0 0 ... 0 0 0 0

AND 命令は常に導通します。

AND 命令が実行できる A、B、C の組み合わせは以下の通りです。

Aのタイプ	Bのタイプ	Cのタイプ
整数	整数	整数
整数配列	整数配列	整数配列
整数	整数定数	整数
整数配列	整数定数	整数配列

AND 命令には、3 種類あります。

1. すべての変数が配列でないときは、32 ビットの単純な論理積が演算されます。

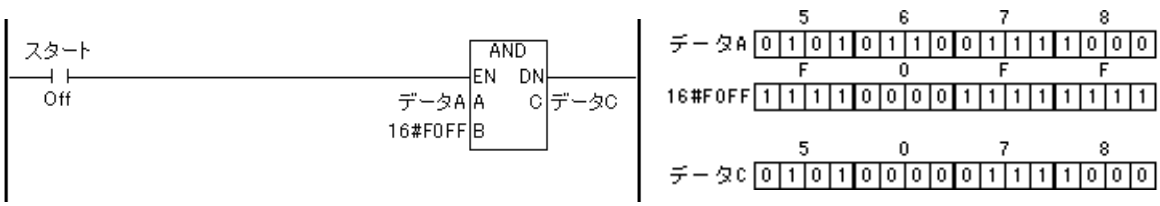
2.A と C が配列で、B が整数配列でないときは、A のそれぞれの要素と B を論理積演算し、結果は、C の対応するそれぞれの要素に格納されます。A と C の配列は、同じサイズにしてください。

3.3 つの変数が同じサイズの配列のときは、配列 A と配列 B の論理積が演算されます。結果は配列 C に格納されます。

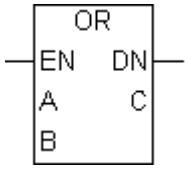
動作例

スタートが ON すると、**データ A** の BCD4 桁データの 3 桁目を ”0 ” にマスクし、その結果が **データ C** に格納されます。

例)**データ A** が ” 16#5678 ”(16 進数の 5678)の場合、**データ C** には ” 16#5078 ” が格納されます。



9.2.10 OR(論理和)



OR 命令を実行すると、A または B のビットが ON のときは、C のビットが ON になります。

それ以外のときは、C のビットが OFF になります。

A	演算子	B	C
ON	OR	ON	ON
ON		OFF	ON
OFF		ON	ON
OFF		OFF	OFF

整数 A	0	1	1	0	...	1	1	0	0
整数 B	1	1	0	0	...	0	0	0	1
整数 C	1	1	1	0	...	1	1	0	1

OR 命令は、常に導通します。

OR 命令が実行できる A、B、C の組み合わせは以下の通りです。

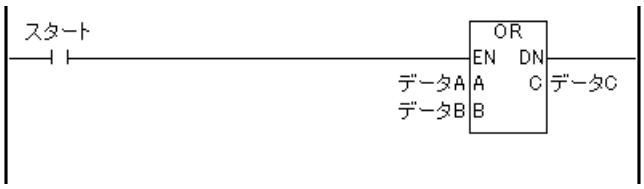
Aのタイプ	Bのタイプ	Cのタイプ
整数	整数	整数
整数配列	整数配列	整数配列
整数	整数定数	整数
整数配列	整数定数	整数配列

OR 命令には 3 種類あります。

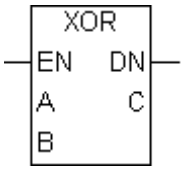
1. **A、B の変数が両方とも整数のときは、32 ビットの単純な論理和が演算されます。**
2. **A と C が配列で、B が整数配列でないとき、A のそれぞれの要素と B を論理和演算し、結果は、C の対応するそれぞれの要素に格納されます。A と C の配列は同じサイズにしてください。**
3. **3 つの変数が同じサイズの配列のときは、配列 A と配列 B の論理和が演算されます。結果は、配列 C に格納されます。**

◆動作例

スタートが ON すると、**データ A** と**データ B** の論理和が**データ C** に格納されます。



9.2.11 XOR(排他的論理和)



XOR 命令を実行すると、A か B の一方のビットが ON のときは、C のビットが ON になります。
それ以外のときは、C のビットが OFF になります。

A	演算子	B	C
ON	XOR	ON	OFF
ON		OFF	ON
OFF		ON	ON
OFF		OFF	OFF

整数 A	0	1	1	0	...	1	1	0	0
整数 B	1	1	0	0	...	0	0	0	1
整数 C	1	0	1	0	...	1	1	0	1

XOR 命令は、常に導通します。

XOR 命令が実行できる A、B、C の組み合わせは以下の通りです。

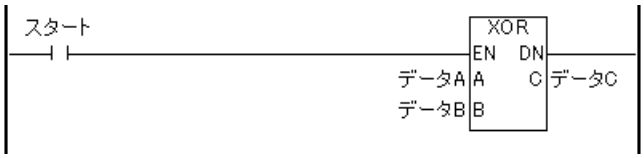
Aのタイプ	Bのタイプ	Cのタイプ
整数	整数	整数
整数配列	整数配列	整数配列
整数	整数定数	整数
整数配列	整数定数	整数配列

XOR 命令には、以下の 3 種類があります。

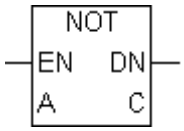
1. **A、B の変数が両方とも整数のときは、32 ビットの単純な排他的論理和が演算されます。**
2. **A と C が配列で、B が整数配列でないとき、A のそれぞれの要素と B を排他的論理和演算し、結果は、C の対応するそれぞれの要素に格納されます。A と C の配列は同じサイズにしてください。**
3. **3 つの変数が同じサイズの配列のときは、配列 A と配列 B の排他的論理和が演算されます。結果は、配列 C に格納されます。**

◆動作例

スタートが ON すると、**データ A** と**データ B** の排他的論理和が**データ C** に格納されます。



9.2.12 NOT(ビット反転)



NOT 命令を実行すると、A のビットが OFF のときには C のビットが ON になり、A のビットが ON のときには、C のビットが OFF になります。

A	演算子	C
ON	NOT	OFF
OFF		ON

整数 A

0	1	1	0	...	1	1	0	0

整数 C

1	0	0	1	...	0	0	1	1

NOT 命令は、常に導通します。

NOT 命令が実行できる A、C の組み合わせは以下の通りです。

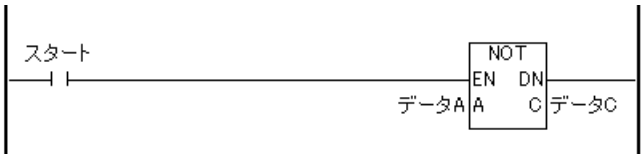
Aのタイプ	Cのタイプ
整数	整数
整数配列	整数配列

NOT 命令には 2 種類あります。

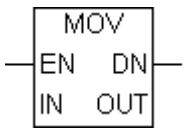
1. **A の変数が整数のときは、32 ビットの単純なビット反転が演算されます。**
2. **A の変数が配列のときは、A の配列全体のビット反転が演算されます。結果は、C に格納されます。このとき、A と C は同じサイズにしてください。**

◆動作例

スタートが ON すると、**データ A** の値がビット反転されて、**データ C** に格納されます。



9.2.13 MOV(転送)



MOV 命令を実行すると、IN は OUT にコピーされます。

IN と OUT の変数タイプが違うときは、結果は OUT の変数タイプに変換されます。

配列を転送するときは、IN と OUT を同じ変数タイプ・サイズにしてください。

この命令は常に導通します。MOV 命令が実行できる IN、OUT の組み合わせは次項の通りです。

INのタイプ	OUTのタイプ
ディスクリート配列	INと同じサイズのディスクリート配列
整数	整数または実数の変数または配列
整数配列	INと同じサイズの整数配列または変数
整数定数	整数または実数の変数または配列
実数	整数または実数の変数または配列
実数配列	INと同じサイズの実数配列または変数
実数定数	整数または実数の変数または配列

実数から整数に変換したときに、値が大きすぎて転送できないときは、#Overflowが ONになります。このとき、結果は不定です。

以下の例では、MOV 命令の使用例について説明しています。

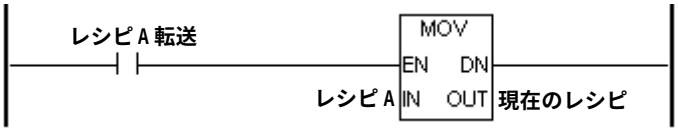
例 1: 変数をクリアする

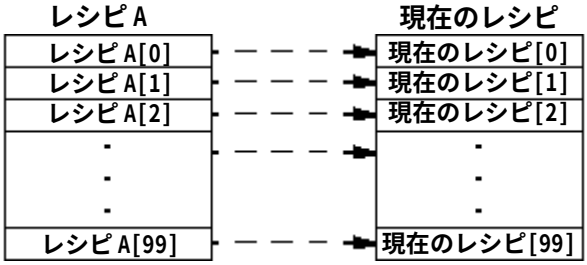
変数をクリアするときは、その変数に 0 を転送すると簡単です。

例 2: 配列をブロック転送をする

ブロック転送をするときは、同じタイプの 2 つの配列を指定して転送すると簡単です。

例えば、要素数 100 の配列**レシピ A** を、同じタイプ、同じサイズの配列**現在のレシピ**に転送するときは、配列**レシピ A** を 1 回の MOV 命令で転送すると簡単です。





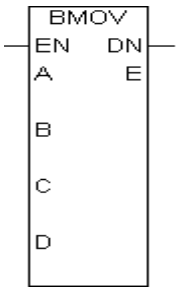
ロジックプログラム中で、配列全体を表現する場合は変数名のみ記述してください。

例) ○　レシピ A

×　レシピ A[*]

×　レシピ A[100]

9.2.14 BMOV(ブロック転送)



A:転送元配列変数名
B:配列A[B]より開始
C:配列E[C]へ
D:転送データ数
E:転送先配列変数名

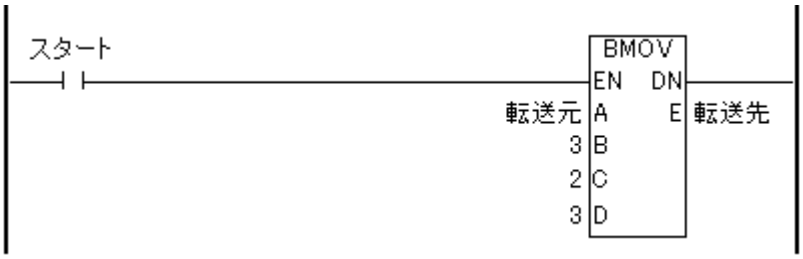
BMOV命令を実行すると、ある一つの配列の要素のいくつかを、別の配列の要素のいくつかにコピーすることができます。すなわち、配列A[B]から始まるD個の要素が、配列E[C]から始まるD個の要素にコピーされます。

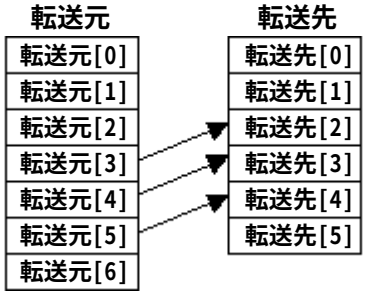
整数配列に対してのみ有効です。配列は、違うサイズでも転送できます。この命令は常に導通します。BMOV命令が実行できるA、B、C、D、Eはそれぞれ以下の通りです。

A、Eのタイプ	B、C、Dのタイプ
整数配列	整数
	整数定数

◆動作例

要素数7の整数配列**転送元[3]**、**転送元[4]**、**転送元[5]**を要素数6の整数配列**転送先[2]**、**転送先[3]**、**転送先[4]**にコピーする場合、以下のようになります。





転送元[3]は、**転送先[2]**にコピーされます。

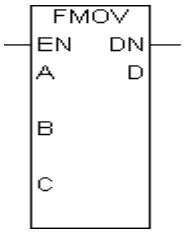
転送元[4]は、**転送先[3]**にコピーされます。

転送元[5]は、**転送先[4]**にコピーされます。

コントローラはRUN中にBMOV命令で配列AやEの存在しない要素を参照していないかどうかをチェックします。無効な配列の参照が行われた場合、メジャー異常が発生し、#Faultcodeに2が設定されます。

参照→8.2.17 #Faultcode

9.2.15 FMOV(フィル転送)



A:転送元データ
B:配列 D[B]より開始
C:転送データ数
D:転送先配列変数名

FMOV 命令を実行すると、配列 D[B]の要素から C 個分に A を格納します。

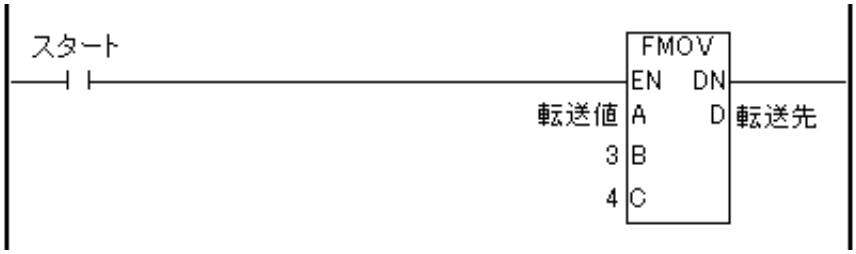
整数配列に対してのみ、有効です。この命令は常に導通します。

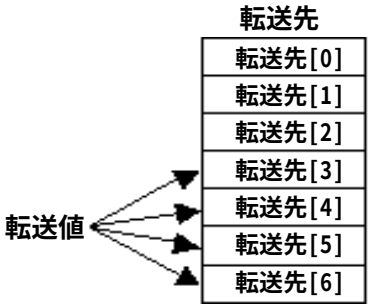
FMOV 命令が実行できる A、B、C、D はそれぞれ以下の通りです。

A、B、Cのタイプ	Dのタイプ
整数	整数配列
整数定数	

◆動作例

要素数 7 の整数配列**転送先[3]**、**転送先[4]**、**転送先[5]**、**転送先[6]**に**転送値**の値をコピーする場合、以下のようになります。





転送値は、**転送先[3]**にコピーされます。

転送値は、**転送先[4]**にコピーされます。

転送値は、**転送先[5]**にコピーされます。

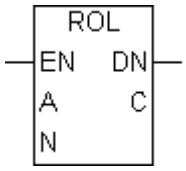
転送値は、**転送先[6]**にコピーされます。

コントローラは RUN 中に FMOV 命令で配列 D の存在しない要素を参照していないかどうかをチェックします。無効な配列の参照が行われた場合、メジャー異常が発生し、

#Faultcode に 2 が設定されます。

参照→8.2.17 #Faultcode

9.2.16 ROL(左回転)



A:回転させる変数名
N:シフトビット数
C:格納先変数名

ROL 命令を実行すると、A のビットが N ビット左方向にシフトします。

左端のビット（最上位ビット）は、右端のビット（最下位ビット）にローテーションします。結果は C に格納されます。

この命令は常に導通します。

ROL 命令が実行できる A、N、C の組み合わせは以下の通りです。

Aのタイプ	Nのタイプ	Cのタイプ
整数	整数または整数定数	整数
整数配列	整数または整数定数	Aと同じサイズの配列
整数定数	整数または整数定数	整数

ROL 命令には 2 種類あります。

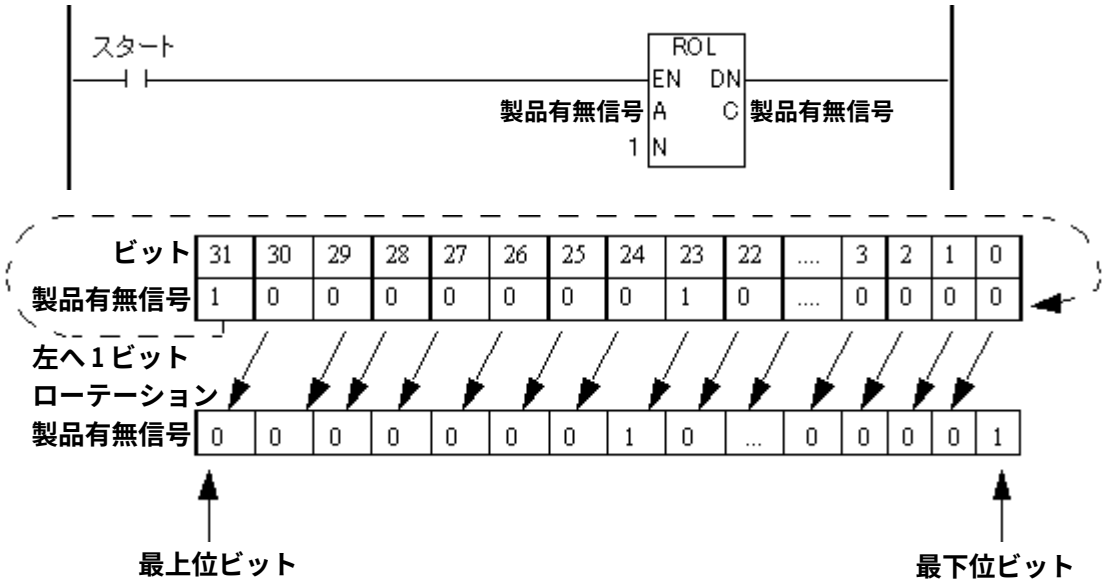
1. **A と C が整数のときは、単純に 32 ビットローテーションされます。N は、0 以上 31 以下にしてください。**
2. **A と C が同じサイズの配列のときは、A は大きなブロックとして扱われます。ビットは要素から次の要素へシフトされます。それぞれの要素内のみでローテーションするわけではありません。N は 0 以上（32 × 配列サイズ - 1）以下にしてください。**

N が範囲外のときは、#Overflow が ON になります。このとき、結果は不定です。

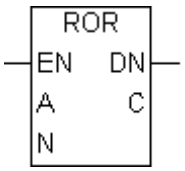
参照→ 8.2.20 #Overflow

◆動作例

以下の例では、**製品有無信号**を使用した 1 ビットローテーションについて説明しています。



9.2.17 ROR(右回転)



A:回転させる変数名
N:シフトビット数
C:格納先変数名

ROR 命令を実行すると、A のビットが N ビット右方向にシフトされます。

右端のビット（最下位ビット）は、左端のビット(最上位ビット)にローテーションします。

結果は C に格納されます。

この命令は常に導通します。

ROR 命令が実行できる A、N、C の組み合わせは以下の通りです。

Aのタイプ	Nのタイプ	Cのタイプ
整数	整数または整数定数	整数
整数配列	整数または整数定数	Aと同じサイズの配列
整数定数	整数または整数定数	整数

ROR 命令には 2 種類あります。

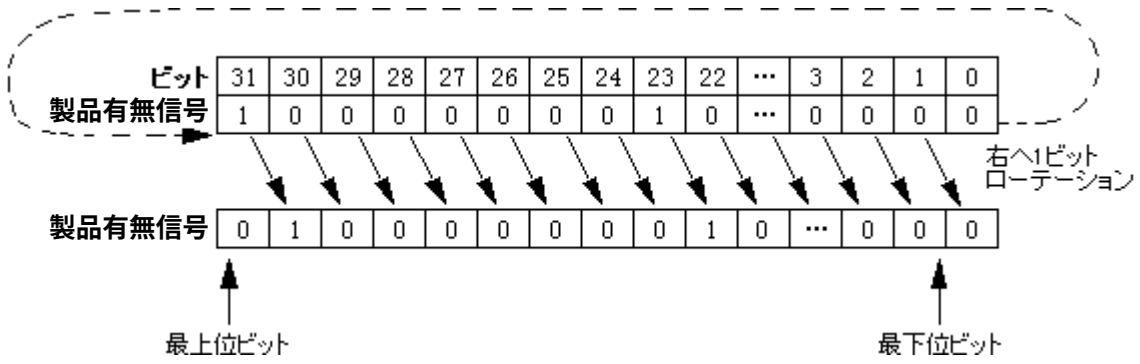
1. **A と C が両方とも配列でないときは、単純に 32 ビットローテーションされます。N は 0 以上 31 以下にしてください。**
2. **A と C が同じサイズの配列のときは、A は大きなブロックとして扱われます。ビットは要素から次の要素へシフトされます。それぞれの要素内のみでローテーションするわけではありません。N は 0 以上（32 ×配列サイズ -1）以下にしてください。**

N が範囲外のときは、#Overflow が ON になります。このとき、結果は不定です。

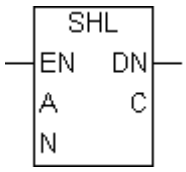
参照→ 8.2.20 #Overflow

◆動作例

以下の例では、**製品有無信号**を使用した１ビットローテーションについて説明しています。



9.2.18 SHL(左シフト)



A:回転させる変数名
N:シフトビット数
C:格納先変数名

SHL 命令を実行すると、A のビットが N ビット左方向にシフトします。

1 ビットシフトするたびに、左端のビット（最上位ビット）は失われ、右端の空ビットには 0 が格納されます。結果は C に格納されます。

この命令は常に導通します。

SHL 命令が実行できる A、N、C の組み合わせは以下の通りです。

Aのタイプ	Nのタイプ	Cのタイプ
整数	整数または整数定数	整数
整数配列	整数または整数定数	Aと同じサイズの配列
整数定数	整数または整数定数	整数

SHL 命令には 2 種類あります。

1. **A と C が両方とも配列でないときは、単純に 32 ビットシフトします。N は 0 以上 31 以下にしてください。**
2. **A と C が同じサイズの配列のときは、A は大きな整数として扱われます。ビットは要素から次の要素へシフトされます。それぞれの要素の左端のビットは失われません。ただし、最後尾の要素の左端ビットは失われます。N は 0 以上（32 × 配列サイズ - 1）以下にしてください。**

