はじめに

このたびは、GLC用ロジックプログラム開発ソフト「Pro-Control Editor Ver.4.0」をご採用いただき、誠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) 「Pro-Control Editor」(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「ソフトウェア使用許諾条件」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) 本製品を使用したことによるお客様の損害、および免失利益、または第三者からのいかなる請求につきましても、当社はその責任を負いかねますので、あらかじめご了承ください。

(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三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 2001 November

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	米国Microsoft社
Intel, Pentium	米国Intel社
Pro-face, Flex Network	(株)デジタル
Ethernet	米国Western Digital社
NEC, PC-9800	日本電気(株)
IBM, VGA, PC/AT	米国IBM社
Adobe, Acrobat	アドビシステムズ社

なお、上記商号･商標類で、本書での表記と正式な表記が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
MS-DOS	Microsoft® MS-DOS® 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

対応機種一覧

Pro-Control Editor Ver.4.0で対応している機種は以下のとおりです。本書では以下のようなシリーズ名または商品名を用いて説明します。下表の「GPタイプ」とは、GP-PRO/PB Ⅲ for Windows Ver.6.0で選択する時の名称です。

シリーズ名		商品名	型式	GPタイプ
GLC100シリーズ	GLC100シリーズ	GLC100L	GLC100-LG41-24V	GLC100L
		GLC100S	GLC100-SC41-24V	GLC100S
GLC300シリーズ	GLC300シリーズ	GLC300T	GLC300-TC41-24V	GLC300T
GLC2000シリーズ	GLC2300シリーズ	GLC2300L	GLC2300-LG41-24V	GLC2300L
		GLC2300T	GLC2300-TC41-24V	GLC2300
	GLC2400シリーズ	GLC2400T	GLC2400-TC41-24V	GLC2400
	GLC2600シリーズ	GLC2600T	GLC2600-TC41-24V	GLC2600

マニュアルの読み方

「GP-PRO/PB Ⅲ C-Pack01」のマニュアルは7冊で構成されています。マニュアルの内容は別記の表をご覧ください。マニュアルはPDFファイルとしてCD-ROM(Disc2)に収録されています（導入ガイドのPDFファイルはありません。）。また、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場合があります。［スタート］ボタンをクリックし、［プログラム］、［Pro-face］、［ProPB3 C-Package］の順にポイントし、［お読みください］をクリックして表示された内容をご覧ください。

（株）デジタル製ハードウェアに関する詳しい説明は、各機種の「ユーザーズマニュアル」（別売）をご覧ください。

GP-PRO/PBIII C-Package01	
導入ガイド	**インストール方法と基本的な使い方について説明します。**
Pro-Control Editor Ver.4.0	
ユーザーズマニュアル（本書）	**GLCとの組み合わせに関するソフトウェア的な設定や変数、命令について説明します。**
オペレーションマニュアル	**準備から運転までの操作を習得するための演習とエラーメッセージの一覧を説明します。Pro-Control Editorで登録した変数をGP-PRO/PBIIIで使用する方法などについても説明します。**
GP-PRO/PBIII for Windows Ver.6.0	
オペレーションマニュアル	**GP画面を作成するソフトウェアの操作手順と機能を説明します。**
タグリファレンスマニュアル	**GPの画面上機能を指定する「タグ」について説明します。**
パーツリスト	**GP画面を作成するソフトに用意されているパーツと図記号について説明します。**
機器接続マニュアル（PLC接続マニュアル）	**GPと各社PLC、温度調節器、インバータなどの接続について説明します。**

※ GP-PRO/PBIIIのマニュアルはGP画面の作成について書かれています。GLCの作画を行う場合は、GPをGLCに読み替えて操作してください。

※ 上記のPDFマニュアル以外にもオンラインヘルプで詳しく説明していますのでご覧ください。

タグなどのアドレス設定時は標準インストール時にインストールされるレイアウトシートを利用されると便利です。

レイアウトシートには「デバイス割り付け表」と「タグレイアウトシート」があります。

それぞれMicrosoft Excelデータとしてインストールされているのでご利用ください。

各ファイルの場所とファイル名を以下に示します。

なお、Microsoft Excelのご利用方法は該当商品マニュアルを参照ください。

フォルダ名	ファイル名	内容
Pro-face\propbwin\sheet	**Device1J.xls**	**デバイス割り付け表**
	TAG1J.xls、TAG2J.xls、TAG3J.xls、TAG4J.xls	**タグレイアウトシート**

CD-ROM内のPDFマニュアルはAdobe Acrobat Readerで閲覧できます。

お問い合わせ

Pro-Control 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**

(ただし祝祭日および弊社夏期・冬季休暇期間を除く)

東京TEL　　(03)5821-1105

名古屋TEL　　(052)932-4093

大阪TEL　　(06)6613-3115

目次

第4章　命 令

第5章　LSエリアリフレッシュ

第6章　I/Oドライバ

表記のルール

本書は、以下のルールで表記します。

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、(株)デジタル製品についての技術的なご質問・ご相談にお答えします。

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。

■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。
※	脚注で説明している語句についています。
Pro-Control Editor	GLC のロジックプログラムを作成/転送/モニタを行う機能をもったソフトウェアです。
コントローラ	GLC に組み込まれている制御機能を指します。
GP-PRO/PBⅢ (画面作成ソフト)	GP-PRO/PBⅢ for Windows Ver.6.0 を指します。
GLC	(株)デジタル製グラフィック ロジック コントローラの総称です。
外部通信機器	PLC(プログラマブルコントローラ)、温調器、インバータなどの周辺機器を指します。ただし、Flex Network、ユニワイヤ、DIO で接続する機器を除きます。

使用上の注意

⚠警 告

・ GLCでは人命や重大な物的損傷にかかわる制御は決して行わないでください。

■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。**

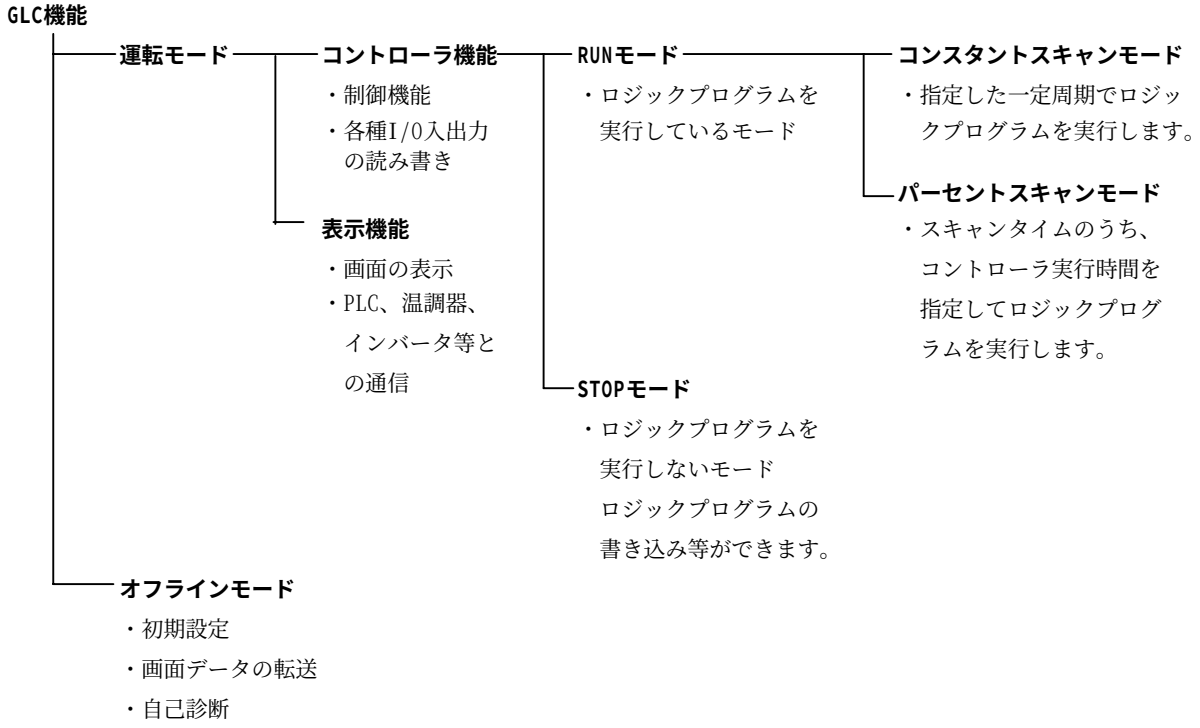
・ **CD-ROMの記録面の磁性体面(シャッターの中)に手を触れないでください。**