N が範囲外のときは、#Overflow が ON になります。このとき、結果は不定です。

参照→ 8.2.20 #Overflow

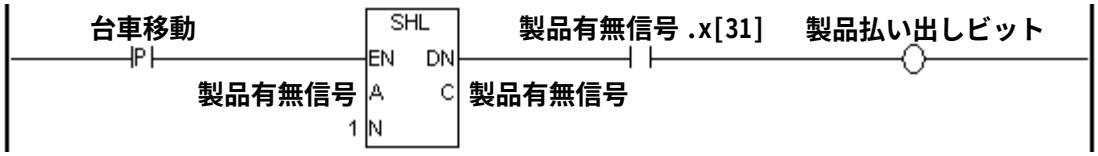
◆動作例

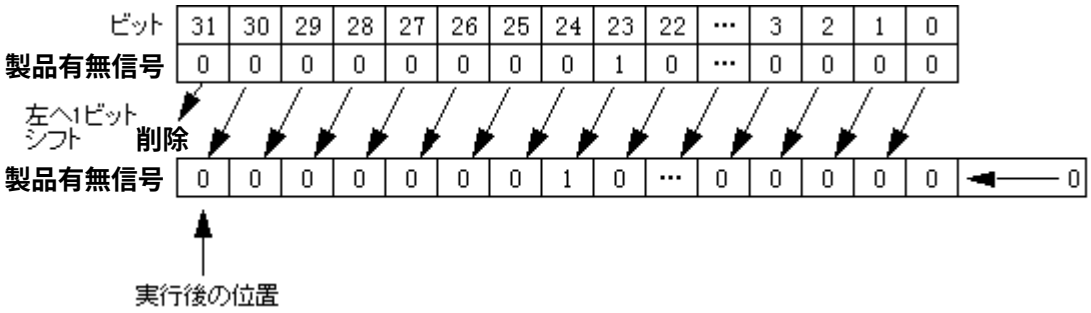
以下の例では、1 ビット左シフトを実行してビットの位置を把握する方法について説明しています。

製品有無信号のそれぞれのビットは、実際の製品位置を表します。

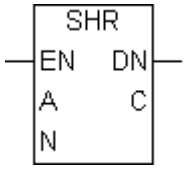
台車移動が ON すると、ビットは次の位置へ左シフトします。

ビットが変数の最終ビット位置（31）にくると、**製品払い出しビット**が ON になり、動作が終了したことがわかります。


台車移動
P
SHL
EN
DN
製品有無信号
A
C
製品有無信号
1
N
製品有無信号 .x[31]
製品払い出しビット


ビット
31 30 29 28 27 26 25 24 23 22 … 3 2 1 0
製品有無信号
0 0 0 0 0 0 0 0 1 0 … 0 0 0 0
左へ1ビット
シフト
削除
製品有無信号
0 0 0 0 0 0 0 1 0 … 0 0 0 0 0
0
実行後の位置

9.2.19 SHR(右シフト)



A:回転させる変数名
N:シフトビット数
C:格納先変数名

SHR 命令を実行すると、A のビットが N ビット右方向にシフトします。

1 ビットシフトするたびに右端のビット（最下位ビット）は失われ、左端の空ビットには 0 が格納されます。結果は C に格納されます。

この命令は常に導通します。

SHR 命令が実行できる A、N、C の組み合わせは以下の通りです。

Aのタイプ	Nのタイプ	Cのタイプ
整数	整数または整数定数	整数
整数配列	整数または整数定数	Aと同じサイズの配列
整数定数	整数または整数定数	整数

SHR 命令には 2 種類あります。

1. **A と C が配列でないときは、単純に 32 ビットシフトします。N は 0 以上 31 以下にしてください。**
2. **A と C が同じサイズの配列のときは、A は大きな整数として扱われます。ビットは要素から次の要素へシフトされます。それぞれの要素の右端のビットは失われません。ただし、最初の要素の右端ビットは失われます。N は 0 以上（32 ×配列サイズ -1）以下にしてください。**

N が範囲外のときは、#Overflow が ON になります。このとき、結果は不定です。

参照→ 8.2.20 #Overflow

◆動作例

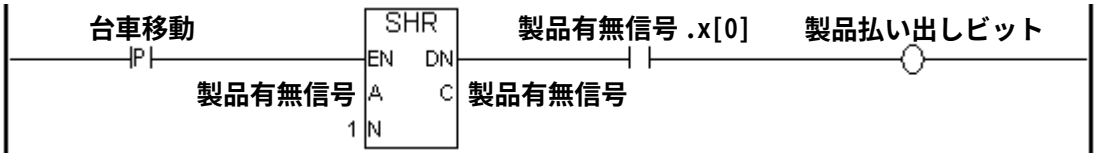
・ビットを扱う場合

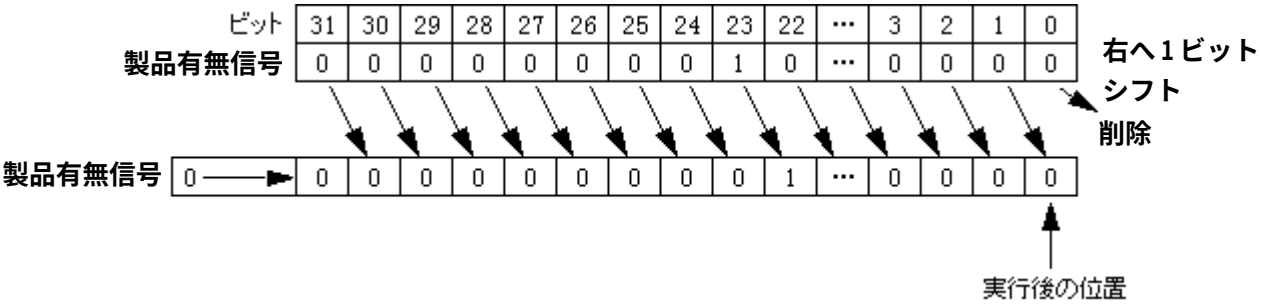
1ビット右シフトを実行してビットの位置を把握する方法について説明しています。

製品有無信号のそれぞれのビットは、実際の製品位置を表します。

台車移動が ON すると、ビットは次の位置へ右シフトします。

ビットが変数の最終ビット位置（0）にくると、**製品払い出しビット**が ON になり、動作が終了したことがわかります。





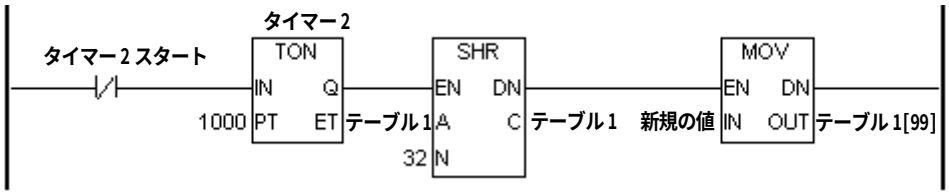
・配列を扱う場合

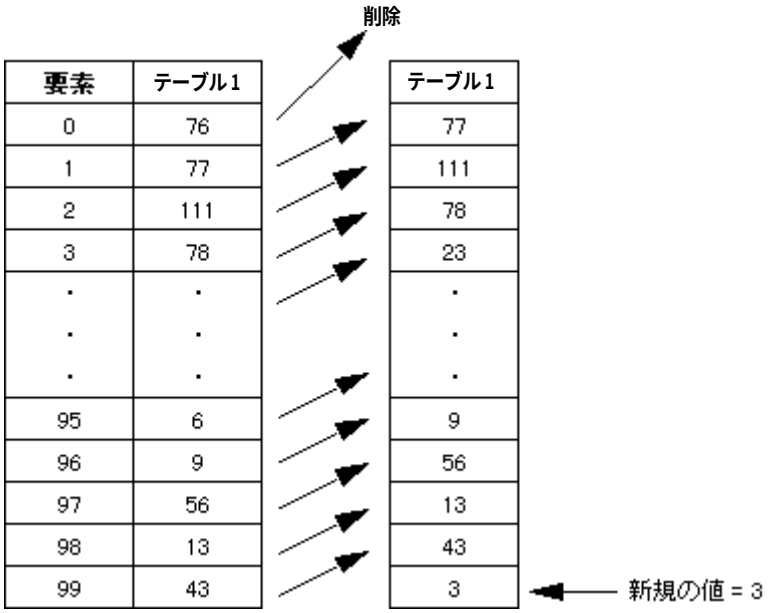
SHR命令を使用して整数配列の中で各要素の値を移動することについて説明しています。

32 ビットシフトでは、32 ビット整数全体がシフトします。

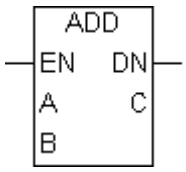
１秒ごとに、整数配列**テーブル 1** の整数値が要素 0 の方向に１つずつ移動します。

新規の値は、整数配列**テーブル 1** の最後の要素（**テーブル 1[99]**）に格納されます。





9.2.20 ADD(加算)



A:データ
B:データ
C:格納先変数名

ADD 命令を実行すると、A と B が加算され、結果が C に格納されます。

A と B の両方が整数または整数定数のとき、ADD 命令は整数加算をします。

それ以外は、浮動小数点加算されますが、処理速度が遅くなる場合があります。

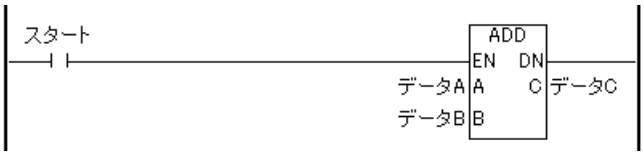
この命令は常に導通します。ADD 命令が実行できる A、B、C はそれぞれ以下の通りです。

Aのタイプ	Bのタイプ	Cのタイプ
整数	整数	整数または実数
整数定数	整数定数	整数または実数
実数	実数	整数または実数
実数定数	実数定数	整数または実数

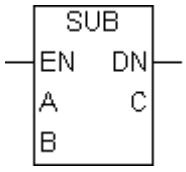
・結果CがCの変数タイプで表現できる範囲を超えるときは、#Overflow が ON になり、加算の結果は不定です。
参照→ 8.2.20 #Overflow
・AまたはBのどちらかが実数のときは、AとBの両方を実数に変換して加算します。ただし、Cが整数のときは、結果はC に格納されるので、小数点以下は切り捨てられます。

◆動作例

スタートが ON すると、**データ A** と**データ B** が加算され、計算結果が**データ C** に格納されます。



9.2.21 SUB(減算)



A:データ
B:データ
C:格納先変数名

SUB 命令を実行すると、A から B が減算され、結果が C に格納されます。

A と B のタイプが整数または整数定数のときは、SUB 命令は整数減算をします。

それ以外は、浮動小数点減算されますが、処理速度が遅くなる場合があります。

この命令は常に導通します。SUB 命令が実行できる A、B、C はそれぞれ次項の通りです。

Aのタイプ	Bのタイプ	Cのタイプ
整数	整数	整数または実数
整数定数	整数定数	整数または実数
実数	実数	整数または実数
実数定数	実数定数	整数または実数

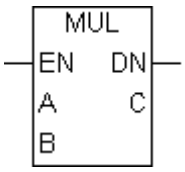
・結果CがCの変数タイプで表現できる範囲を超えるときは、#Overflowが ONになり、減算の結果は不定です。
参照→ 8.2.20 #Overflow

・AまたはBのどちらかが実数のときは、AとBの両方を実数に変換して減算します。ただし、Cが整数のときは、結果はCに格納されるので、小数点以下は切り捨てられます。

◆動作例

スタートが ONすると、**データA**から**データB**が減算され、計算結果が**データC**に格納されます。

9.2.22 MUL(乗算)



A:データ
B:データ
C:格納先変数名

MUL命令を実行すると、AとBが乗算され、結果がCに格納されます。

AとBの両方が整数または整数定数のとき、整数乗算されます。

それ以外は、浮動小数点乗算されますが、処理速度が遅くなる場合があります。

この命令は常に導通します。MUL命令が実行できるA、B、Cはそれぞれ以下の通りです。

Aのタイプ	Bのタイプ	Cのタイプ
整数	整数	整数または実数
整数定数	整数定数	整数または実数
実数	実数	整数または実数
実数定数	実数定数	整数または実数

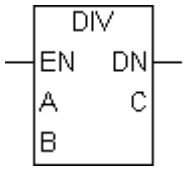
・結果CがCの変数タイプで表現できる範囲を超えるときは、#Overflowが ONになり、乗算の結果は不定です。
参照→ 8.2.20 #Overflow

・AまたはBのどちらかが実数のときは、AとBの両方を実数に変換して乗算します。ただし、Cが整数のときは、結果はCに格納されるので、小数点以下は切り捨てられます。

◆動作例

スタートがONすると、**データA**と**データB**が乗算され、計算結果が**データC**に格納されます。

9.2.23 DIV(除算)



A:データ
B:データ
C:格納先変数名

DIV命令を実行すると、AがBで除算され、結果はCに格納されます。

AとBの両方が整数または整数定数のときは、整数除算されます。

それ以外は、浮動小数点除算されますが、処理速度が遅くなる場合があります。

この命令は常に導通します。DIV命令が実行できるA、B、Cはそれぞれ以下の通りです。

Aのタイプ	Bのタイプ	Cのタイプ
整数	整数	整数または実数
整数定数	整数定数	整数または実数
実数	実数	整数または実数
実数定数	実数定数	整数または実数

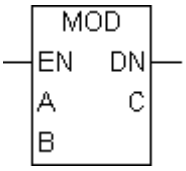
・Bが0のときや、結果Cが Cの変数タイプで表現できる範囲を超えるときは、#OverflowがONになり、除算の結果は不定です。
参照→ 8.2.20 #Overflow

・AまたはBのどちらかが実数のときは、AとBの両方を実数に変換して除算します。ただし、Cが整数のときは、結果はCに格納されるので、小数点以下は切り捨てられます。

◆動作例

スタートがONすると、**データA**が**データB**で除算され、計算結果が**データC**に格納されます。

9.2.24 MOD(剰余算)



A:データ
B:データ
C:格納先変数名

MOD 命令を実行すると、A が B で除算され、余りが C に格納されます。A、B が整数または整数定数のときだけ実行されます。

この命令は常に導通します。MOD 命令が実行できる A、B、C は以下の通りです。

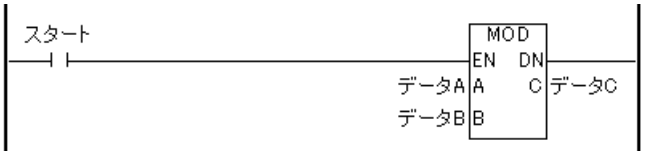
Aのタイプ	Bのタイプ	Cのタイプ
整数定数	整数	整数
整数	整数定数	整数

0 で除算したときは、#Overflow が ON になり、剰余算の結果は不定です。

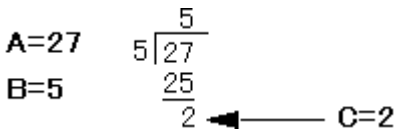
参照→ 8.2.20 #Overflow

◆動作例

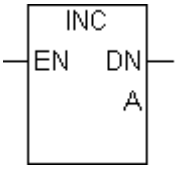
スタートが ON すると、**データ A**が**データ B** で除算され、余りが**データ C** に格納されます。



以下の例では、整数 27 が 5 で除算され、結果 2 が C に格納されます。



9.2.25 INC(インクリメント)



A:データ

INC 命令を実行すると、A に 1 を加えます。この命令は常に導通します。

INC 命令が実行できる A は以下の通りです。

Aのタイプ
整数

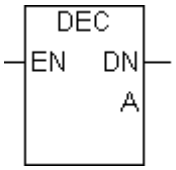
Aが0x7FFFFFFFのとき0x80000000となり、#0verflow がセットされます。

参照→ 8.2.20 #0verflow

◆動作例

スタートがONすると、**データA**に1加算されます。

9.2.26 DEC(デクリメント)



A:データ

DEC命令を実行すると、Aから1を引きます。この命令は常に導通します。

DEC命令が実行できるAは以下の通りです。

Aのタイプ
整数

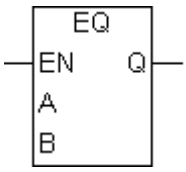
Aが0x80000000のとき0x7FFFFFFFとなり、 #0verflow がセットされます。

参照→ 8.2.20 #0verflow

◆動作例

スタートがONすると、**データA**から1減算されます。

9.2.27 EQ(比較：=)



A:データ
B:データ

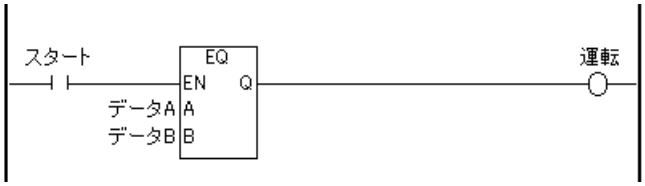
この命令はA = Bのときに導通します。EQ命令が実行できるA、Bはそれぞれ以下の通りです。

Aのタイプ	Bのタイプ
整数	整数
整数定数	整数定数
実数	実数
実数定数	実数定数

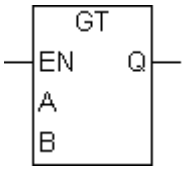
実数値を比較する際、注意が必要です。例えば、演算結果が1.99999999999になることがありますが、これは2.00000000000と等しくありません。

◆動作例

スタートがONで、**データA**と**データB**の数値が等しいときに運転します。



9.2.28 GT(比較：>)



A:データ
B:データ

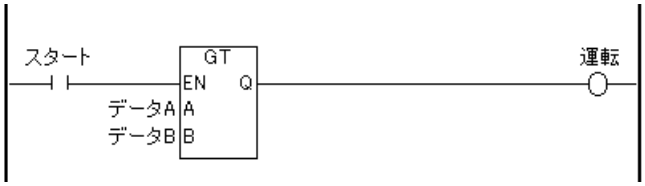
この命令はA > Bのときに導通します。GT命令が実行できるA、Bはそれぞれ以下の通りです。

Aのタイプ	Bのタイプ
整数	整数
整数定数	整数定数
実数	実数
実数定数	実数定数

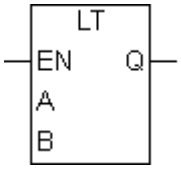
実数値を比較する際、注意が必要です。例えば、計算結果が2.0000000000001になることがありますが、これは2より大きいです。

◆動作例

スタートがONで、**データA**が**データB**の数値より大きいときに運転します。



9.2.29 LT(比較：<)



A:データ
B:データ

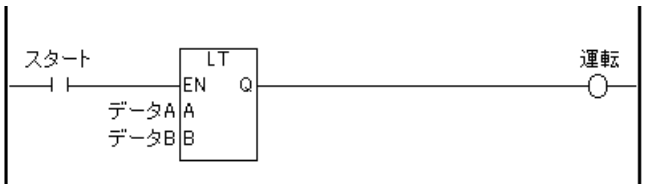
この命令はA < Bのときに導通します。LT命令が実行できるA、Bはそれぞれ以下の通りです。

Aのタイプ	Bのタイプ
整数	整数
整数定数	整数定数
実数	実数
実数定数	実数定数

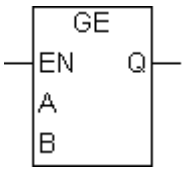
実数値を比較する際、注意が必要です。例えば、計算結果が1.99999999999になることがありますが、これは2より小さいです。

◆動作例

スタートがONで、**データA**が**データB**の数値がより小さいときに運転します。



9.2.30 GE(比較：>=)



A:データ
B:データ

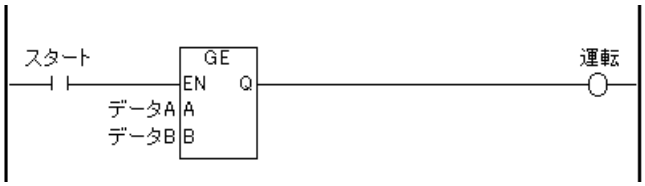
この命令はA ≧ Bのときに導通します。GE命令が実行できるA、Bはそれぞれ以下の通りです。

Aのタイプ	Bのタイプ
整数	整数
整数定数	整数定数
実数	実数
実数定数	実数定数

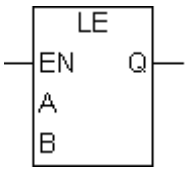
実数値を比較する際、注意が必要です。例えば、計算結果が1.99999999999になることがありますが、これは2以上ではありません。

◆動作例

スタートがONで、**データA**が**データB**の数値以上のときに運転します。



9.2.31 LE(比較：<=)



A:データ
B:データ

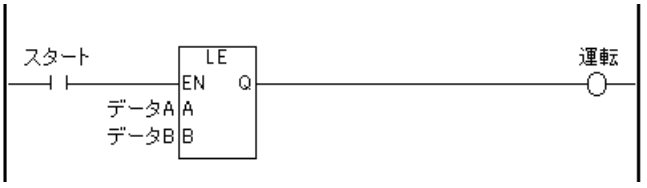
この命令はA ≦ Bのときに導通します。LE命令が実行できるA、Bはそれぞれ以下の通りです。

Aのタイプ	Bのタイプ
整数	整数
整数定数	整数定数
実数	実数
実数定数	実数定数

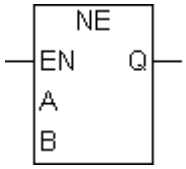
実数値を比較する際、注意が必要です。例えば、計算結果が2.000000000001になることがありますが、これは2以下ではありません。

◆動作例

スタートがONで、**データA**が**データB**の数値以下のときに運転します。



9.2.32 NE(比較：<>)



A:データ
B:データ

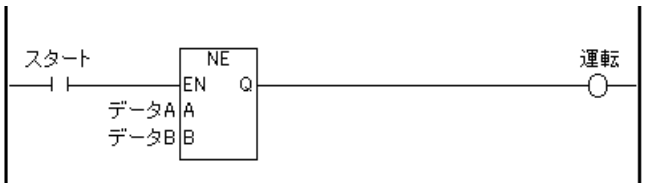
この命令はA ≠ Bのときに導通します。NE命令が実行できるA、Bはそれぞれ以下の通りです。

Aのタイプ	Bのタイプ
整数	整数
整数定数	整数定数
実数	実数
実数定数	実数定数

実数値を比較する際、注意が必要です。例えば、計算結果が1.99999999999になることがありますが、これは2と等しくありません。

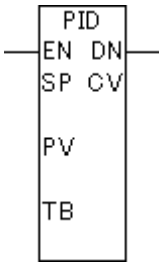
◆動作例

スタートがONで、**データA**と**データB**の数値が等しくないときに運転します。



9.2.33 PID(PID 演算)

コントロールブロック変数



SP : 目標値
PV : 現在値
TB : タイバック
CV : 出力値

PID 命令は、アナログ入力や温度入力からの測定値（現在値）と、あらかじめ設定された値（目標値）を比較して、現在値と目標値の差をなくすように出力値を調節します。

PID 制御を行う場合、P 制御、I 制御、D 制御をそれぞれ自由に組み合わせて制御することができます。後述するそれぞれのパラメータを設定することで、これらの制御を実行します。

PID 制御で算出される出力値は、原理的に次式で表されます。

$$CV=KC\left(E+Reset\int_0^t (E)dt+Rate\frac{d(E)}{dt}\right)$$

KC : 比例係数※1
E : 偏差（SP-PV または PV-SP）
Reset : 積分回数※1
Rate : 微分時間※1

後述する[チューニング]タブでサンプリング時間を調節することによって、偏差に乗るノイズの影響を小さくすることができます。偏差のフィルタ結果は次式で表されます。

$$EF_n=EF_{n-1}+\frac{T_{Loop}}{T_{Filter}}\left(E_n-EF_{n-1}\right)$$

EF : 偏差のフィルタ結果
Tloop : データ取得周期
TFilter: サンプリング時間
E : 偏差（SP-PV または PV-SP）

※1 後述するチューニングタブで設定します。コントロールブロック変数の値ではありません。

■動作概要

PID 命令が導通すると、PID 演算を行い操作量を調節して出力します（自動モード）。下図のように非導通の場合は一定の操作量を出力します（手動モード）。手動モードの出力値はタイバックで設定します。

ロジックプログラムで PID 命令を使用する際、まずコントロールブロック変数と SP、PV、TB、CV に変数を割り付けてください。

◆各パラメータの変数

変数	内容	変数タイプ
SP	目標値	整数、整数定数、整数配列
PV	現在値	整数、整数配列
TB	タイバック 命令が導通していない場合、ここで設定された値が出力されます。	整数、整数定数、整数配列
CV	出力値	整数、整数配列

◆コントロールブロック変数

PID 命令に変数を割り付けると、その変数には下表の要素数 7 の配列が自動的に割り付けられます。要素[0]はステータス、要素[1]～[6]は PID 制御の微調整を行うことができます。

要素番号		詳細
0	ビット0	モード切り替えフラグ
	ビット1	PID命令処理の完了フラグ
	ビット2	PID処理無効範囲フラグ
	ビット3	出力値の上限オーバー
	ビット4	出力値の下限オーバー
	ビット5	積分回数処理オーバー
1		比例係数
2		毎分あたりの積分回数
3		1回あたりの微分時間
4		PID処理無効範囲
5		オフセット
6		サンプリング時間

・ コントロールブロック変数の変数タイプは、保持型変数になります。

重要

・ 比例係数、毎分あたりの積分回数、1 回あたりの微分回数に代入される値は、「チューニング」タブの比例係数、積分回数、微分回数の値を 1000 倍した値になります。

■コントロールブロック変数の要素[0]のステータス

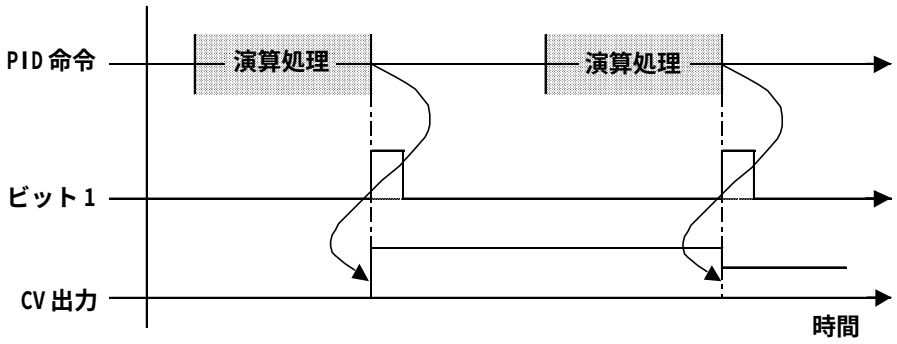
◆モード切り替えフラグ

ロジックプログラムでPID命令が導通している場合にビット0がONになります。

ビット0	モード
ON	自動モード（PID演算）
OFF	手動モード

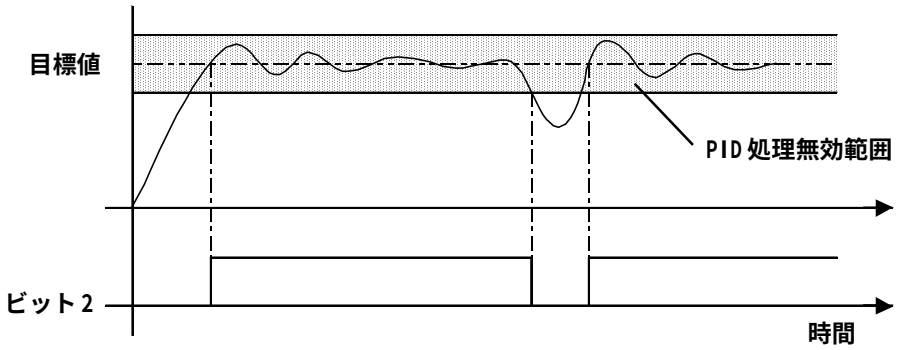
◆PID命令処理の完了フラグ

演算処理が終了してCV値を出力するタイミングでビット1がONになります。ビット1は、1スキャン実行の間ONします。



◆PID処理無効範囲フラグ

[PID]ダイアログボックスの[チューニング]タブ、もしくはコントロールブロック変数の要素[4]で指定した範囲内において、現在値が目標値に達したときにONになり、現在値が範囲外となったときOFFになります。

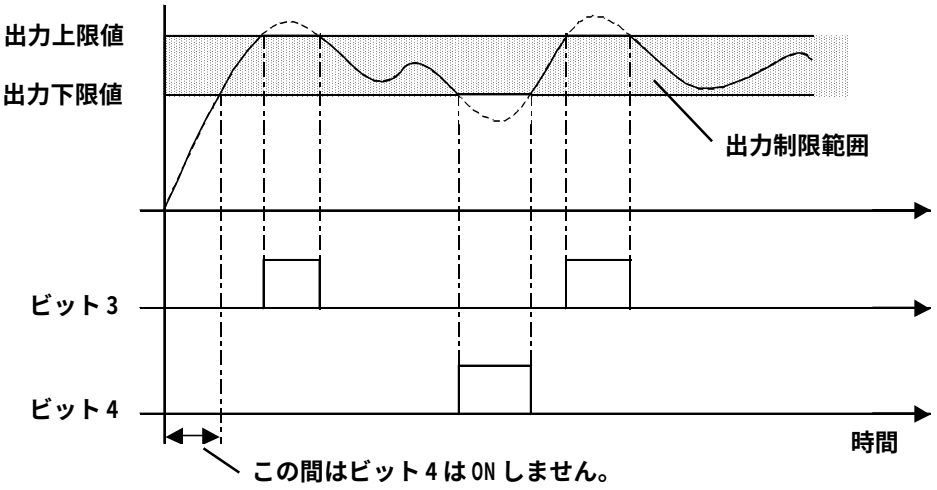


・ [PID]ダイアログボックスの[チューニング]タブについては、後述するPID制御の微調整とモニタリングを参照してください。

◆出力値の上限オーバーと下限オーバー

[PID]ダイアログボックスの[セットアップ]タブで指定した上限に出力値がある場合にビット3がONになり、下限に出力値がある場合にビット4がONになります。

各ステータスのビットがONになってもPID演算は続行され、設定した上限値または下限値で出力されます。



・ [PID]ダイアログボックスの[セットアップ]タブについては、後述するPID制御の微調整とモニタリングを参照してください。

◆積分回数処理オーバー

[PID]ダイアログボックスの[セットアップ]タブで指定した範囲外にて積分回数が処理される場合、ビット5がONになります。ステータスのビットがONになってもPID演算は続行され、設定した上限値で出力されます。

・ [PID]ダイアログボックスの[セットアップ]タブについては、後述するPID制御の微調整とモニタリングを参照してください。

■コントロールブロック変数の要素[1]～[6]

PID制御の微調整を行います。詳しくは、■PID制御の微調整とモニタリングを参照してください。

■ PID制御の微調整とモニタリング

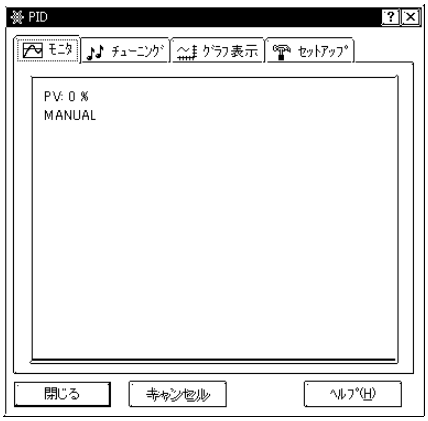
PID命令に、コントロールブロック変数と各パラメータの変数（SP、PVなど）を設定してPID命令をダブルクリックすると、下記の[PID]のダイアログボックスが表示されます。

このダイアログボックスでは、PID制御の微調整やモニタリングを行うことができます。

・各パラメータを自動調整するオートチューニング機能はありません。

◆モニタ

モニタリングモード中にモニタを行うと、そのPID命令の実行結果をモニタすることができます。モニタリングモードの詳細については、**参照→「第3章 モニタリングモードでの動作確認」**



グラフ線種

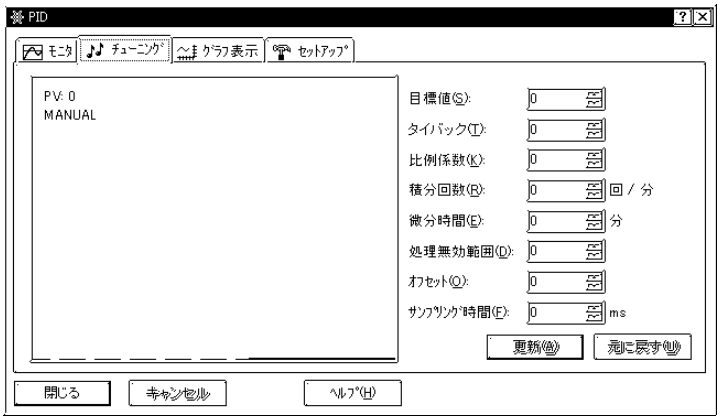
モニタリングしている各項目の線種および線色は下表のようになります。

項目	線種/色	
目標値	黒点線	------
現在値	黒直線	————
出力値	青直線	————

・グラフの線種 / 色の変更はできません。

◆チューニング

モニタリングモード中に各値を調節することができます。ここで設定した値は、各パラメータの変数（SP、TB）やコントロールブロック変数の要素[1]～[6]に反映されます。モニタリングモードの詳細については、**参照→「第3章 モニタリングモードでの動作確認」**

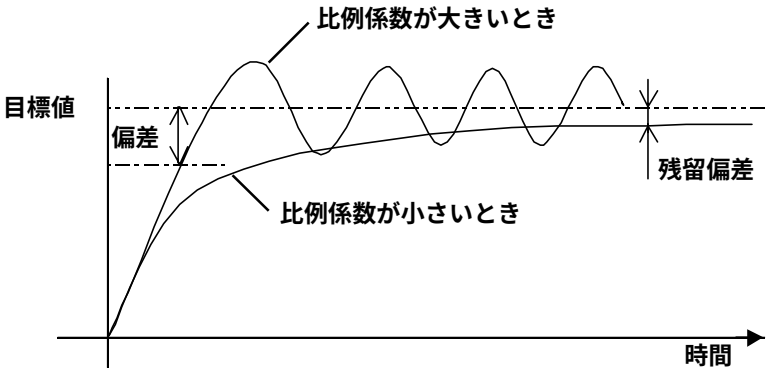


目標値：　目標値を設定します。

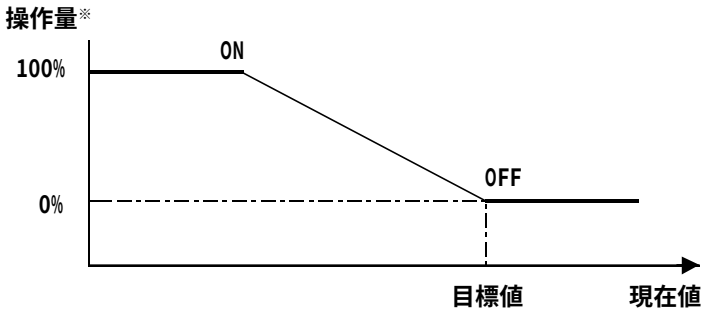
タイバック：　ロジックプログラムでPID命令が導通していない場合、ここで設定された値が出力されます。

比例係数：　目標値と現在値との偏差に比例した操作量が出力されます。

比例係数を小さくすると、目標値に近づけようとする操作量は小さくなりオーバーシュートをなくしますが、残留偏差が大きくなる原因になります。また、比例係数を大きくすると目標値に近づけようとする操作量は大きくなり目標到達時間は短くなりますが、ハンティングする原因になります。



・ 比例制御では、現在値が目標値より小さければ操作量は100%で最大となり、目標値と現在値が一致（偏差なし）すると操作量が0%になります。

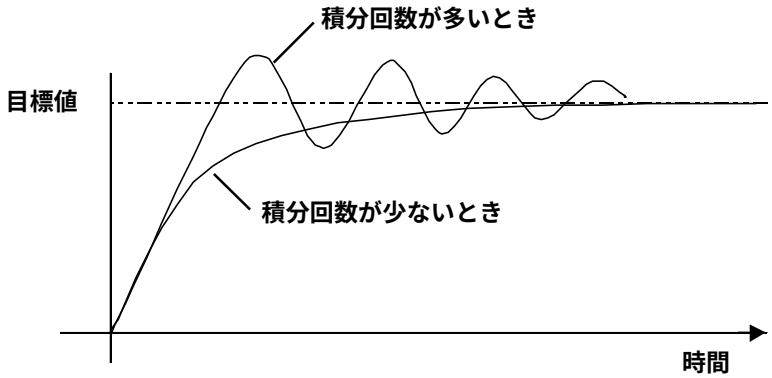


※ 操作量：単位時間あたりの出力量

積分回数：　比例制御だけでは、目標値に近づくと操作量が小さくなりすぎ、操作量（制御出力）が偏差を埋めるだけの値が得られなくなります。そのわずかな誤差を残留偏差といい、積分制御をもちいることによって、この残留偏差を無くすことができます。

積分制御では、偏差を時間的に累積して、ある大きさになれば操作量を増して偏差をなくすように調節する制御方式です。
積分回数は、単位時間あたりに積分を行う回数を設定します。

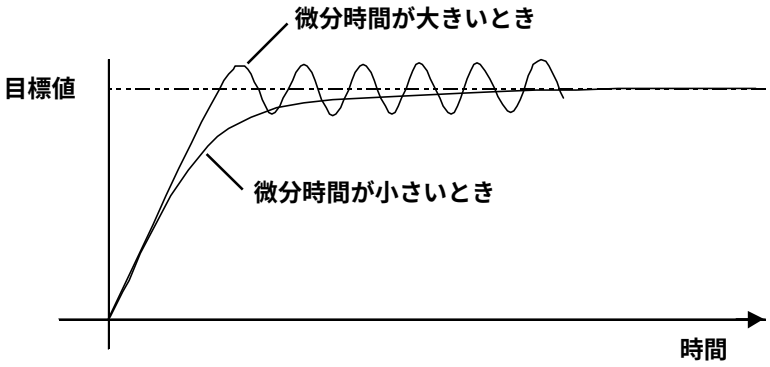
積分回数を多くすると、目標値に近づけようとする操作量は大きくなり目標到達時間は短くなりますが、オーバーシュート、ハンティングの原因となります。
また、積分回数を少なくすると、目標値に近づけようとする操作量は小さくなりオーバーシュート、ハンティングをなくしますが、目標到達時間は長くなります。



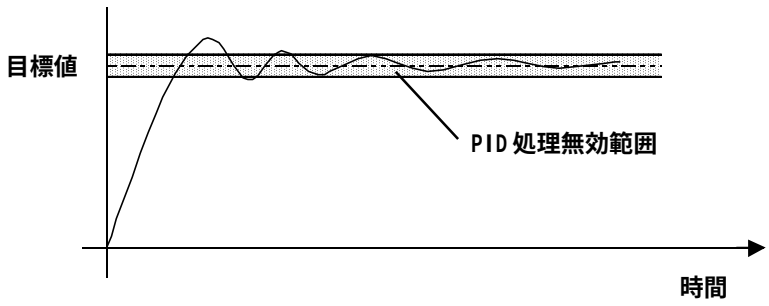
微分時間：　一定の時間（時定数）を必要とする比例制御や積分制御では、外乱に対してすばやく反応できず、すぐには元の目標値には戻せません。

微分制御では、外乱に対して偏差を見て、前回偏差との差が大きいときには大きな操作量を与え、機敏に反応するように働きます。

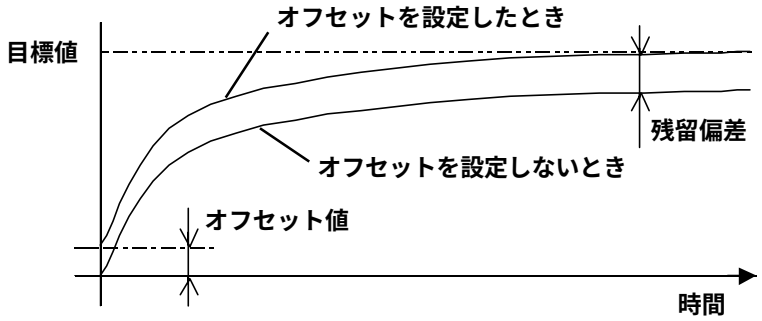
微分時間を大きくすると、外乱に対しての復旧時間は短くなりますが、オーバーシュート、短い周期のハンティングの原因となります。
また、微分時間を小さくすると、オーバーシュート、ハンティングをなくしますが、外乱に対しての復旧時間は長くなります。



処理無効範囲：「処理無効範囲」ではPID制御は行わず、最小出力の値を出力してハンティングのない滑らかな制御を行います。



オフセット：　オフセット値を設定します。比例制御などで生じた残留偏差を少なくすることができます。



サンプリング時間：「データ取得周期」で取得した接続機器の測定データのノイズ除去を目的とします。前回のフィルタ結果と今回の取得データをもとに算出する移動平均です。

サンプリング時間を設定することで、接続機器の測定データの1つが思いもよらない値となっても、前回の測定データとの平均をとって演算するので出力値への影響は小さくなります。

サンプリング時間はデータ取得周期より大きく設定してください。また、サンプリング時間を0に設定するとフィルタは無効になります。

データ取得周期に関しては、後述する「セットアップタブ / コントロール」を参照してください。

チューニングタブからコントロールブロック変数への反映

チューニングタブの各設定項目は、パラメータの変数（SP、TB）とコントロールブロック変数の要素[1]～[6]に反映されます。

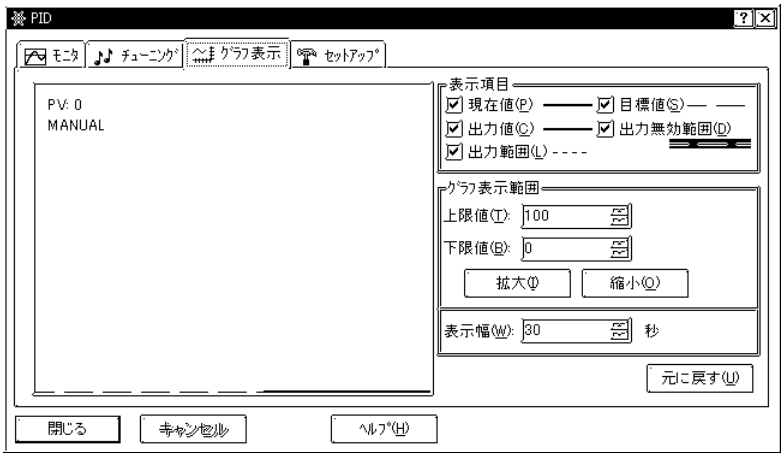
下表は、チューニングタブとパラメータの変数、チューニングタブとコントロールブロック変数の対応表になります。比例係数、積分回数、微分時間は、チューニングタブで設定した1000倍の値がコントロールブロック変数内に書き込まれます。

チューニングタブ	倍率		パラメータの変数	
目標値	×1	⇨	SP	変数
タイバック	×1		TB	変数

チューニングタブ	倍率		コントロールブロック変数	
比例係数	×1000		比例係数	コントロールブロック変数[1]
積分回数	×1000		毎分あたりの積分回数	コントロールブロック変数[2]
微分時間	×1000	⇨	1回あたりの微分回数	コントロールブロック変数[3]
処理無効範囲	×1		PID処理無効範囲	コントロールブロック変数[4]
オフセット	×1		オフセット	コントロールブロック変数[5]
サンプリング時間	×1		サンプリング時間	コントロールブロック変数[6]

◆グラフ表示

現在値(PV)、目標値(SP)、出力値(CV)、出力無効範囲、出力範囲の値をそれぞれモニタリングすることができます。また、モニタリングの設定もできます。



表示項目

モニタリングしている各項目のグラフ線種および線色は下表のようになります。

項目	線種/色	
目標値	黒点線	ーーーーー
現在値	黒直線	――――
出力値	青直線	――――
出力範囲	赤点線	--------
出力無効範囲	グレーゾーン	

・グラフの線種 / 色の変更はできません。

グラフ表示範囲

上限値：グラフ表示範囲の上限値を設定します。

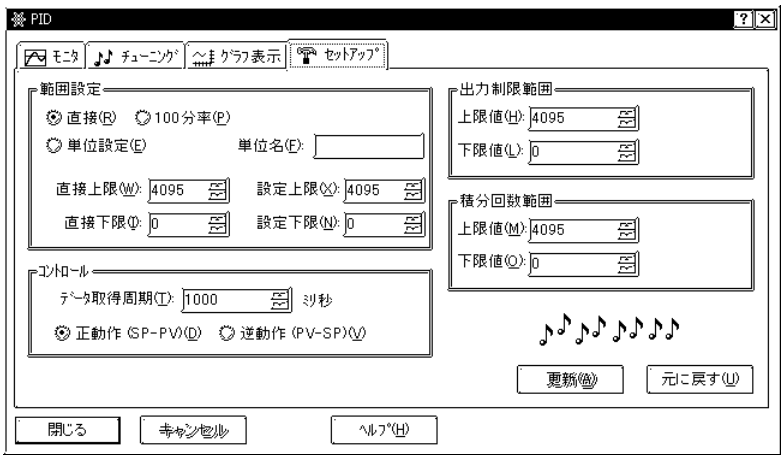
下限値：グラフ表示範囲の下限値を設定します。

表示幅：表示時間の幅を設定します。グラフ表示のサンプリング時間はエディタの[ファイル]メニューから[オプション]を選択し、オプションダイアログボックスの「モニタ」タブの表示周期で変更できます。