・ **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。**

第1章 コントローラ機能

1.1 動作モード概要

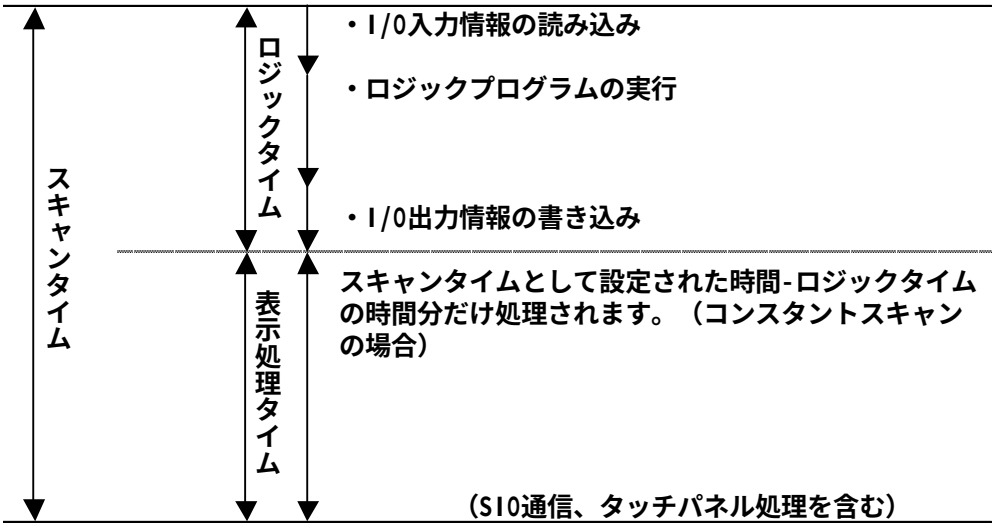
GLCには、画面の表示を実行する機能と制御を実行するコントローラ機能があります。
GLCの動作モードの概要を以下に示します。