・過去のモニタを見ることはできません。

◆セットアップ

プログラミングモード中に、全パラメータに設定できる範囲（上限値と下限値）をあらかじめ設定することができます。



・モニタリングモード中は設定できません。

範囲設定

「直接」、「100 分率」、「単位設定」は、PV の現在値とモニタリングなどの表示部分での数値との変換率を設定します。直接上限と直接下限の値は PV の現在値で、設定上限と設定下限の値はモニタリングなどの表示部分での値を指定します。

直接　　：変換率 0 で接続機器への入出力の値がそのまま表示されます。

この「直接」を選択した場合は、直接上限と直接下限、設定上限と設定下限の値は下記のような設定になります。

直接上限の値＝設定上限の値

直接下限の値＝設定下限の値

100 分率　：表示部分に 100 分率された値が表示されます。

この「100 分率」を選択した場合は、直接上限と直接下限、設定上限と設定下限の値は下記のような設定になります。

直接上限、直接下限の値＝接続機器によるユーザー設定

設定上限の値＝ 100、設定下限の値＝ 0

単位設定　：表示部分でユーザー設定による n 分率された値が表示されます。

この「単位設定」を選択した場合は、直接上限と直接下限、設定上限と設定下限の値は下記のように設定してください。

直接上限、直接下限の値＝接続機器によるユーザー設定

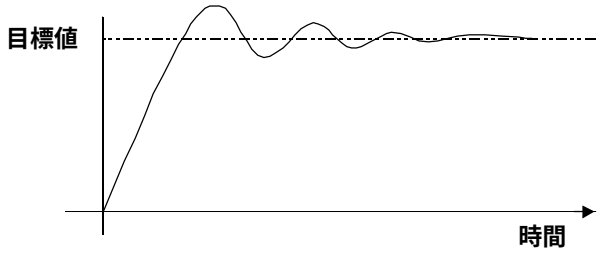
設定上限の値＝ n、設定下限の値＝ 0

コントロール

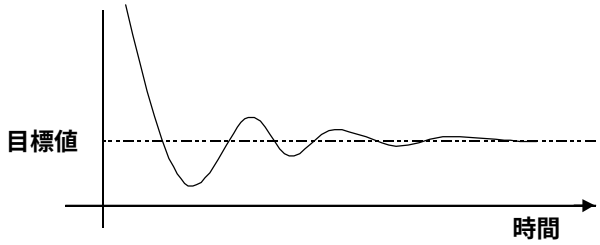
データ取得周期： 接続機器からデータを取得する時間の周期を設定します。データを取得する周期が出力を更新する周期になります。

サンプリング時間を設定することでフィルタ機能を使用することができますが、サンプリング時間はデータ取得周期より大きくなければなりません。

正動作(SP-PV) ： 現在値が目標値より小さいときに操作量を増加させる制御を行う場合に指定します。(暖房など)



逆動作(PV-SP) ： 現在値が目標値より大きいときに操作量を増加させる制御を行う場合に指定します。(冷房など)



出力制限範囲

出力値の上限値と下限値を設定します。ここで設定された値の範囲外の出力であれば、上限値もしくは下限値の値で出力され、コントロールブロック変数の要素[0]のビット3もしくはビット4のステータスビットがONします。詳しくは、◆出力値の上限オーバーと下限オーバーを参照してください。

積分回数範囲

コントロールブロック変数の要素[2]の毎分あたりの積分回数の上限値と下限値を設定します。

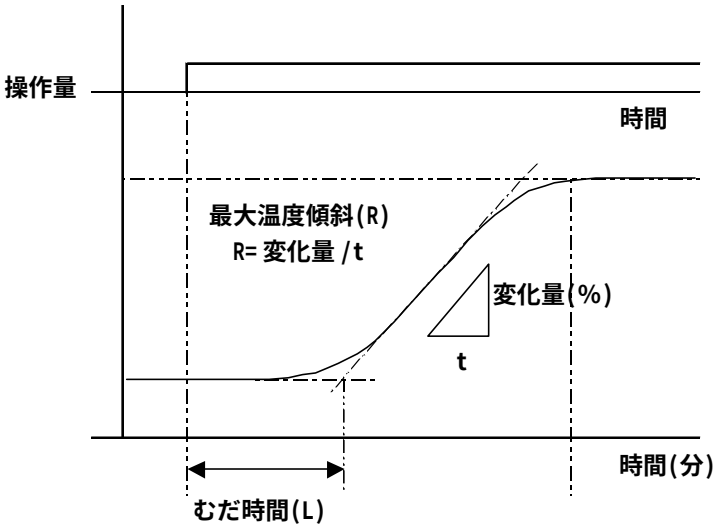
■ PID 定数の調整法

温度制御の場合を例にして説明します。PIDの制御結果を最適なものにするためには、P(比例要素)・I(積分要素)・D(微分要素)の各定数を最適値にする必要があります。さまざまな制御対象に対して、PID定数を温度特性から導き出す方法としてステップ応答法があります。

ただし、制御対象、用途によっては最適値にならない場合がありますので、そのときは[PID]ダイアログボックスの[チューニング]タブで調整してください。

ステップ応答法

目標値を設定して、制御対象に対して操作量100%をステップ状に出力します。このときの下表の温度特性グラフより、最大温度傾斜(R)とむだ時間(L)を計測します。



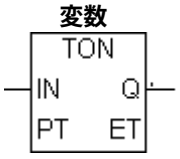
最大温度傾斜(R)とむだ時間(L)を計測した値を下記の方程式に代入して”比例係数”、”積分回数”、”微分時間”の定数を算出します。算出した値を「PID/チューニング」ダイアログボックスに代入してください。

”比例係数” = 100/(0.83・R・L) [%]

”積分回数” = 1/(2・L) [回/min]

”微分時間” = 0.5・L [min]

9.2.34 TON(オンディレータイマ)



IN:タイマ起動用ビット
PT:タイマ設定時間
Q:タイムアップフラグ
ET:タイマ現在値

タイマ入力ビット IN が導通してから、設定した時間 PT(ms 単位)経過後、タイマ出力ビット Q が ON になります。

◆動作概要

専用変数	内容	変数タイプ
変数.PT	設定値	整数
変数.ET	現在値	整数
変数.Q	タイマ出力ビット	ディスクリート
変数.TI	タイマ計測ビット	ディスクリート

タイマ起動用ビット IN が導通すると、TON 命令に起動がかかるために

・経過時間**変数 .ET** が ms 単位で増加します。

・タイマ計測ビット**変数 .TI** が ON になります。

・タイマ出力ビット**変数 .Q** が OFF になります。

経過時間**変数 .ET** が増加して、設定時間**変数 .PT** に等しくなると

・経過時間**変数 .ET** は現在値を保持します。

・タイマ計測ビット**変数 .TI** が OFF になります。

・タイマ出力ビット**変数 .Q** が ON になり、導通します。

TON 命令に起動をかけるタイマ起動用ビット IN の導通をやめると

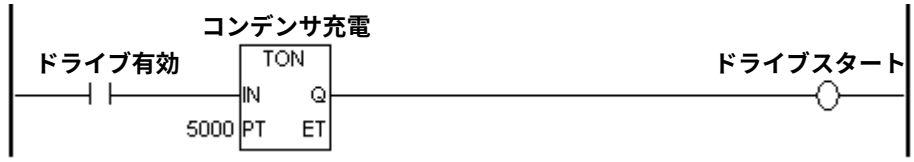
・経過時間**変数 .ET** は 0 にリセットされます。

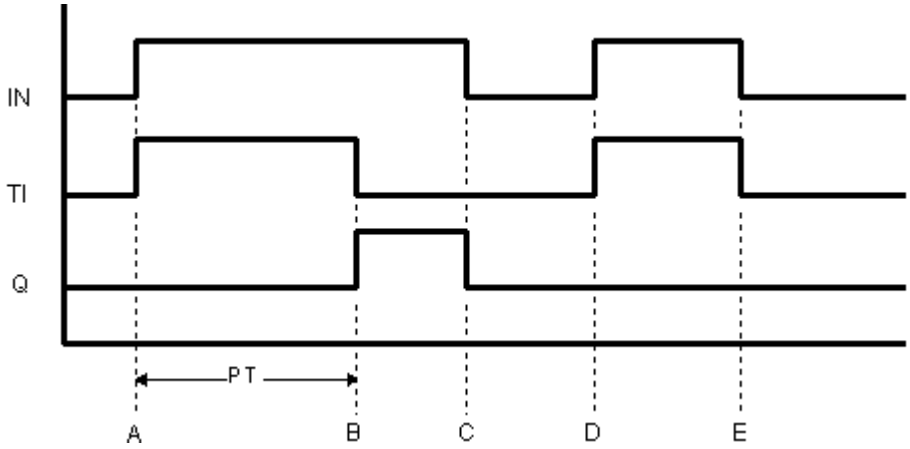
・タイマ計測ビット**変数 .TI** が OFF になります。

・タイマ出力ビット**変数 .Q** が OFF になります。

◆動作例

以下の例では、**ドライブ有効**をONにしてから5秒後にドライブをスタートさせます。





A　タイマ入力ビットINがONになり、タイマ計測ビットTIがONになります。タイマの計測が始まり、経過時間ETが増加します。タイマ出力ビットQはOFFのままです。

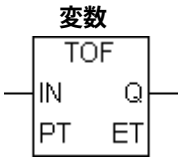
B　経過時間ETが設定時間PTに等しくなると、タイマ出力ビットQがONになります。経過時間ETは、設定時間PTの値のままです。タイマ計測ビットTIはOFFになります。

C　タイマ入力ビットINがOFFになり、タイマ出力ビットQはOFFになります。経過時間ETが0にリセットされます。

D　タイマ入力ビットINがONになり、タイマ計測ビットTIがONになります。タイマの計測が始まり、経過時間ETが増加します。

E　経過時間ETが設定時間PTに達する前にタイマ入力ビットINがOFFになり、タイマ出力ビットはOFFのまま、経過時間ETが0になります。経過時間ETは0にリセットされます。

9.2.35 TOF(オフディレータイマ)



IN:タイマ起動用ビット
PT:タイマ設定時間
Q:タイムアップフラグ
ET:タイマ現在値

タイマ入力ビット IN の導通が止まってから、設定した時間 PT(ms 単位)経過後、タイマ出力ビット Q が OFF になります。

◆動作概要

専用変数	内容	変数タイプ
変数.PT	設定値	整数
変数.ET	現在値	整数
変数.Q	タイマ出力ビット	ディスクリート
変数.TI	タイマ計測ビット	ディスクリート

タイマ起動用ビット IN が導通すると、TOF 命令に起動がかかるために

・経過時間**変数 .ET** が 0 にリセットされます。

・タイマ計測ビット**変数 .TI** が OFF になります。

・タイマ出力ビット**変数 .Q** が ON になり、導通します。

TOF 命令に起動をかけるタイマ起動用ビット IN の導通をやめると

・経過時間**変数 .ET** が ms 単位で増加します。

・タイマ計測ビット**変数 .TI** が ON になります。

・タイマ出力ビット**変数 .Q** は ON のままです。

経過時間**変数 .ET** が増加して、設定時間**変数 .PT** に等しくなると

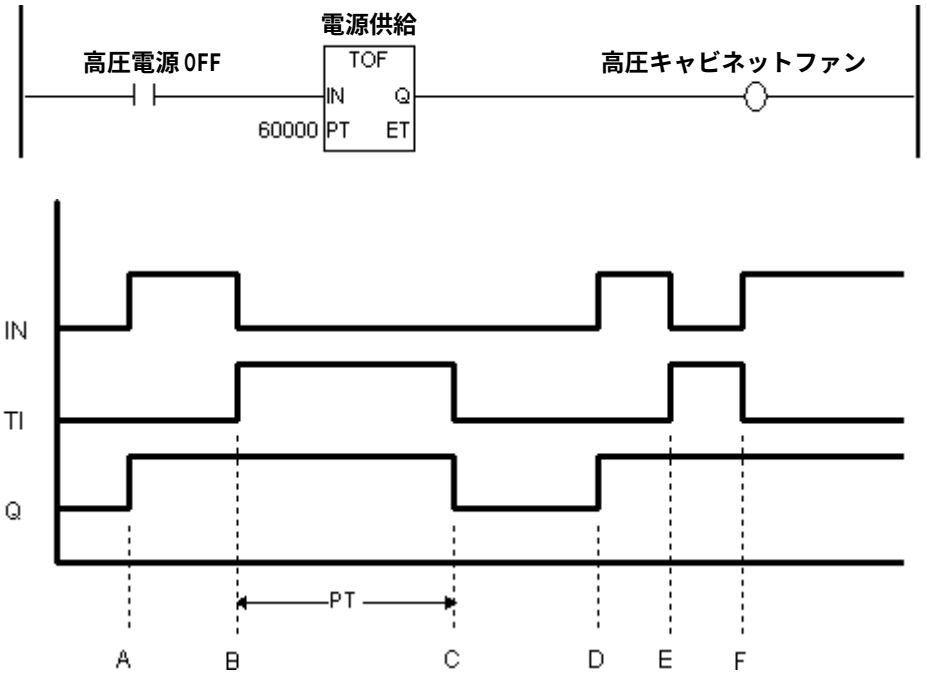
・経過時間**変数 .ET** は、設定値を保持します。

・タイマ計測ビット**変数 .TI** が OFF になります。

・タイマ出力ビット**変数 .Q** が OFF になります。

◆動作例

以下の例では、高圧電源をOFFにしても、ONの間と同じように高圧キャビネットのファンを１分間（60,000ms）動作させています。



A　タイマ入力ビットINがONになります。タイマ計測ビットTIはOFFのままです。タイマ出力ビットQがONになります。経過時間ETが0にリセットされます。

B　タイマ入力ビットINがOFFになります。タイマが計測を開始します（TIはONになります)。タイマ出力ビットはONのままです。

C　まず、経過時間ETが設定時間PTに等しくなります。タイマ出力ビットQがOFFになります。タイマは計測を終了します（TIはOFFになります)。経過時間ETは設定時間と等しいままです（ET=PT)。

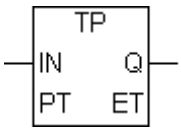
D　タイマ入力ビットINがONになります。タイマ計測ビットTIはOFFのままです。タイマ出力ビットQはONになります。経過時間ETは0にリセットされます。

E　タイマ入力ビットINがOFFになります。タイマが計測を開始します（TIはONになります)。タイマ出力ビットQはONのままです。

F　経過時間ETが設定時間PTに達する前に、タイマ入力ビットINがONになり、タイマは計測を停止します。(TIはOFFになります。)タイマ出力ビットQはONのままで経過時間ETは0にリセットされます。

9.2.36 TP(パルスタイマ)

変数



IN:タイマ起動用ビット
PT:タイマ設定時間
Q:タイムアップフラグ
ET:タイマ現在値

1回のタイマ入力ビットINへの導通で設定時間PT(ms単位)の間タイマ出力ビットQがONになります。

◆動作概要

専用変数	内容	変数タイプ
変数.PT	設定値	整数
変数.ET	現在値	整数
変数.Q	タイマ出力ビット	ディスクリート
変数.TI	タイマ計測ビット	ディスクリート

タイマ起動用ビットINが導通すると、TP命令に起動がかかるために

・経過時間**変数.ET**がms単位で増加します。

・タイマ計測ビット**変数.TI**がONになります。

・タイマ出力ビット**変数.Q**がONになり、導通します。

経過時間**変数.ET**が増加して、設定時間**変数.PT**に等しくなると

・TP命令に導通しているときは、経過時間**変数.ET**は設定値を保持します。

・導通していないときは、すぐに0にリセットされます。

・タイマ計測ビット**変数.TI**がOFFになります。

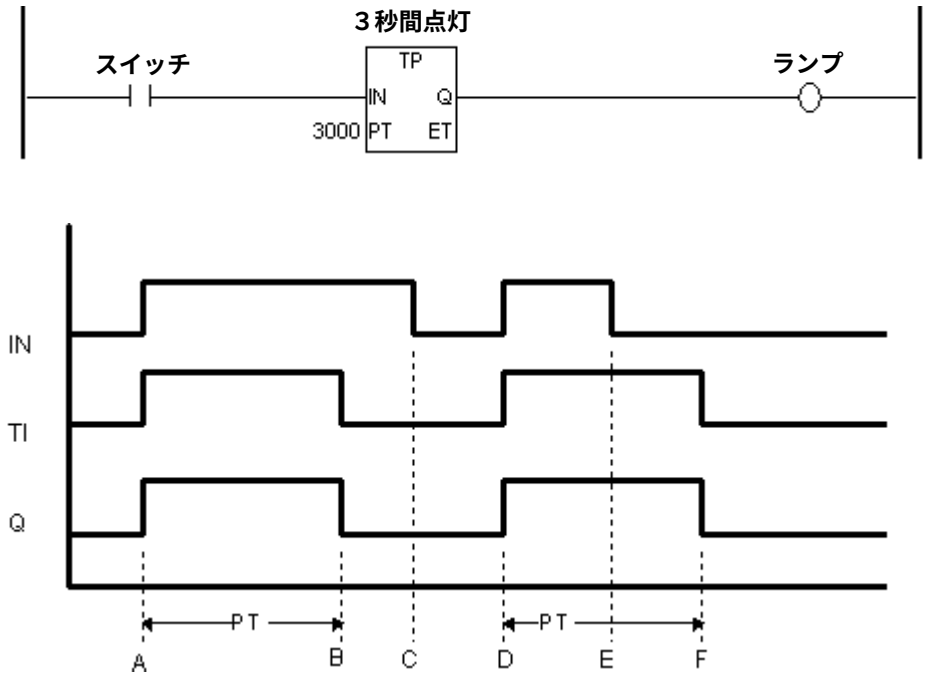
・タイマ出力ビット**変数.Q**がOFFになり、導通を終了します。

TP命令に起動をかけるタイマ起動用ビットINの導通をやめると

経過時間**変数.ET**が設定時間**変数.PT**に達しているときは、経過時間**変数.ET**が0にリセットされ、タイマ出力ビットQはOFFになります。それ以外のときは、計測を続け、タイマ出力ビット**変数.Q**はONのままです。

◆動作例

以下の例では、スイッチを押すと、３秒間だけランプが点灯します。



A　タイマ入力ビット IN が ON になります。タイマが計測を開始します（TI が ON になります)。タイマ出力ビット Q が ON になります。

B　まず、経過時間 ET が設定時間 PT に等しくなります。タイマ出力ビット Q が OFF になります。タイマが計測を終了します（TI が OFF になります)。経過時間 ET は設定時間の値のままです（ET=PT)。

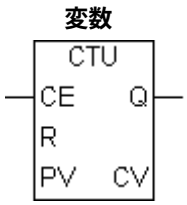
C　タイマ入力ビット IN が OFF になります。経過時間 ET は 0 にリセットされます。

D　タイマ入力ビット IN は ON になります。タイマが計測を開始します（TI が ON になります)。タイマ出力ビット Q が ON になります。

E　タイマ入力ビット IN は OFF になります。タイマは計測を続けます（TI は ON のままです)。タイマ出力ビット Q は ON のままです。

F　まず、経過時間 ET が設定時間 PT に等しくなります。タイマ出力ビット Q が OFF になります。タイマが計測を終了します（TI は OFF になります)。タイマ入力ビット IN が OFF なので経過時間 ET が 0 にリセットされます。

9.2.37 CTU(アップカウンタ)



CE:カウンタ起動用ビット
R:カウンタリセットビット
PV:カウンタ設定値
Q:カウンタ出力
CV:カウンタ現在値

◆動作概要

専用変数	内容	変数タイプ
変数.PV	設定値	整数
変数.CV	現在値	整数
変数.R	カウンタリセット	ディスクリート
変数.UP	アップカウンタ	ディスクリート
変数.QU	アップカウンタ出力	ディスクリート
変数.QD	ダウンカウンタ出力	ディスクリート
変数.Q	カウンタ出力	ディスクリート

カウンタ起動用ビットCEが導通すると、カウンタリセットビット**変数.R**がOFFで、現在値**変数.CV**が設定値**変数.PV**より小さいときに、現在値**変数.CV**が1加算されます。

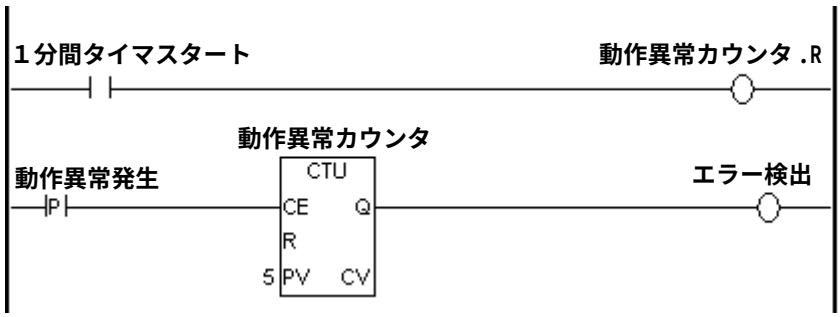
現在値**変数.CV**が設定値**変数.PV**と等しくなると、カウンタ出力ビット**変数.Q**がONになり、導通します。

カウンタリセットビット**変数.R**がONのときは、現在値**変数.CV**が0にリセットされます。

カウンタ出力ビット**変数.Q**もOFFします。

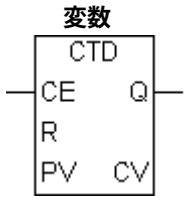
◆動作例

以下の例では、1分間に**動作異常**を5つカウントすると、エラーを知らせます。



カウンタはスキャン毎に更新されます。例のようなイベントをカウントするためには、CTU命令の前にPT命令を挿入してください。CTU命令はレベル入力です。

9.2.38 CTD(ダウンカウンタ)



CE:カウンタ起動用ビット
R:カウンタリセットビット
PV:カウンタ設定値
Q:カウンタ出力
CV:カウンタ現在値

◆動作概要

専用変数	内容	変数タイプ
変数.PV	設定値	整数
変数.CV	現在値	整数
変数.R	カウンタリセット	ディスクリート
変数.UP	アップカウンタ	ディスクリート
変数.QU	アップカウンタ出力	ディスクリート
変数.QD	ダウンカウンタ出力	ディスクリート
変数.Q	カウンタ出力	ディスクリート

カウンタ起動用ビットCEが導通すると、カウンタリセットビット**変数.R**がOFFのときに、現在値**変数.CV**が1減算されます。

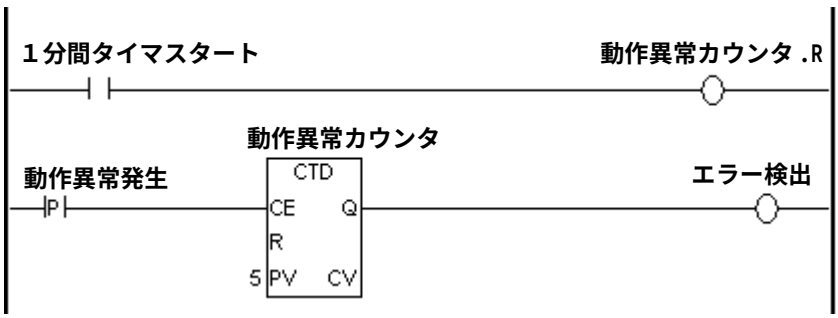
現在値**変数.CV**が1減算され0以下になると、カウンタ出力ビット**変数.Q**はONになり、導通します。

カウンタリセットビット**変数.R**がONのときは、設定値**変数.PV**が現在値**変数.CV**にセットされます。カウンタ出力ビット**変数.Q**もOFFします。

◆動作例

以下の例では、1分間に**動作異常**を5つカウントすると、エラーを知らせます。

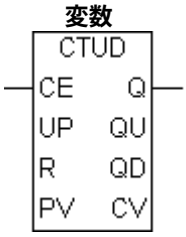
タイマは、カウンタを1分ごとにリセットします。



カウンタはスキャン毎に更新されます。例のようなイベントをカウントするためには、CTD命令の前にPT命令を挿入してください。

9.2.39 CTUD(アップダウンカウンタ)

変数



CE:カウンタ起動用ビット
UP:カウンタアップ指令
R:カウンタリセットビット
PV:カウンタ設定値
Q:カウンタ出力
QU:カウンタアップフラグ
QD:カウンタダウンフラグ
CV:カウンタ現在値

◆動作概要

専用変数	内容	変数タイプ
変数.PV	設定値	整数
変数.CV	現在値	整数
変数.R	カウンタリセット	ディスクリート
変数.UP	アップカウンタ	ディスクリート
変数.QU	アップカウンタ出力	ディスクリート
変数.QD	ダウンカウンタ出力	ディスクリート
変数.Q	カウンタ出力	ディスクリート

CTUD 命令は、カウンタアップ指令**変数 .UP** が ON のときは、CTU 命令(アップカウンタ)を実行するときと同じになります。**変数 .UP** が OFF のときは、CTD 命令(ダウンカウンタ)を実行するときと同じになります。

実行後に現在値**変数 .CV** が設定値**変数 .PV** 以上になったときは、**変数 .Q** と**変数 .QU** が ON になります。

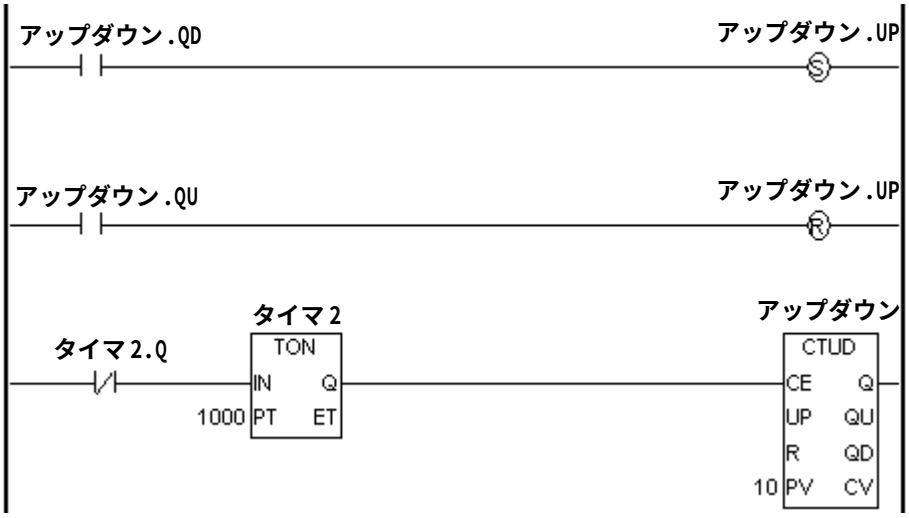
実行後に現在値**変数 .CV** が 0 以下になったときは、**変数 .Q** と**変数 .QD** が ON になります。

◆動作例

以下の例では、CTUD 命令を実行して 0 から 10 まで繰り返しカウントされています。10 までカウントすると、0 に戻ります。

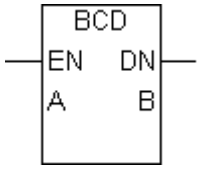
タイマ 2 は、アップダウンカウンタにパルスを毎秒出力します。

カウンタ**アップダウン**が 0 になると UP ビットが ON になり、カウンタ**アップダウン**が 10（プリセット値）になると UP ビットが OFF になります。



カウンタアップ指令**変数.UP**がONのときにカウンタリセットビット**変数.R** がONになると、現在値**変数.CV**が0になります。カウンタアップ指令**変数.UP**がOFFのときにカウンタリセットビット**変数.R**がONになると、現在値**変数.CV**には設定値**変数.PV**が入ります。

9.2.40 BCD(BCD 変換)



A:データ
B:結果格納先

BCD 命令を実行すると、2 進数 の A を 2 進化 10 進数 に変換し、結果を B に格納します。

この命令は、エラーが発生すると導通しません。BCD 命令が実行できる A、B はそれぞれ以下の通りです。

Aのタイプ	Bのタイプ
整数	整数
整数定数	

変換可能な A の最大値は 0x5F5E0FF です。A が大きすぎるとき、#FaultCode はエラーコードで更新され、#Overflow がセットされます。

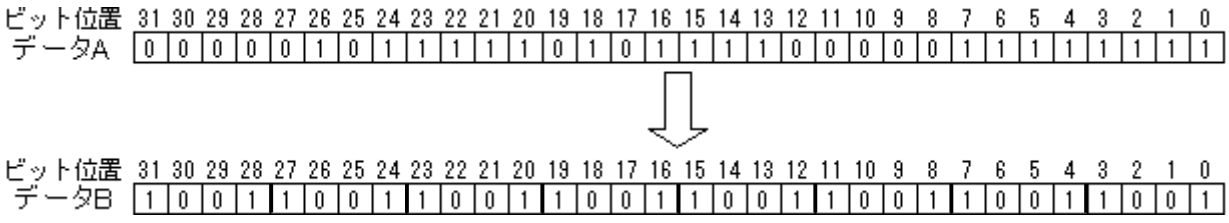
参照→8.2.17 #Faultcode、8.2.20 #Overflow

変換できない値を変換しようとした場合、B の値は不定となります。

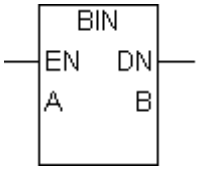
◆動作例

スタートが ON すると、**データ A** を BCD 変換し、**データ B** に格納します。

例)**データ A** に BIN データ ”99999999 ” を設定し、BCD 変換した場合



9.2.41 BIN(バイナリ変換)



A:データ
B:結果格納先

BIN 命令を実行すると、2 進化 10 進数 の A を 2 進数 に変換し、結果を B に格納します。

この命令は、エラーが発生すると導通しません。BIN 命令が実行できる A、B はそれぞれ以下の通りです。

Aのタイプ	Bのタイプ
整数	整数
整数定数	

A が有効な 2 進化 10 進数でないとき、#FaultCode はエラーコードで更新され、#Overflow がセットされます。

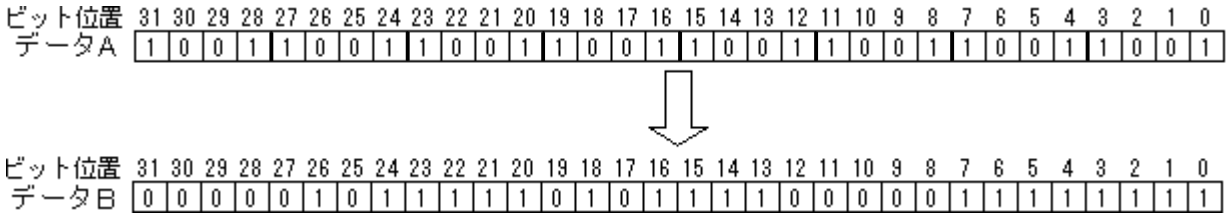
参照→8.2.17 #Faultcode、8.2.20 #Overflow

変換できない値を変換しようとした場合、B の値は不定となります。

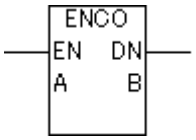
◆動作例

スタートが ON すると、**データ A** を BIN 変換し、**データ B** に格納します。

例)**データ A** に BCD データ ”99999999 ” を設定し、BIN 変換した場合



9.2.42 ENCO(エンコード)



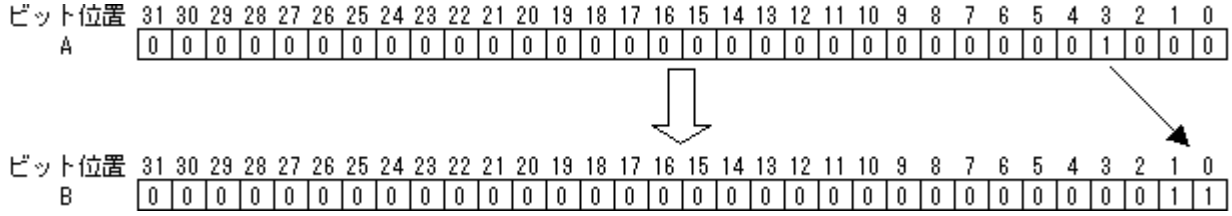
A:データ
B:結果格納先

Aに入力された値をエンコードしてBに出力します。Aの32ビットの内、ONしているビット位置を読みとり、2進数の値としてBに出力します。Aに複数ビットがONしている場所は最上位ビット位置を出力します。

この命令は常に導通します。ENCO命令に有効な変数タイプの組み合わせは以下の通りです。

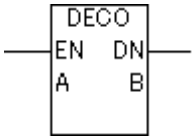
Aのタイプ	Bのタイプ
整数	整数
整数配列	Aと同じサイズの整数配列
整数定数	整数

例：Aに0x00000008を入力した場合、出力Bは0x00000003になります。



・入力Aに0が入っているとマイナー異常#0verFlowをONにし、#FaultCodeにエラーコード ”13”をセットします。**参照→ 8.2.20 #Overflow**
・変数の修飾語は対応しません。(ビット指定、ワード指定、バイト指定)

9.2.43 DECO(デコード)



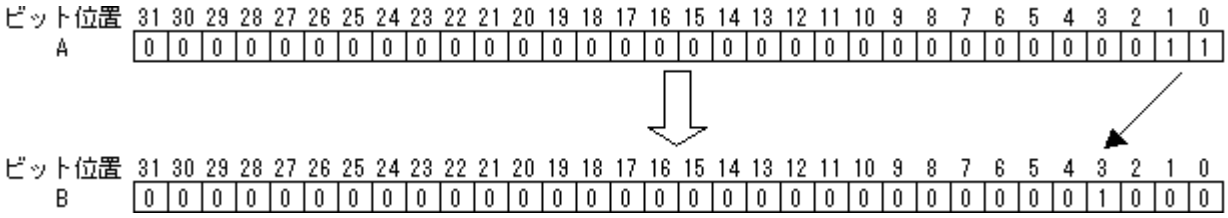
A:データ
B:結果格納先

A に入力された値をデコードして B に出力します。A を 2 進数の値として読みとり、B の対応するビット位置を立てます。0 ～ 31 までの入力のみが有効となります。

この命令は常に導通します。DECO 命令に有効な変数タイプの組み合わせは以下の通りです。

Aのタイプ	Bのタイプ
整数	整数
整数配列	Aと同じサイズの整数配列
整数定数	整数

例：A に 0x00000003 を入力した場合、出力 B は 0x00000008 になります。



・入力 A に 0 ～ 31 以外の値が入力されるとマイナー異常 #OverFlow を ON にし、#FaultCode にエラーコード“ 13 ”をセットします。**参照→ 8.2.20 #Overflow**
・変数の修飾語は対応しません。(ビット指定、ワード指定、バイト指定)

9.2.44 JMP(ジャンプ)

—>> LabelName

JMP 命令に導通すると、指定したラベルへジャンプします。

JSR 命令と違い、ジャンプ元のラングへは自動的に戻りません。

START、SUB START、SUB END を越えてジャンプすることはできません。

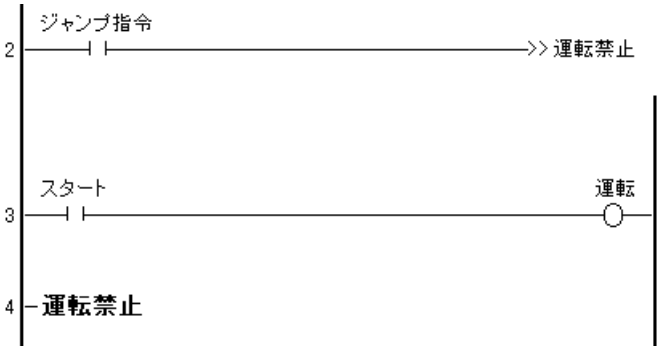
上方向にジャンプすると、無限ループになることがあります。

プログラム全体に要する時間がウォッチドッグタイマの値を超えないようにしてください。

参照→8.2.28 #WatchDogTime

◆動作例

ジャンプ指令が ON していると、ラベル名**「運転禁止」**のラング 4 へジャンプし、ラング 4 以降を実行します。ラング 3 の命令は実行されません。



9.2.45 JSR(ジャンプサブルーチン)

—>> SubroutineName<<

JSR 命令に導通すると、指定したサブルーチンへジャンプします。

サブルーチンが終了すると、ジャンプ元の JSR 命令へ戻り、次の命令の実行が続けられます。サブルーチン名は重複して設定できません。

JSR 命令は、ラングの最後に置いてください。

制限事項

・サブルーチンからのサブルーチンジャンプは、最高 128 回可能です。

プログラム全体に要する時間がウォッチドッグタイマの値を超えないようにしてください。
参照→8.2.28 #WatchDogTime

9.2.46 RET(リターンサブルーチン)

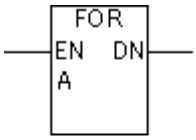
—<RETURN>—

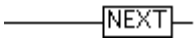
RET 命令に導通すると、サブルーチンから強制的に呼び出し元の JSR 命令に戻り、次のラングから命令の実行が続けられます。

サブルーチンが終了すると、SUB END 命令により自動的に呼び出し元に戻るため、RET 命令は必ずしも使用する必要はありません。

RET 命令は、ラングの最後に置いてください。

9.2.47 FOR、NEXT(繰り返し)





Aで指定した回数、FORとNEXTの間のロジックプログラムを繰り返し実行します。FOR～NEXT命令間の処理を無条件にA回実行すると、NEXT命令の次ステップの処理を行います。

Aが0以下の場合は、FORとNEXTの間のロジックプログラムは実行せず、NEXT命令の次ステップの処理にジャンプします。

この命令は、常に導通します。FOR～NEXT命令が有効な変数タイプは以下の通りです。

Aのタイプ
整数
整数配列
整数定数

制限事項

・必ず、FOR命令に対応したNEXT命令を入れてください。

・同一ラング上で、FOR～NEXT命令の前後に命令を入れないでください。

・ネスティングは最高64回入れることができます。64回を超えるとメジャー異常になり＃FaultCodeにエラーコード4が表示されます。

・1回のネスティングでスタックを「2」使用します。ロジックプログラムで使用できるスタック数は合計で128です。FOR～NEXT命令以外にスタックを使用する命令はJSR命令があり、使用するスタックは「1」です。**参照→9.2.45 JSR命令**

・エディタのエラーチェックで表示されるエラー、警告については、**参照→「第4章 エラーと警告」**

・＃FaultCodeのエラーコードについては、
参照→8.2.17 ＃FaultCode

・ネスティングの回数を指定する時は、プログラム全体に要する時間がウォッチドッグタイマの値を超えないようにしてください。
参照→8.2.28 #WatchdogTime

MEMO

このページは、空白です。
ご自由にお使いください。

第10章
LSエリアリフレッシュ

10.1 LSエリアリフレッシュの概要

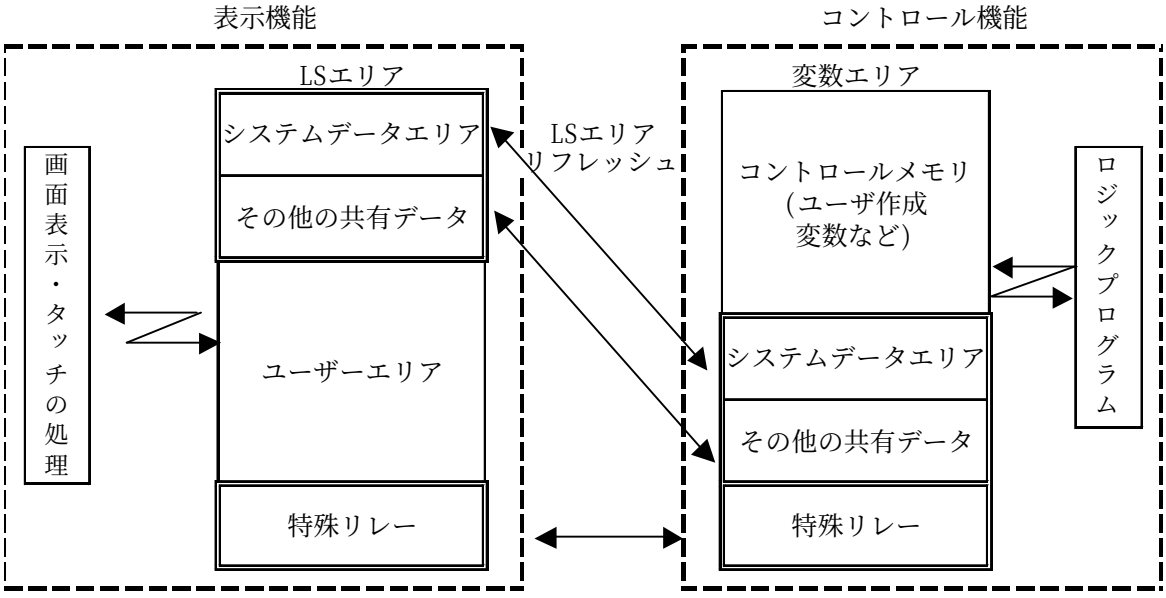
■LSエリアリフレッシュ機能

LTでは、画面切り替えや時計などをLSエリアのシステムデータエリアで管理しています。これらは表示機能として処理されています。

そのためコントロール機能上で画面切り替えや時計などのシステムデータエリア内に割り付けられた機能を使用する場合は、LSエリアを変数として登録し、表示機能とコントロール機能間でLSエリアのデータ共有をする必要があります。

これをLSエリアリフレッシュといいます。

また、システムデータエリア以外に表示機能とコントロール機能で共有させたいデータがある場合にもLSエリアリフレッシュを使用します。



LSエリアリフレッシュのタイミングとロジックシンボルデータ更新タイミングは非同期です。どちらかのトリガでもう一方のデータ更新をロジック上でプログラミングする際はインターロックをかけてください。

10.2 LSエリアリフレッシュの設定

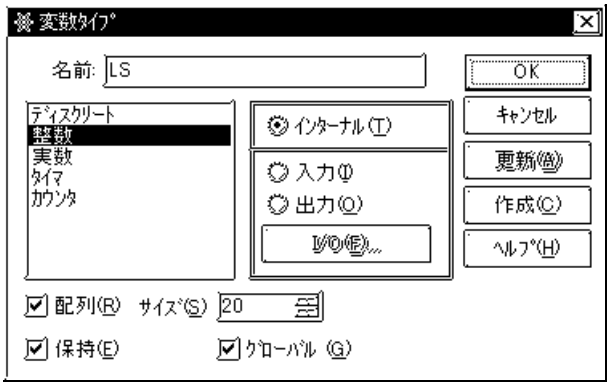
ロジックプログラムにてLSエリアを指定するには、LT Editorにて変数を登録する必要があります。ここではLT Editorにて登録する方法を説明します。

■ 登録方法

LT Editorより、[データ]メニュー→[変数タイプ]を選択すると、[変数タイプ]ダイアログボックスが表示されます。

「LS」という名前(半角で「LS」と入力)の変数をインターナル整数、配列として登録します。

サイズはシステムエリア分で20ワード、その他に共有させたいデータのワード数分を足して算出します。(例: システムデータエリア以外に16ワードのデータを共有したい場合はシステムデータエリア20ワード+16ワード=36ワードを入力)



- **特殊リレーエリアは「LSS」という変数名になります。**
- **「LS」のサイズは最大276ワードです。**

変数とアドレスとの関係は下表に示した通りです。

変数名※1	アドレス	LSアドレス	
LS[0]	0	LS0000	システムデータエリア
LS[1]	1	LS0001	
・ ・	・ ・	・ ・	
LS[19]	19	LS0019	
・ ・	・ ・	・ ・	その他の共有データ
LS[275]	275	LS0275	
LSS[0]	2032	LS2032	特殊リレー
LSS[1]	2033	LS2033	
・ ・	・ ・	・ ・	
LSS[15]	2047	LS2047	