重要 **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GLCはリセットされます。**

1.1.1 GLCのスキャンの概略

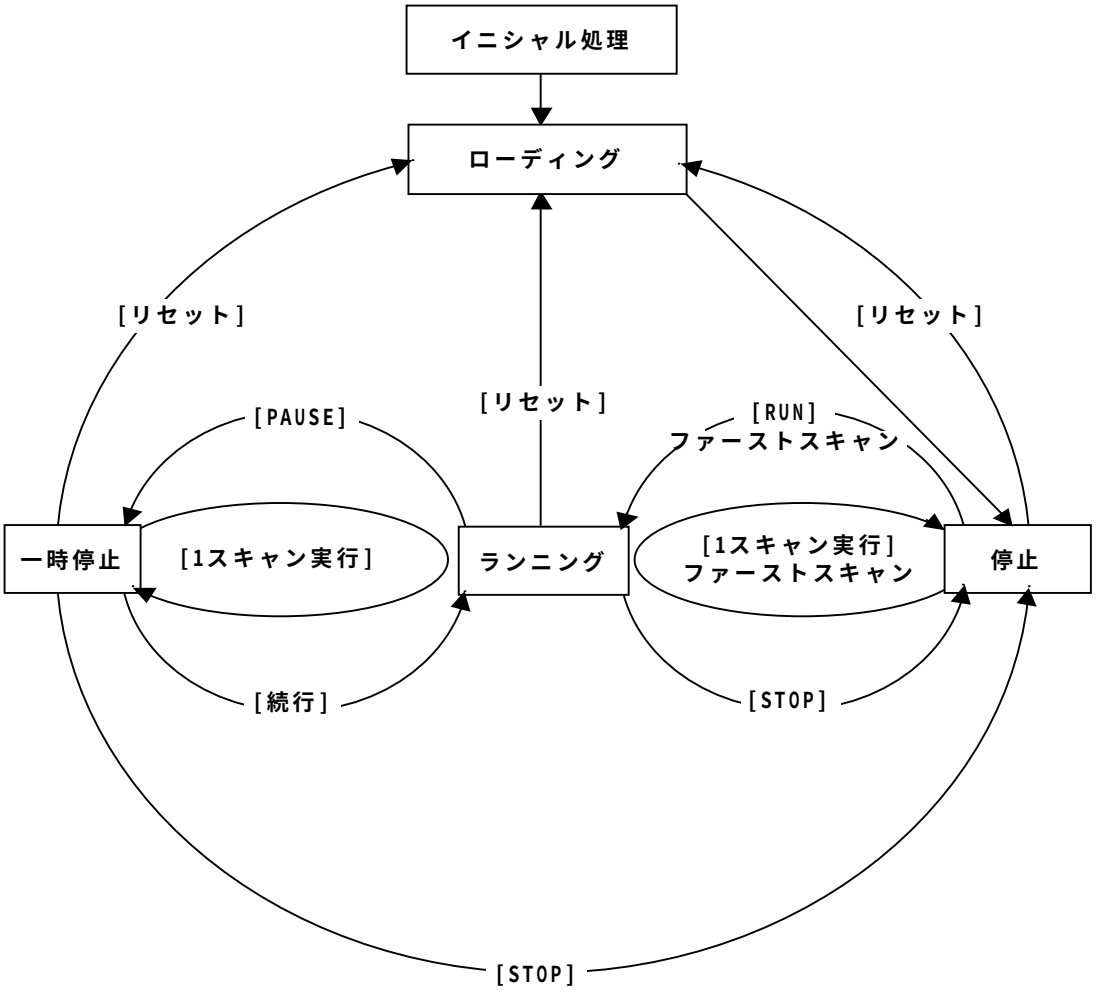
GLCのスキャンタイムはロジックプログラムを実行するコントローラと画面、SIO通信、タッチパネル処理が含まれ下図のようになっています。



参照→1.1.3 RUNモードの流れ

1.1.2 コントローラ機能概略遷移図

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

この状態においてロジックプログラムをプログラムの格納メモリから[RUN]可能なメモリへ読み込みます。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。GLCのオフラインモードにあるコントローラ設定で「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。

コントローラ設定については参照→各GLCのユーザーズマニュアル

■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GLC本体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

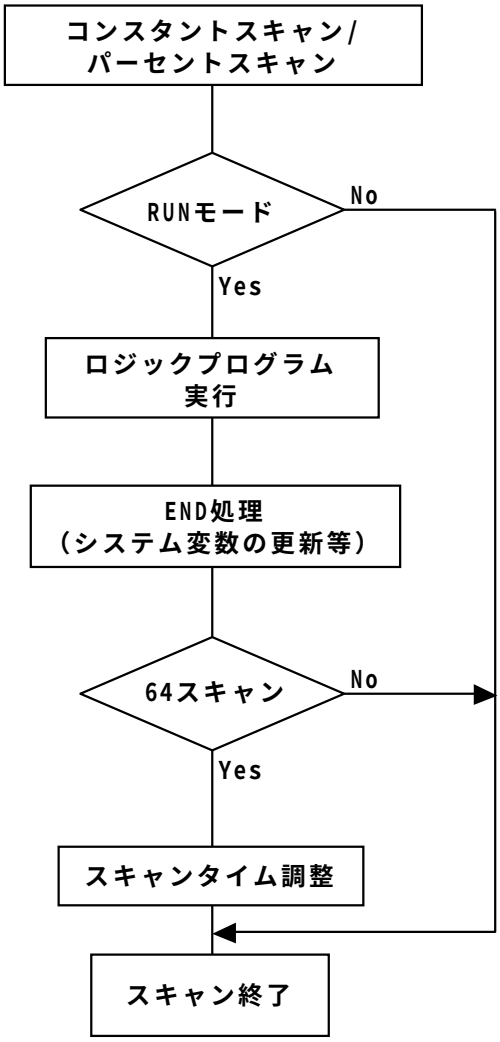
この状態はロジックプログラム実行エンジンの一時停止状態です。I/Oのウォッチドッグタイムアウトを避けるため、I/O読み込みとI/O書き込みを実行します。しかし、ロジックプログラムを実行しないため、出力の状態は変化しません。コマンドを受けるとそれぞれの状態に移ります。