LT Type Cについては機器接続マニュアル(LT Editorに付属)を参照してください。

※1 LTでアクセスする場合の変数名

10.2.1 接続機器を使用しない場合のLSエリア

重要 **・ 以下すべての表にある予約エリアは使用しないでください。**

■ システムデータエリア

システムデータエリアは下記のような構成となっています。

本エリアをコントローラ機能のロジックプログラムでデータ更新することで「LTの画面切り替え」や「バックライトのON/OFF」などを制御することができます。

重要 **・ このエリアは、LT Editorにて「LS」というインターナル整数配列変数を登録することで使用可能になります。**

アドレス	変数名 ※1	内容	機能	ビット	備考
1	LS[1]	ステータス		0、1	予約
				2	プリント中
				3	設定値書き込み
				4～9	予約
				10	バックライト切れ検出
				11～15	予約
2	LS[2]	エラーステータス LTのエラー発生時に、対応するビットがONされます。 一度ONになったビットは、電源をOFFしてから再度ONするか、オフラインモードから再度運転モードに切り替えるまで保持されます。		0、1	未使用
				2	システムROM/RAM
				3	画面記憶メモリチェックサム
				4	SIOフレミング
				5	SIOパリティ
				6	SIOオーバーラン
3	LS[3]			7、8	未使用
				9	内部記憶メモリの初期化が必要
				10	タイマークロック異常
				11～15	未使用
4	LS[4]	時計データ「年」	「年、月、日、時、分」のデータがそれぞれBCD2桁で格納されています。 <例> 1992年2月1日17時15分	0～7	BCD2桁で西暦の下2桁のデータを格納
				8～15	未使用
5	LS[5]	時計データ「月」		0～7	BCD2桁で01～12の月データを格納
				8～15	未使用
6	LS[6]	時計データ「日」		0～7	BCD2桁で01～31の日付データを格納
				8～15	未使用
7	LS[7]	時計データ「時」		0～7	BCD2桁で00～23の時間データを格納
				8～15	未使用
8	LS[8]	時計データ「分」		0～7	BCD2桁で00～59の分データを格納
				8～15	未使用
10	LS[10]	割り込み出力(タッチOFF時)	スイッチ部品でワード書き込みを行うとタッチOFF後、下位8ビットの内容が割り込みコードとして出力されます。(FFhは出力しません)		

※1 LTでアクセスする場合の変数名

アドレス	変数名 ※1	内容	機能	ビット	備考
11	LS[11]	コントロール		0	バックライト
				1	ブザーON
				2	プリント開始
				3	予約
				4	ブザー音　0:出力　1:非出力
				5	予約
				6	タッチパネルを押す事により表示OFFからONへ変更した時の割り込み出力 (割り込みコード:FFh) 0:割り込み出力しない 1:割り込み出力する
				7～10	予約
				11	ハードコピー出力 0：表示、1：非出力キャンセル
				12～15	予約
12	LS[12]	画面表示のON/OFF	FFFFhならば画面表示が消えます。 0hの場合は画面表示します。FFFFh､0h以外の値は予約		
13	LS[13]	割り込み出力	LTのスイッチ部品などのデータを使って絶対書き込みでデータを書くと、下位8ビットの内容が割り込みコードとして出力されます。(FFhは出力しません。)		
15	LS[15]	表示画面番号	画面番号を書き込むと表示画面が切り替わります。	0～14	切り替え画面番号1～8999 (ただしBCD入力の場合は1～1999)
				15	強制画面切り替え 0:通常　1:強制画面切り替え
16	LS[16]	予約			
17	LS[17]				
18	LS[18]				
19	LS[19]				

■ 特殊リレー

特殊リレーは下記のような構成になっています。

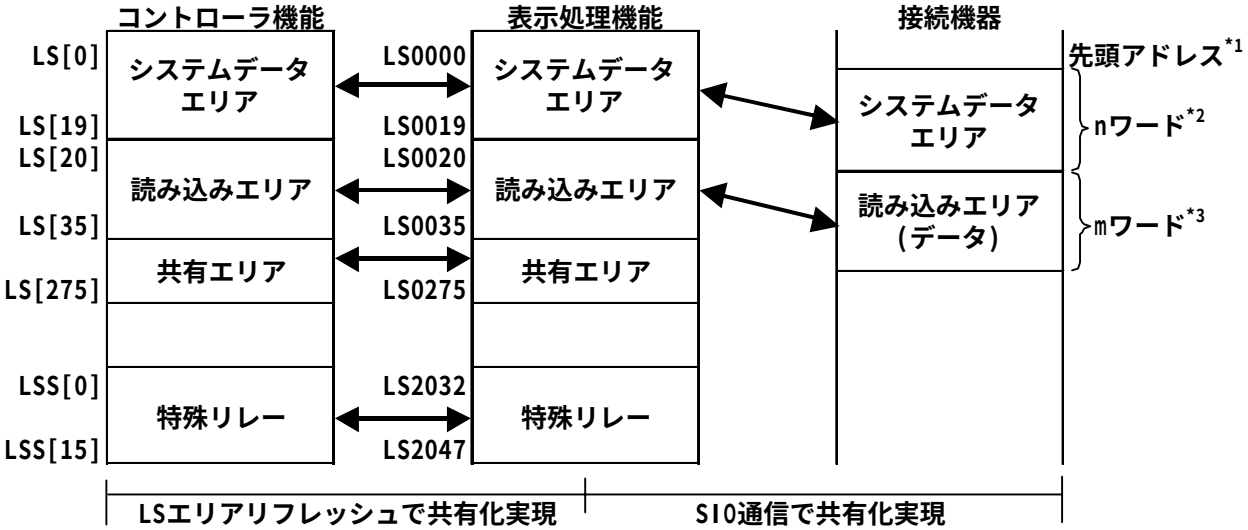
重要 **・ LT Editorにて変数名「LSS」というインターナルの整数配列を登録することで使用可能になります。**

アドレス	変数名※1	概要
2032	LSS[0]	共通リレー情報
2033	LSS[1]	予約
2034	LSS[2]	
2035	LSS[3]	1秒バイナリカウンタ
2036	LSS[4]	部品のスキャンタイム
2037	LSS[5]	予約
2038	LSS[6]	部品のスキャンカウンタ
2039	LSS[7]	予約
2040	LSS[8]	
2041	LSS[9]	
2042	LSS[10]	
2043	LSS[11]	
2044	LSS[12]	
2045	LSS[13]	
2046	LSS[14]	
2047	LSS[15]	

※1　LTでアクセスする場合の変数名

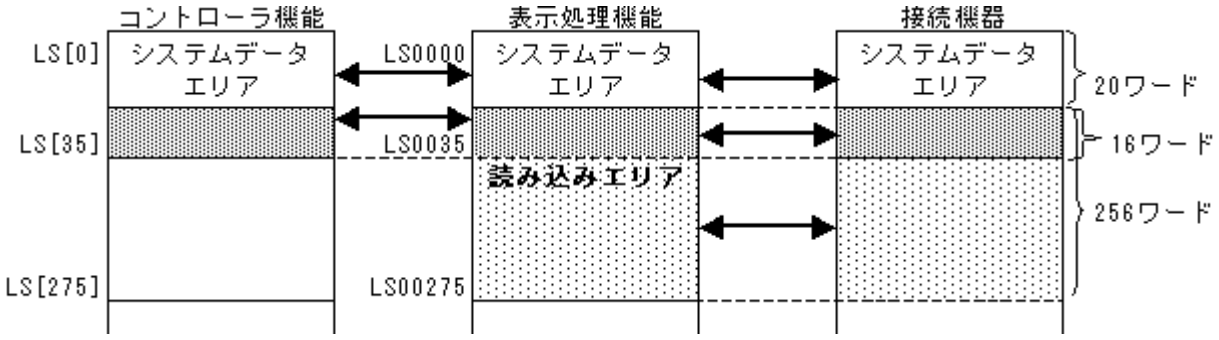
10.3 LT と接続機器のデータ共有について

コントロール機能で接続機器のデータを使用する場合は、LS エリアを経由してデータ共有をおこないます。ただし、コントローラ機能と接続機器のデータレジスタのデータ共有が 16 ワードを越えた場合、画面表示機能が著しく低下する場合があります。

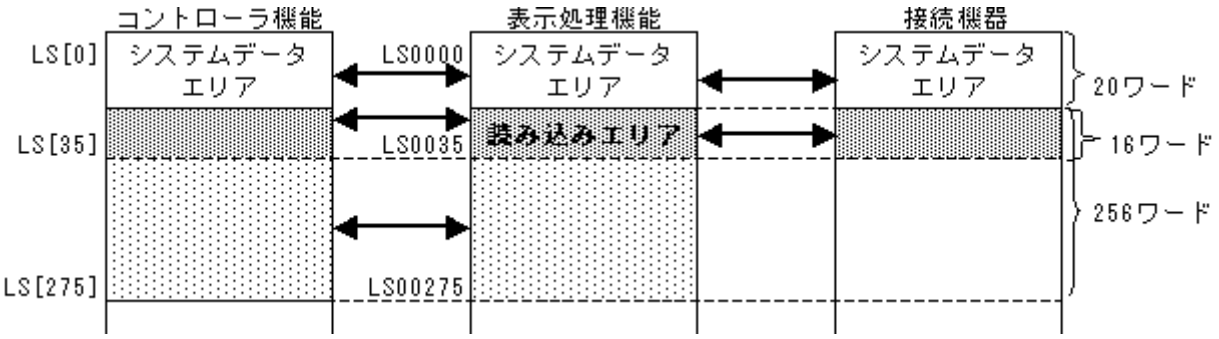


読み込みエリアまたは変数 LS を 16 ワードを超えて設定したい場合、読み込みエリアは 256 ワード、変数 LS のサイズは 276 ワードまで設定可能です。コントローラ機能、表示処理機能、接続機器で共有するデータは 16 ワードまでに設定することを推奨いたします。

例）変数 LS のサイズを 36 ワードに、読み込みエリアを 256 ワードに設定した場合



例）変数 LS のサイズを 276 ワードに、読み込みエリアを 16 ワードに設定した場合



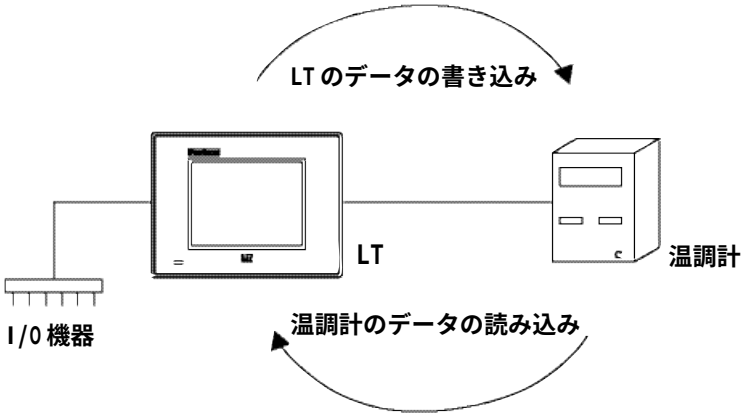
***1 初期設定で指定したシステム先頭アドレスのことです。**

***2 n=0～20　初期設定で指定したシステムデータエリアの選択項目数によって異なります。**

***3 m=0～16　初期設定で指定した読み込みエリアの大きさによって異なります。**

重要
- **コントロールのロジックプログラムとLTの部品、接続機器のロジックプログラムで同一変数にデータ更新した場合、どのデータが優先されるかタイミングにより異なります。**
- **LTにおいて、読み込みエリアにデータ書き込みを行う場合は、部品による書き込みとコントローラ機能のロジックプログラムによる書き込みが競合しないようにご注意ください。**

MEMO
- 読み込みエリアを上手に使ってLTと接続機器のデータ共有を行うと、LTを接続機器の子機として利用したり、FA向けPOPマシンや、生産管理用I/O情報収集端末の構築に有効に使用できます。



10.3.1 LT Type C と接続機器のデータ共有時の注意点

LT Type Cと接続機器のデータ共有は、コントローラ機能によるシステムエリアの制御と接続機器からの読み込みデータをコントローラ機能で参照する場合に活用してください。活用方法としてはコントローラ機能でLS0000～LS0035とLS2032～LS2047を頻繁にデータ更新するようなことは避け、初期セットや運転指示変更のパラメータセットなどプリセットに関するデータの授受に限定して使用することをおすすめします。

上記のLSエリアのデータ更新頻度を上げるとLSエリアリフレッシュが1スキャン内に実行されない恐れがあります。その場合、「接続機器との通信異常」などの異常が発生することがありますので注意してください。

変数LSは整数変数のため、32ビット長になります。

システムデータエリアが16ビット長の場合、下位16ビットが有効になります。

第11章 I/Oドライバ

ここではLTに内蔵されたI/Oを使用する際に必要なI/Oドライバについて説明します。

LT-Type Hのドライバについては「LT Type Hシリーズ I/O設定ユーザーズマニュアル」を参照してください。

11.1 I/Oドライバについて

LT Editorでは、外部入出力を扱う場合、LTに内蔵されたI/Oに対応したI/Oドライバが必要になります。

・ I/Oでエラーが生じた時にコントローラを停止する場合、以下のようなロジックプログラムを作成してください。ただし、異常の検出からロジックプログラム停止まで1スキャンずれることがあります。
下の例では、#IOFaultでI/Oのエラーを検出して#Commandに1を入れてロジックの実行をストップしています。

I/Oにエラーが生じると#IOFaultがONになります。エラーの詳細な情報は#IOStatusで確認することができます。

参照→ 8.2.19 #IOFault、8.2.21 #Command

11.2 Flex Networkドライバ

LTのオフラインモードにあるFlex Networkドライバメニューについて説明します。

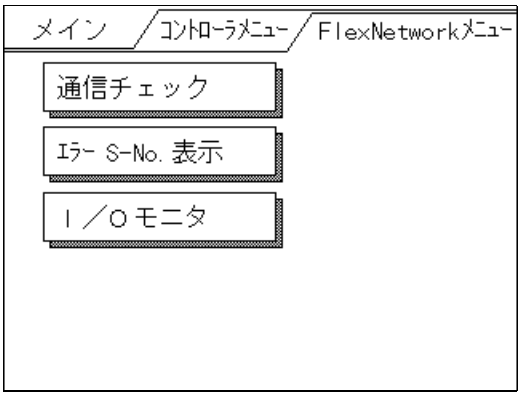
Flex Networkドライバメニューを実行するには、あらかじめLT EditorよりFlex Networkドライバをダウンロードしておいてください。Flex Networkドライバは、LT Type B、Type B+、Type Cで使用します。

オフラインモードに移る方法は、「**LTシリーズユーザーズマニュアル」(別売)**を参照してください。

11.2.1 Flex Networkユニットの自己診断

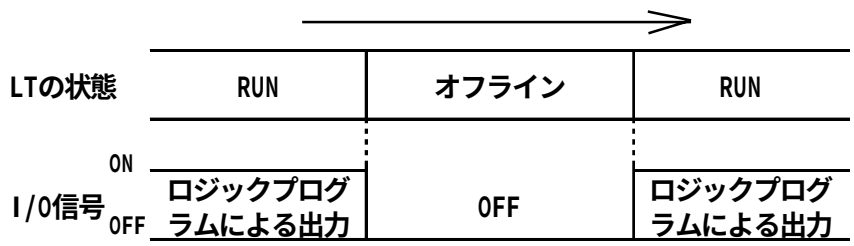
ここではFlex Networkユニットの自己診断の方法について説明します。LT本体の自己診断については、参照→「**LTシリーズユーザーズマニュアル」(別売)**。
コントローラメニューで[Flex Networkドライバ]を選択すると、下の画面が表示されます。

＜通信チェックを選択する場合＞



重要

- **ロジックプログラムのRUN状態から、オフラインモードへの移行またはリセットした場合のLTおよびI/O信号の動作は、I/Oユニット側での出力ホールドの設定にかかわらず、以下の通りです。オフラインモードへの移行やリセットは、これらの動作を十分考慮したうえで行ってください。**



ただし、リセットの場合は、I/O信号がOFFになるタイミングは不定となります。

11.2.2 通信チェック

接続されているFlex Network I/Oユニットの数と各I/Oユニットに設定されているS-No.（局番）をチェックします。

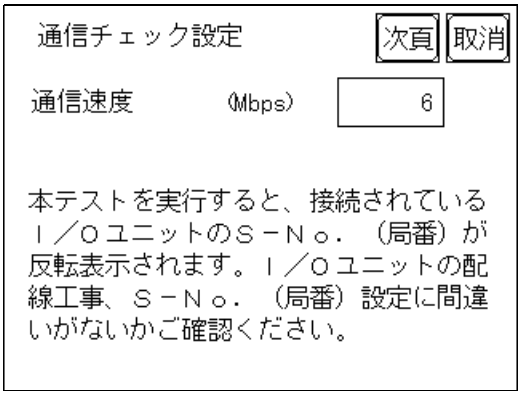
通信チェックにより、I/Oユニットについて以下の確認が行えます。

- 接続されているI/Oユニットの確認
- 故障しているI/Oユニット（通信部）の確認

◆通信チェックの手順

① [通信チェック]を押すと以下の**［通信チェック設定］画面**が表示されます。

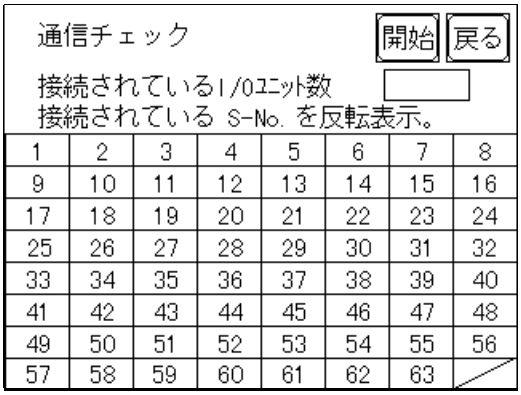
[通信速度]は「6Mbps」、「12Mbps」から選択します。通信速度を速くするとノイズの影響を受けやすくなるので、通常は「6Mbps」で使用してください。



② **[次頁]**ボタンを押すと、以下の**［通信チェック］画面**に切り替わります。

[開始]ボタンを押すと、通信チェックが開始されます。

接続されているI/OユニットのS-No.（局番）が反転表示されます。



③ [戻る]ボタンを押すと、**[Flex Networkメニュー］画面**に戻ります。

11.2.3 エラー S-No.

ロジックプログラムの実行中にエラーコード 841 が発生した場合に、通信回路から外れた I/O ユニットや故障した I/O ユニットの S-No.（局番）をチェックします。

参照→11.2.5 Flex Network 使用時のトラブルシューティング

◆エラー S-No. の手順

[Flex Network ドライバメニュー] 画面で**[エラー S-No. 表示]**を押すと、以下の **[エラー S-No. 表示] 画面**が表示され、エラーチェックが開始されます。

接続されている I/O ユニットの S-No.（局番）が表示され、そのうち異常のある I/O ユニットの S-No. が反転表示されます。

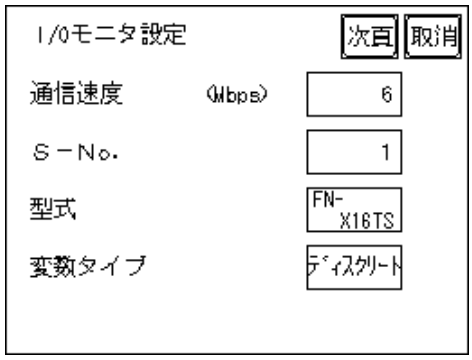
11.2.4 I/Oモニタ（I/O工事接続チェック）

LTとI/Oユニット間の各入出力端子の通信チェックを行います。入力部のチェックは、I/Oユニットの出力信号をLTでモニタリングし、出力部のチェックは、LTからI/Oユニットに出力信号を送り、I/Oユニット側でモニタリングしてチェックしてください。

■I/Oモニタのチェック手順

①［コントローラメニュー］画面で[Flex Networkドライバ]を選択し、［Flex Networkドライバメニュー］画面を表示します。

②［Flex Networkドライバメニュー］画面で[I/Oモニタ]を選択すると、以下の［I/Oモニタ設定］画面が表示されます。



・通信速度

「6Mbps」、「12Mbps」から選択します。通信速度を速くするとノイズの影響を受けやすくなるので、通常は「6Mbps」で使用してください。

・S-No.（局番）

「1-63」から選択します。

・型式

「X16TS」、「Y08RL」、「Y16SK」、「Y16SC」、「XY08TS」、「AD04AH」、「DA04AH」の中から選択します。

FN-X32TS、FN-XY16SK、FN-XY16SC、FN-XY32SK、FN-XY32SCについては該当する機種が選択肢にありません。下表の「I/Oモニタで代用する型式」を選択することでチェックすることができます。

I/Oモニタするユニットの型式			FN-X32TS	FN-XY16SK FN-XY16SC		FN-XY32SKS FN-XY32SCS
I/Oモニタで代用する型式			X16TS	X16TS	Y16SK または Y16SC	XY08TS
S-No.	+0	入力	0-15	0-15	-	0-7
		出力	-	-	0-15	0-7
	+1	入力	16-31	-	-	8-15
		出力	-	-	-	8-15
	+2	入力	-	-	-	16-23
		出力	-	-	-	16-23
	+3	入力	-	-	-	24-31
		出力	-	-	-	24-31

<FN-X32TS の場合 >

「X16TS」を代用します。

S-No. に I/O ユニットに設定した局番を指定すると下位 16 ビット(0 ～ 15 ビット目)がモニタできます。

S-No. に I/O ユニットに設定した局番に 1 つ加えた数値を指定すると上位 16 ビット(16 ～ 31 ビット目)がモニタできます。

<FN-XY16SK、FN-XY16SC の場合 >

入力は「X16TS」を、出力は「Y16SK」または「Y16SC」を代用します。

入力、出力を一度にモニタすることはできません。

<FN-XY32SK、FN-XY32SC の場合 >

「XY08TS」を代用します。

S-No. に I/O ユニットに設定した局番を指定すると 0 ～ 7 ビット目の入出力がモニタできます。

S-NO. に I/O ユニットに設定した局番に 1 を加算した数値を指定すると 8 ～ 15 ビット目の入出力がモニタできます。

S-NO. に I/O ユニットに設定した局番に 2 を加算した数値を指定すると 16 ～ 23 ビット目の入出力がモニタできます。

S-NO. に I/O ユニットに設定した局番に 3 を加算した数値を指定すると 24 ～ 31 ビット目の入出力がモニタできます。

・変数タイプ

「ディスクリート」、「ワード」から選択します。

※「FN-AD04AH」、「FN-DA04AH」は「ワード」のみの設定です。

③**[次頁]**ボタンを押すと、次画面が表示されます。
次画面は、各 I/O ユニットの型式によって異なります。ご使用の I/O ユニットの型式をご確認の上、該当する説明をご参照ください。

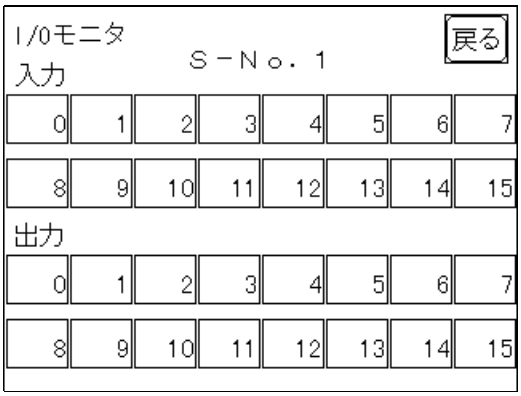

高速カウンタ、1軸位置決めユニットで、本I/Oモニタを使用することはできません。

＜FN-X16TS ／ FN-XY08TS ／ FN-Y08RL ／ FN-Y16SK ／ FN-Y16SC/FN-XY16SK ／ FN-XY16SC ／ FN-X32TS の場合＞

◆ I/O モニタ（[変数タイプ] が「ディスクリート」の場合）

入力部分は入力のあった端子番号が反転表示します。出力部分は端子番号をタッチして反転表示させると出力されます。

[I/O モニタ]画面は選択した**[変数タイプ]**によって異なります。

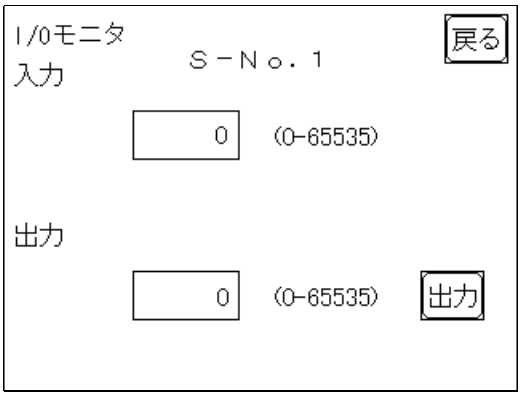


上記画面は、Flex Network システムの 1 つの I/O ユニットの最大入出力点数を表示しています。I/O ユニットの機種により、入力点数、出力点数は異なります。0 を先頭に各 I/O ユニットの持つ点数範囲内で使用してください。

入力専用の I/O ユニットの場合は入力部分のみ、出力専用の I/O ユニットの場合は出力部分のみ、入出力混合の I/O ユニットの場合は入力部分、出力部分の両方を使用してください。

◆ I/O モニタ（[変数タイプ] が「ワード」の場合）

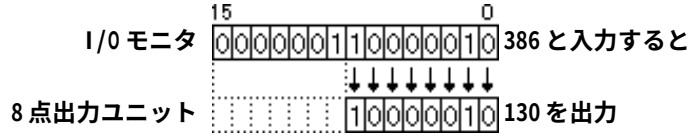
入力部分は入力のあったデータが表示されます。出力部分はテンキーでデータを入力してください。データ表示位置をタッチすると、テンキーパットが表示されます。データ入力後、**[出力]** ボタンを押すとデータが出力されます。データ表示は 10 進数です。



重要 ・ 各 I/O ユニットの I/O 点数に応じて、出力できる範囲のデータを入力してください。

I/O点数	入出力範囲
8点	0～255
16点	0～65535

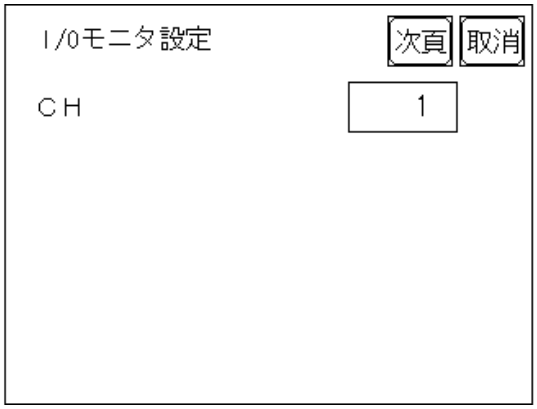
［I/O モニタ設定］画面で選択した「型式」に応じた点数分のデータが I/O ユニットに出力されます。
出力例）
8 点出力の I/O ユニットに 8 ビットで表現できないデータを設定すると、8 ビットを越えるデータは無視されます。



＜ FN-AD04AH ／ FN-DA04AH の場合＞

◆ I/O モニタ設定（チャンネル設定）

チャンネル部分をタッチすると、選択可能な設定内容が順次切り替わります。

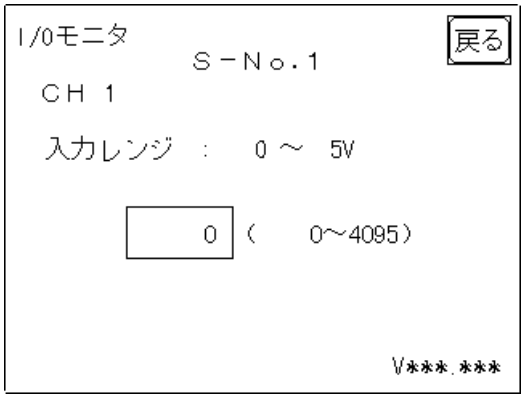


[次頁]ボタンを押すと、次の**[I/O モニタ]画面**に切り替わります。FN-AD04AH と FN-DA04AH では画面が異なります。

＜FN-AD04AH の場合＞

◆I/O モニタ

入力データを表示します。



[戻る]ボタンを押すと、**[I/O モニタ設定]画面**に戻ります。

・A/D 変換表

入力レンジ設定	入力範囲
0～5V	0～4095
1～5V	0～4095
0～10V	0～4095
-5～5V	-2048～2047
-10～10V	-2048～2047
0～20mA	0～4095
4～20mA	0～4095

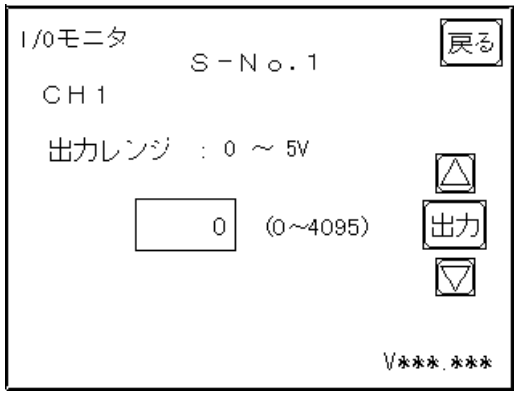
重要

- **フィルタタイプ、A/D 変換サンプル回数、最大／最小除外設定は、I/O ユニット側に保存されている設定内容で動作します。I/O ユニット側に保存されている設定内容を変更するには、LT Editor から設定内容を変更し、LT にロジックプログラムをダウンロードします。その後、ロジックプログラムを RUN モードにして有効になります。**
- **レンジ切り替えスイッチの設定内容は I/O ユニットの電源投入時のみユニット内部に読み込まれます。レンジ切り替えスイッチの設定を変更する時は、必ず I/O ユニットの電源を一度切ってから再投入してください。**
- **I/O ユニット側のレンジ切り替えスイッチの設定内容はロジックプログラムを RUN モードに移行する時に読み込まれます。レンジ切り替えスイッチの設定を変更する時は、ロジックプログラムを一度 STOP モードにしてから RUN モードにしてください。レンジが一致していないと、正常にデータが読み込めません。**

＜FN-DA04AH の場合＞

◆ I/O モニタ

テンキーでデータを入力してください。データ表示位置をタッチすると、テンキーパットが表示されます。データ入力後、[出力]ボタンを押すとデータが出力されます。データ表示は10進数です。



重要 ・上矢印、下矢印を押すと、加算／減算された後に I/O ユニットに出力を行います。
・[戻る]ボタンを押すと、I/O ユニット側で出力ホールド設定にしていても、出力がクリアされます。

・D/A 変換表

入力レンジ設定	入力範囲
0～5V	0～4095
1～5V	0～4095
0～10V	0～4095
-5～5V	-2048～2047
-10～10V	-2048～2047
0～20mA	0～4095
4～20mA	0～4095

重要 ・ レンジ切り替えスイッチの設定内容は I/O ユニットの電源投入時のみユニット内部に読み込まれます。レンジ切り替えスイッチの設定を変更する時は、必ず I/O ユニットの電源を一度切ってから再投入してください。
・ I/O ユニット側のレンジ切り替えスイッチの設定内容はロジックプログラムを RUN モードに移行する時に読み込まれます。レンジ切り替えスイッチの設定を変更する時は、ロジックプログラムを一度 STOP モードにしてから RUN モードにしてください。レンジが一致していないと、正常にデータが書き込めません。

11.2.5 Flex Network使用時のトラブルシューティング

ここでは、Flex Network使用時の異常とその対処方法を示します。参考にしてください。

■Flex Networkの入力/出力異常

Flex Network使用時の入力/出力異常につきましては、「Flex Networkユーザーズマニュアル（別売）」をご覧ください。

■エラーコード

I/Oエラーは、I/Oの読み込み/書き込みのエラーです。I/Oエラーが発生すると、コントローラは#IOStatusにエラーコードを書き込みます。発生するエラーの内容と対処方法を説明します。

◆設定エラー

エラーコード	内容	対処方法
501	I/Oターミナルに割り当てられる内部変数エラー	割り当てられている変数タイプを設定し直してください。
502	出力ターミナルに割り当てられる入力変数エラー	
503	入力ターミナルに割り当てられる出力変数エラー	
504	整数ターミナルに割り当てられるディスクリート変数エラー	
505	ディスクリートターミナルに割り当てられる整数変数エラー	
506	ドライバでサポートされていない変数タイプです	変数タイプを見直してください。
507	ターミナルに変数が割り当てられていません	すべてのターミナルに変数を割り当ててください。
801	ターミナル番号が重複しています	LTEファイルが破損しているか、LTEファイルのダウンロード中に障害が発生した可能性があります。
802	S-No.が重複しています	2つ以上のI/OユニットがS-No.を使用しています。S-No.が重複しないように設定し直してください。
803	S-No.が範囲を超えています	LTEファイルが破損しているか、LTEファイルのダウンロード中に障害が発生した可能性があります。
804	アナログユニットでS-No.が範囲を重複しています	2つ以上のI/OユニットがS-No.を使用しています。アナログユニットはS-No.を4局占有します。S-No.が重複しないように設定し直してください。
805	高速カウンタユニットでS-No.が範囲を重複しています	2つ以上のI/OユニットがS-No.を使用しています。高速カウンタユニットはS-No.を8局占有します。S-No.が重複しないように設定し直してください。
806	1軸位置決めユニットでS-No.が範囲を重複しています	2つ以上のI/OユニットがS-No.を使用しています。位置決めユニットはS-No.を4局占有します。S-No.が重複しないように設定し直してください。

◆初期化エラー

エラーコード	内容	対処方法
821	Flex Network I/Fユニットがありません	内蔵Flex Network I/Fユニットから読み出したID番号が正しくありません。内蔵Flex Network I/Fユニットの故障が考えられます。エラーコードを記録して、(株)デジタル サポートダイヤルまでお問い合わせください。
822	イニシャル異常 イニシャル処理でFlex Networkドライバとの Flex Network I/Fユニットの同期が取れていません	内蔵Flex Network I/Fユニットの故障が考えられます。エラーコードを記録して、(株)デジタル サポートダイヤルまでお問い合わせください。
823	アナログユニット設定異常	通信線に断線がないか、またはI/0ユニットに電源が入っていないか、またはI/0ユニットが故障している恐れがありますので確認してください。

◆ランタイムエラー

エラーコード	内容	対処方法
841	接続されているI/0ユニットに異常（断線、故障）があります	断線していないか確認してください。**参照→Flex Networkユーザーズマニュアル（別売）**
842	アナログユニット（A/D変換ユニット）へ入力するセンサの出力信号線の断線	出力信号線に断線が考えられます。センサの出力信号線をチェックしてください。
843	高速カウンタユニットに異常があります	高速カウンタユニットがエラーを検知しました。**参照→Flex Network高速カウンタユニットユーザーズマニュアル（別売）**
844	高速カウンタユニットのイニシャル異常	通信線に断線がないか、またはI/0ユニットに電源が入っていないか、またはI/0ユニットが故障している恐れがありますので確認してください。
845	高速カウンタユニットとの通信異常	通信線に断線がないか、またはI/0ユニットに電源が入っていないか、またはI/0ユニットが故障している恐れがありますので確認してください。
846	1軸位置決めユニットに異常があります	位置決めユニットがエラーを検知しました。**参照→Flex Network1軸位置決めユニットユーザーズマニュアル（別売）**
847	1軸位置決めユニットとの通信異常	通信線に断線がないか、またはI/0ユニットに電源が入っていないか、またはI/0ユニットが故障している恐れがありますので確認してください。

◆内部エラー

エラーコード	内容	対処方法
850 ⋮ 859	ドライバエラー システム内に重大なエラーが発生しました	LTをリセットしてください。その後もエラーコードが表示される場合は、周辺環境によりエラーが誘発されているか、LT本体の異常が考えられます。エラーコードを記録して、サポートダイヤル（P.10記載）までお問い合わせください。

11.3 DIOドライバ

LTのオフラインモードにあるDIOメニューについて説明します。DIOメニューを実行するには、あらかじめLT EditorよりDIOドライバをダウンロードしておく必要があります。DIOドライバはLT Type Aで使用します。

オフラインモードに移る方法は、**「LTシリーズユーザーズマニュアル」（別売）を参照してください。**

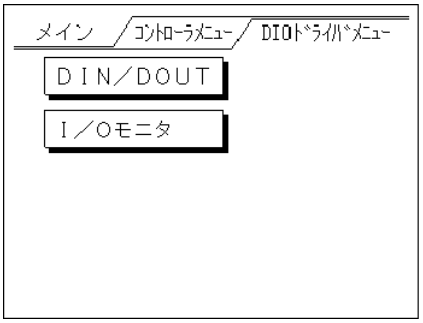
11.3.1 DIOの自己診断

ここではDIOの自己診断の方法について説明します。

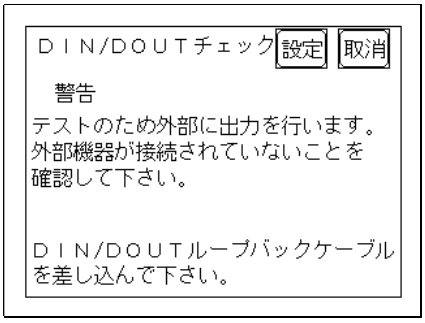
LT本体の自己診断については、
参照→「LTシリーズユーザーズマニュアル」（別売）

コントローラメニューで[DIOドライバ]を選択すると、下の画面が表示されます。

＜DIOドライバを選択する場合＞



[DIN/DOUT]を押すと下の画面が表示されます。

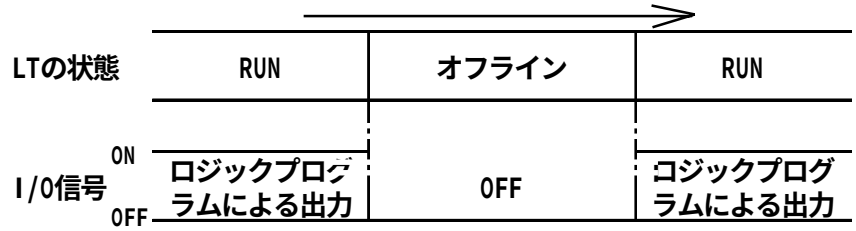


[設定]/[開始]を押すとチェックが開始されます。

このチェックは、出力ユニットから出た信号を入力ユニットで取り込みます。
チェックを行うにはDIN/DOUTループバックケーブルを接続してください。

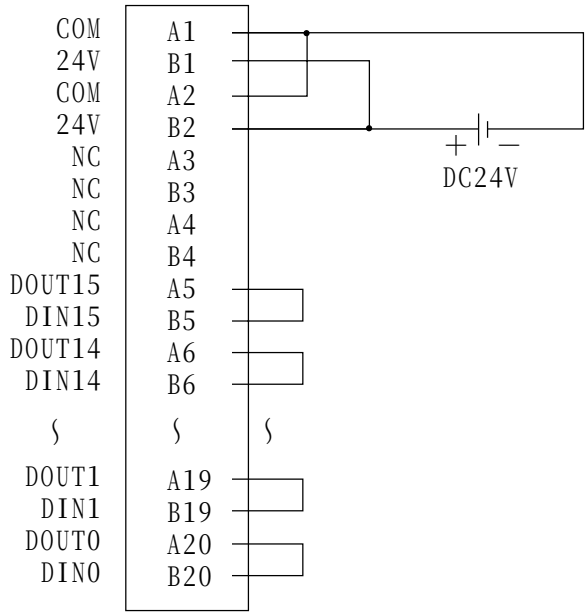
重要

・ ロジックプログラムのRUN状態からオフラインモードへの移行、またはリセットした場合、I/Oの信号はOFFされることがあります。I/Oの信号がOFFされることを十分考慮して行ってください。



ただし、リセットの場合は、I/O信号がOFFになるタイミングは不定となります。

DIN/DOUTループバックケーブルの配線は、以下のとおりです。(シンクタイプの場合)



＜推奨品＞

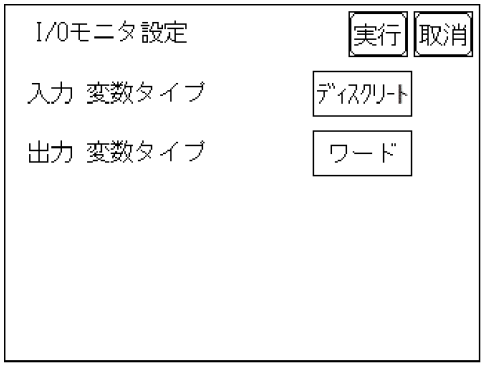
接続方法	メーカー	型番
ハンダ付けタイプ	富士通(株)	FCN-361J040-AU (コネクタ) FCN-360C040-B (カバー)
	(株)デジタル	FN-IFCN01*1 (コネクタ／カバー)
圧着タイプ	富士通(株)	FCN-363J040 FCN-363J-AU/S FCN-360C0404-B
端子台ユニットタイプ	三菱電機(株)	A6TBX36 (端子台ユニット) AC＊＊TB (ケーブル) (＊＊は、ケーブル長を表します。)
	横河電気(株)	TA40-ON

***1 製品自体は富士通（株）製と同じです。**

11.3.2 I/Oモニタ（I/O工事接続チェック）

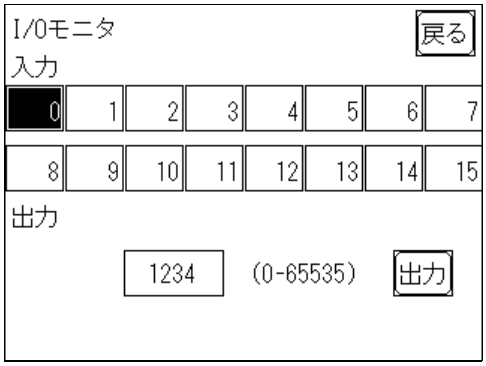
DIOドライバメニューで[I/Oモニタ]を選択すると、下の画面が表示されます。

＜I/Oモニタを選択する場合＞



[入力変数タイプ]は「ディスクリート」、「ワード」より選択します。
[出力変数タイプ]は「ディスクリート」、「ワード」より選択します。

例えば、[I/Oモニタ設定]画面で「入力変数タイプ:ディスクリート」、「出力変数タイプ:ワード」を選択する場合、画面右上の実行ボタンをタッチすると確定し、[I/Oモニタ]画面が表示されます。



入力変数タイプ[ディスクリート]の場合、入力のあった端子番号が反転表示します。
出力変数タイプ[ワード]の場合テンキーでデータを入力してください。LTの場合はデータ表示位置をタッチすると、テンキーパットが表示されます。
データを入力後、「出力ボタン」を押すとデータが出力されます。データ表示は10進数です。

11.3.3 DIO使用時のトラブルシューティング

ここでは、DIO使用時の異常とその対処方法を示します。参考にしてください。

■DIOの入力異常

異常現象	推定原因	対処方法
入力モニタランプは点灯するが、まったく入力できない。	DIOの不良	サポートダイヤル(本マニュアルP.10に記載)までお問い合わせください。
	I/O使用可が未設定	I/O使用可の設定をする
	プログラムの不良	プログラムの修正
入力モニタランプが消灯し、まったく入力できない。	DIOの不良	サポートダイヤル(本マニュアルP.10に記載)までお問い合わせください。
	入力コモン線の配線ミス	コモン線の配線チェック コモン線の断線チェック コモン端子の緩みチェック
	外部入力電圧不良	定格電圧を供給する
	コネクタの接触不良	コネクタを確実にネジで取り付ける
入力すべてがOFFしない。	DIOの不良	サポートダイヤル(本マニュアルP.10に記載)までお問い合わせください。
特定の入力がONしない。	DIOの不良	サポートダイヤル(本マニュアルP.10に記載)までお問い合わせください。
	プログラムの不良	プログラムの修正
	入力線の配線ミス	入力線の配線チェック 入力線の断線チェック 入力端子の緩みチェック
	外部接続機器の異常	外部接続機器の交換
	入力のON時間が短い	入力のON時間を長くする
特定の入力がOFFしない。	DIOの不良	サポートダイヤル(本マニュアルP.10に記載)までお問い合わせください。
	プログラムの不良	プログラムの修正
入力エリアが不規則にON、OFFする。	外部入力電圧不良	定格電圧を供給する
	端子ネジの緩み	ネジ増し締め
	プログラムの不良	プログラムの修正
	コネクタの接触不良	コネクタを確実にネジで取り付ける
	ノイズによる誤動作	ノイズ対策をする サージキラーの取り付け シールドケーブルの使用