[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]コマンドでロジックプログラムを1回だけ(1スキャン)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

1.1.3 RUN モードの流れ

RUNモードの流れの概要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3章 システム変数

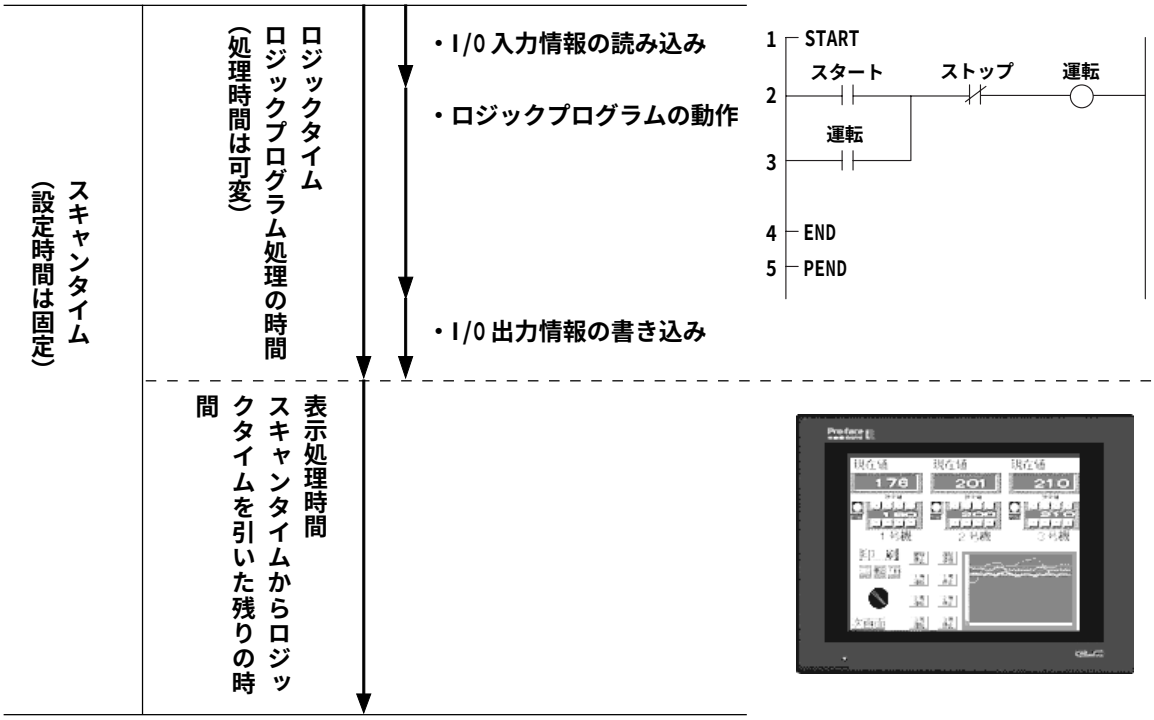
重要 **GLCのスキャン時間には以下の誤差が含まれます。**

機種	誤差
GLC100シリーズ	約 -0.2%
GLC300シリーズ GLC2000シリーズ	約 -0.02%

■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、GLC 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、ロジックタイムがスキャンタイムの50%になるようにスキャンタイムが自動調整されます。**