■DIOの出力異常

異常現象	推定原因	対処方法
出力モニタランプは点灯するが、まったく出力できない。	DIOの不良	サポートダイヤル(本マニュアルP.10に記載)までお問い合わせください。
	出力コモン線の配線ミス	コモン線の配線チェック コモン線の断線チェック コモン端子の緩みチェック
	負荷電源不良	定格電圧を供給する
	コネクタの接触不良	コネクタを確実にネジで取り付ける
出力モニタランプが消灯し、まったく出力できない。	DIOの不良	サポートダイヤル(本マニュアルP.10に記載)までお問い合わせください。
	プログラムの不良 出力エリアをすべてOFFにしている	プログラムの修正
	I/O使用可が未設定	I/O使用可の設定をする
出力すべてがOFFしない。	DIOの不良	サポートダイヤル(本マニュアルP.10に記載)までお問い合わせください。
特定の出力がONしない。	DIOの不良	サポートダイヤル(本マニュアルP.10に記載)までお問い合わせください。
	出力線の配線ミス	出力線の配線チェック 出力線の断線チェック 出力端子の緩みチェック
	外部接続機器の異常	外部接続機器の交換
特定の出力がOFFしない。	DIOの不良	サポートダイヤル(本マニュアルP.10に記載)までお問い合わせください。
	漏れ電流、残留電流による復帰不良	外部機器の設計変更 ダミー抵抗の追加など
出力エリアが不規則にON、OFFする。	負荷電源不良	定格電圧を供給する
	端子ネジの緩み	ネジ増し締め
	プログラムの不良 出力命令が重複している	プログラムの修正
	コネクタの接触不良	コネクタを確実にネジで取り付ける
	ノイズによる誤動作	ノイズ対策をする サージキラーの取り付け シールドケーブルの使用

■エラーコード

I/Oエラーは、I/Oの読み込み/書き込みのエラーです。I/Oエラーが発生すると、コントローラは#IOStatusにエラーコードを書き込みます。ロジックプログラムの実行は続けられます。発生するエラーの内容と対処方法を説明します。

◆設定エラー

エラーコード	内容	対処方法
501	I/Oターミナルに割り当てられる内部変数エラー	割り当てられている変数タイプを設定し直してください。
502	出力ターミナルに割り当てられる入力変数エラー	
503	入力ターミナルに割り当てられる出力変数エラー	
504	アナログターミナルに割り当てられるディスクリート変数エラー	
505	ディスクリートターミナルに割り当てられる整数変数エラー	
506	ドライバでサポートされていない変数タイプです	変数タイプを見直してください。
801	ターミナル番号が重複しています	2つ以上のターミナルが同じターミナル番号を使用しています。転送に失敗した恐れがあります。LTEファイルを再ダウンロードしてください。
802	重複モジュールがあります	モジュール番号が重複しています。2つ以上のモジュールを設定しないでください。
803	モジュール番号が1を越えています	モジュール番号を0に設定してください。
804	モジュール番号が1から始まっています	モジュール番号を0に設定してください。
805	ドライバのコンフィギュレーションエラーです	1つ以上のDIOドライバがI/Oツリーに追加されています。
821	DIOハードウェアユニットがありません。	LTEファイルには、現在接続されているLTよりも多くのモジュールが設定されています。
822	DIO - : ハードウェアユニットがありません。またはタイプが不正です。	内蔵DIOユニットから読み出したID番号が正しくありません。内蔵DIOユニットの故障が考えられます。

◆ランタイムエラー

エラーコード	内容	対処方法
840	読み出しデータが不正です。LT内蔵のDIOから2回連続して読み出した値が異なりました。	入力信号のON時間を長くしてください。
841	出力データが不正です。内部ループバックチェックにてLT内蔵のDIOから出力データの不正を検知しました。	ノイズ等の影響がないかご確認ください。
842	DIO- 出力データが不正です。	内部ループバックチェックで出力データの不正を検知しました。

◆内部エラー

エラーコード	内容	対処方法
850 ⋮ 864	ドライバエラー システム内に重大なエラーが発生しました。	エラーコードを記録して、サポートダイヤル(本マニュアルP.10に記載)までお問い合わせください。

MEMO

このページは、空白です。
ご自由にお使いください。

第12章
エラーと異常処理

12.1 エラーメッセージ

ここでは、LTの左下に表示されるエラーメッセージについて説明します。ここで説明するエラーメッセージはLT Editorに関するものだけです。
その他のエラーメッセージについては
参照→「LTユーザーズマニュアル」（別売）

エラーメッセージ	原因	対処方法
"Invalid ladder file"	LTにロジックプログラムファイルがダウンロードされていないか、またはLT上のロジックプログラムファイルが壊れています。	LT Editorからプロジェクトファイルをダウンロードし直してください。
"Fatal Error: Drive check Failed"	LT上のI/Oドライバが不正です。	ロジックプログラムファイルに設定されているI/OドライバとLTにインストールしたI/Oドライバが同じものか確認してください。
"Global Data Area Too Small"	ダウンロード中にファイルが壊れた可能性があります。	プロジェクトファイルを再ダウンロードしてください。解決されない場合は、サポートダイヤル（LT Editorのマニュアル裏表紙記載）までお問い合わせください。
"Can't Set Priority"	LTのシステムファイルに問題があります。ダウンロード中にファイルが壊れた可能性があります。	プロジェクトファイルを再ダウンロードしてください。
"Exception nnn:[mmm:ooo]"	ロジックプログラムに重大なエラーがあります。	エラーの内容をサポートダイヤル（裏表紙記載）までお問い合わせください。
"Watchdog Error"	コンスタントスキャンタイムがウォッチドッグタイムより長くなっています。	ウォッチドッグタイムをコンスタントスキャンタイムより長く設定してください。

エラーメッセージ	原因	対処方法
"Bad Var: xxx"	変数名「xxx」が見つかりません。次の二つの原因が考えられます。 ・ロジックプログラムファイルがダウンロードされていない。 ・画面上でロジックプログラムファイルに存在しない変数を使用している。	それぞれの対処方法は以下の通りです。 ・ロジックプログラムファイルをダウンロードしてください。
"Bad Array: xxx"	画面ファイルの要素数とロジックプログラムファイルの配列変数の要素数が違います。	プロジェクトファイルを再ダウンロードしてください。
"Bad Type xxx"	ロジックプログラムの変数「xxx」の変数タイプと画面の変数タイプとが異なります。	プロジェクトファイルを再ダウンロードしてください。
"Unknown register type"	変数タイプが存在しません。	プロジェクトファイルを再ダウンロードしてください。
"Register is missing"	書き込もうとした変数が見つかりません。	
"S100 file index is out of range"	読み出そうとした変数が見つかりません。	
"Too many entries in the S100 file"	変数の数が多すぎます。 使用できる変数は2048までです。	
"S100 file is missing"	S100ファイル（変数格納ファイル）が見つかりません。	
"Over Compile count MAX"	使用している部品が多すぎます。	使用している部品を減らして、プロジェクトファイルを再ダウンロードしてください。
"Exception 65532 [xxxx:xxx]" "Exception 65533 [xxxx:xxx]" "Exception 65534 [xxxx:xxx]" "Exception 65535 [xxxx:xxx]"	LT上のヒープメモリが不足しています。プログラム、変数を格納するメモリは足りていますが、ロジックプログラムを実行するメモリが不足しています。	ロジックプログラム、変数またはラベルを減らしてLT Editor上でLTを再セットアップしてください。配列変数の場合、要素数を減らすことも有効です。また、変数名、ラベル名を短くすることも有効です。

12.2 エラーコード

ここではエラーが発生した際に書き込まれる、#FaultCode のエラーコードについて説明します。

エラーコード	程度	原因
0	正常	エラーはありません。
1	マイナー	算術命令の結果、または実数から整数への変換結果がオーバーフローしました。
2	メジャー	配列の領域を越えて参照されました。
3	メジャー	整数（32ビット）の範囲を超えてビットが参照されました。
4	メジャー	スタックがオーバーフローしました。
5	メジャー	不正な命令コードを使用しています。
6	-	システムで予約
7	メジャー	スキャンタイムがウォッチドッグタイムを越えました。
8	-	システムで予約
9	メジャー	ソフトウェアのエラーです。 場合によっては、システムを再起動する必要があります。
10	-	システムで予約
11	-	システムで予約
12	マイナー	BCD/BIN変換エラー
13	マイナー	ENCO/DECOエラー
14	-	システムで予約

「メジャー異常」と「マイナー異常」

メジャー異常が発生すると、コントローラはすぐにロジックプログラムを停止します。

マイナー異常はロジックプログラムの実行が可能なエラーです。

エラーの原因を確認してください。

12.3 プログラムの動作異常

ここではLT Editorのロジックプログラムの動作異常について説明します。

異常現象	推定原因	対処方法
コントロールメモリの電源断保持エリアが保持されない。	電池異常	電池異常
	メモリ異常	本機交換
プログラムが正常に動作しない。	プログラムの転送ミス	LT Editorで、プロジェクトファイルを再ダウンロードする。
STOPモードになっているのにI/O出力される。	出力データRUN/STOP切り換え時、保持クリア制御機能が有効になっている。	当機能を無効にする。
RUNモードになるが、すぐにSTOPモードになってしまう。	命令実行異常などが発生している。または、メジャー異常が発生している。	プログラムを見直し、システム変数 #FaultCodeの内容を確認し、プログラムを修正する。 システム変数 #Commandに書き込みがないか確認し、プログラムを修正する。
LT Editorでモニタリングモードに入れない。	画面作成ソフトからのプロジェクトファイルのダウンロード中に、転送ケーブルが抜けた、LTやパソコンの電源が落ちたなどの電気的ノイズの可能性があります。	転送ケーブルが抜けていないか、ノイズなどの影響がないか確認する。 解決しない場合は、(株)デジタルサポートダイヤルまでご連絡ください。
LT Editorからロジックプログラムファイルをダウンロードできない。		
LT Editorからプロジェクトファイルがダウンロードできない。		
I/O入出力ができない。	I/O使用可※1の設定ができていない。	I/O使用可の設定をしてください。

※1 I/O使用可とは、LT本体や、I/Oユニットへの入出力を可能にする動作です。LTでは、ロジックプログラムのダウンロードを行った後、LTを運転状態にしただけでは外部I/O機器の入出力を行うことはできません(安全のためデフォルトではI/O使用可は設定されていません)。

I/O入出力を行う場合、あらかじめI/O使用可の設定が必要です。

設定方法については「プログラミング編」を参照してください。

付録

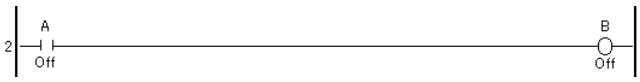
サンプルプログラムのプログラムサイズ使用率、変数のメモリ使用率、ラングのメモリ使用率を示します。

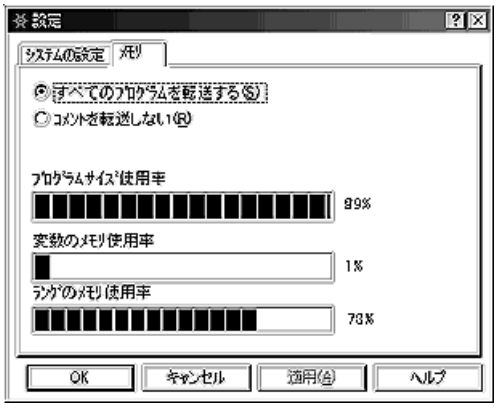
付 .1 メモリの使用率

LT Editorは、ソフトウェアPLCを採用しているために一般のPLCのようなステップ計算によるプログラム容量計算ができません。サンプルプログラムの各メモリの使用率を示しますので参考としてください。

付 .1.1 ディスクリート変数＆基本回路

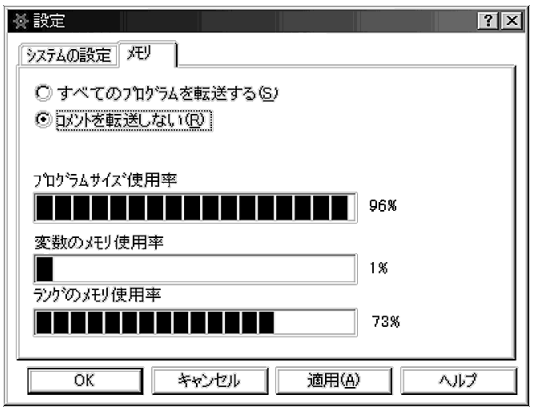
宣言した変数A(ディスクリート)、B(ディスクリート)において、変数はほとんど使用していないため、変数のメモリ使用率は1%となっています。また、4500ラングを使用するとプログラムサイズ使用率は、ほぼ最大の100%となります。

変数構成（システム変数含む）		ラング＆命令構成
ディスクリート	2	 × 4500 ラング
ディスクリート配列	0	
整数	0	
整数配列	0	
実数	0	
実数配列	0	
タイマ	0	
カウンタ	0	



すべてのプログラムを転送する(S)

プログラムサイズ使用率：99%
変数のメモリ使用率：1%
ラングのメモリ使用率：73%

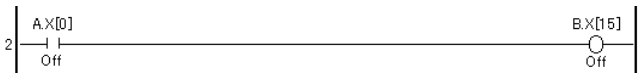


コメントを転送しない(R)

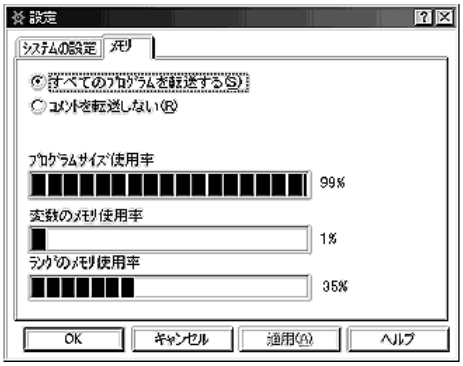
プログラムサイズ使用率：96%
変数のメモリ使用率：1%
ラングのメモリ使用率：73%

付 .1.2 整数変数＆基本回路 1

「付 .1.1 ディスクリート変数＆基本回路」に対し、宣言した変数を配列で使用しているものです。変数を配列で扱うことで、プログラム容量において「付 .1.1 ディスクリート変数＆基本回路」では 4500 ラング扱えたものが 2200 ラングとなっています。

変数構成（システム変数含む）		ラング＆命令構成
ディスクリート	0	
ディスクリート配列	0	
整数	2	
整数配列	0	
実数	0	
実数配列	0	
タイマ	0	
カウンタ	0	

× 2200 ラング



すべてのプログラムを転送する(S)

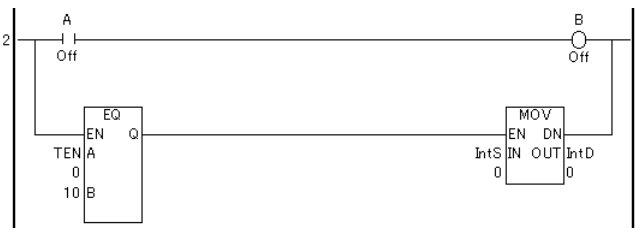
プログラムサイズ使用率：99%
変数のメモリ使用率：1%
ラングのメモリ使用率：35%

コメントを転送しない(R)

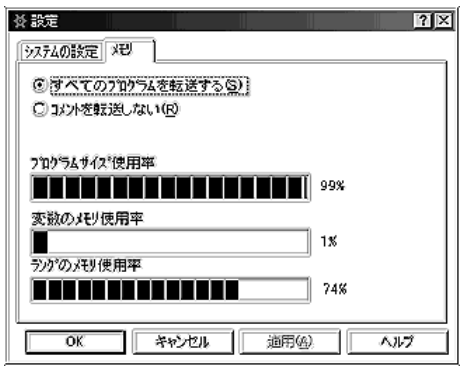
プログラムサイズ使用率：95%
変数のメモリ使用率：1%
ラングのメモリ使用率：35%

付 .1.3 整数変数＆基本回路 2

宣言した変数A(ディスクリート)、B(ディスクリート)、TEN(整数)、IntS(整数)、IntD(整数)の複合で使用した場合、1820ラングを使用するとプログラムサイズ使用率は、ほぼ最大の100%となります。

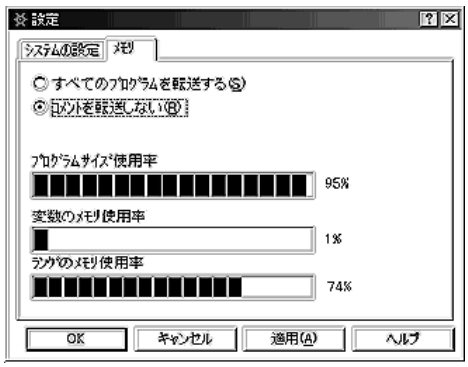
変数構成（システム変数含む）		ラング＆命令構成
ディスクリート	2	
ディスクリート配列	0	
整数	3	
整数配列	0	
実数	0	
実数配列	0	
タイマ	0	
カウンタ	0	

× 1820 ラング



すべてのプログラムを転送する(S)

プログラムサイズ使用率：99%
変数のメモリ使用率：1%
ラングのメモリ使用率：74%

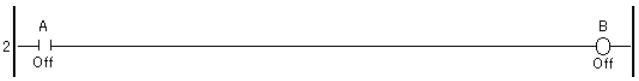


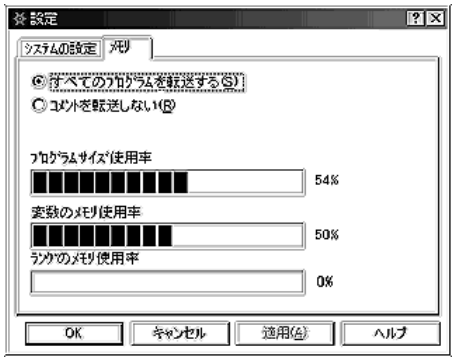
コメントを転送しない(R)

プログラムサイズ使用率：95%
変数のメモリ使用率：1%
ラングのメモリ使用率：74%

付 .1.4 整数変数（グローバル）

宣言した変数A(ディスクリート)、B(ディスクリート)、整数変数を最大設定数の2000個（グローバル）を使用したときのメモリの使用率を示します。

変数構成（システム変数含む）		ラング＆命令構成
ディスクリート	2	
ディスクリート配列	0	
整数	2000	
整数配列	0	
実数	0	
実数配列	0	**×1ラング**
タイマ	0	
カウンタ	0	



すべてのプログラムを転送する(S)

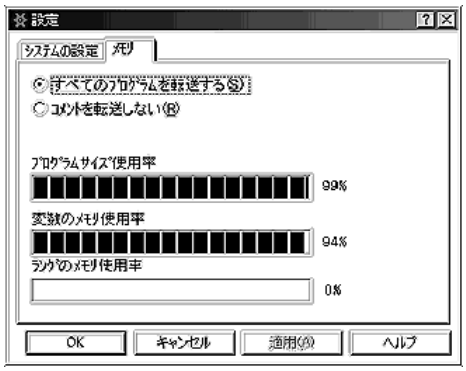
プログラムサイズ使用率：54%
変数のメモリ使用率：50%
ラングのメモリ使用率：0%

付 .1.5 整数変数（ローカル）

「付 .1.4 整数変数（グローバル）」に対して整数変数をローカルに設定したとき、整数変数は 3790 個使用するとプログラムサイズ使用率は、ほぼ最大の 100% となります。

変数構成（システム変数含む）		ラング＆命令構成
ディスクリート	2	
ディスクリート配列	0	
整数	3790	
整数配列	0	
実数	0	
実数配列	0	
タイマ	0	
カウンタ	0	

× 1 ラング

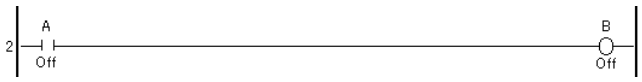


すべてのプログラムを転送する(S)

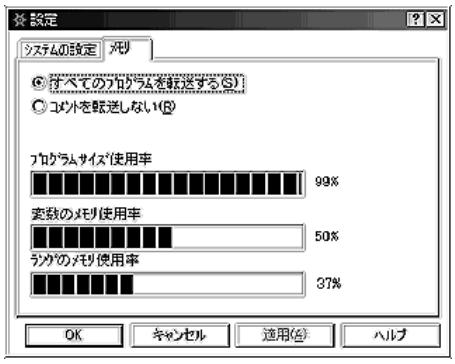
プログラムサイズ使用率：99%
変数のメモリ使用率：94%
ラングのメモリ使用率：0%

付 .1.6 整数変数＆基本回路 3

「付 .1.4 整数変数（グローバル）」に対して、2300 ラングを使用するとプログラムサイズ使用率は、ほぼ最大の 100% となります。

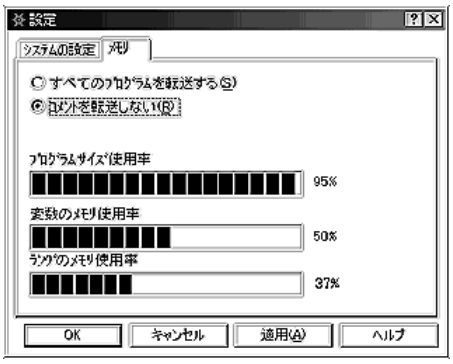
変数構成（システム変数含む）		ラング＆命令構成
ディスクリート	2	
ディスクリート配列	0	
整数	2000	
整数配列	0	
実数	0	
実数配列	0	
タイマ	0	
カウンタ	0	

× 2300 ラング



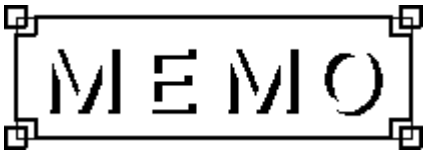
すべてのプログラムを転送する(S)

プログラムサイズ使用率：99%
変数のメモリ使用率：50%
ラングのメモリ使用率：37%



コメントを転送しない(R)

プログラムサイズ使用率：95%
変数のメモリ使用率：50%
ラングのメモリ使用率：37%



このページは、空白です。
ご自由にお使いください。

索引

サ

シ

ス

セ

ソ

タ

テ

ト

ニ

ハ

ヒ

フ

ヘ

ホ

メ

モ

ユ

ヨ

ラ

リ

ロ

ワ

MEMO

このページは、空白です。
ご自由にお使いください。