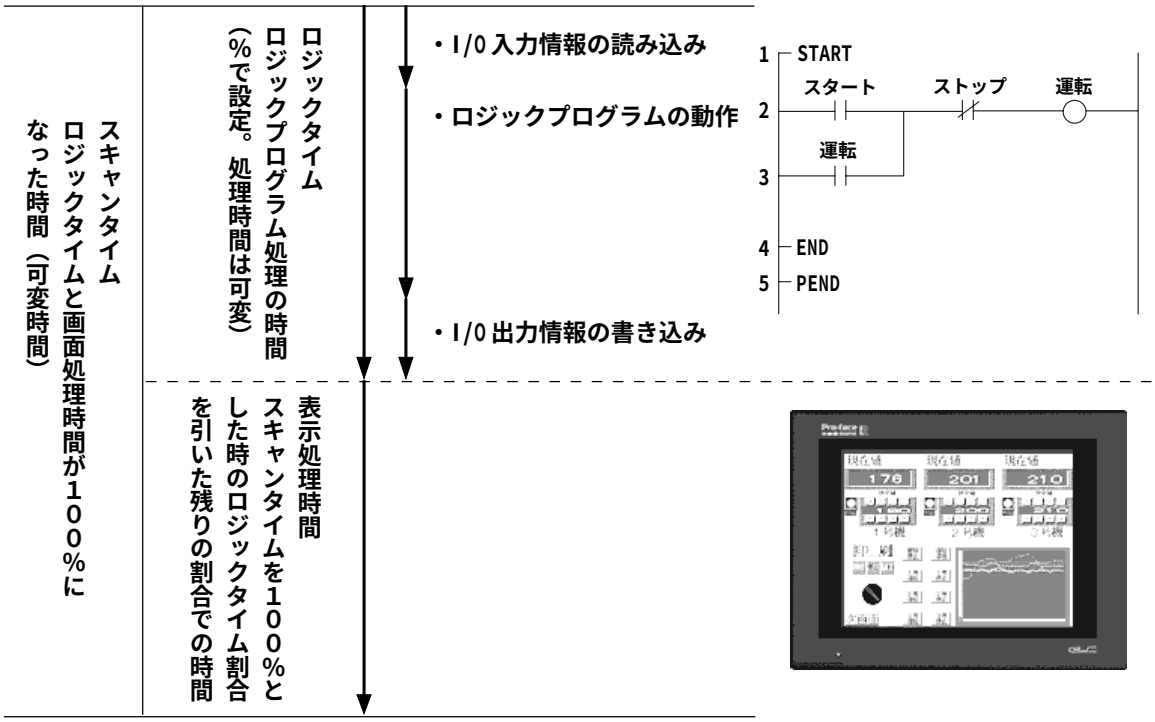
例)コンスタントスキャン設定値が 50ms の場合
ロジックタイムが 30ms の時はスキャンタイムは 60ms になります。

MEMO スキャンタイムの設定は 10ms 単位で入力してください。

■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、

画面処理時間　＝　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、GLC 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。**

MEMO 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。

第2章 変数

ここでは、Pro-Control Editorで用いられる変数について説明します。

2.1 変数名

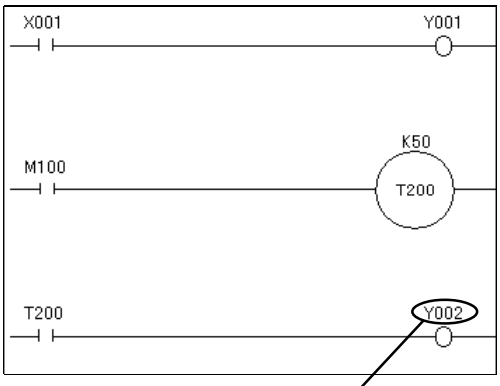
Pro-Control Editorでは、I/Oやカウンタのデータを格納するエリアとして変数を使用します。変数はユーザーで定義し、ロジックプログラムでそのままの名前で使用します。

従来の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)	三菱電機(株)	X001	M100	T200	D00001	
	オムロン(株)	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

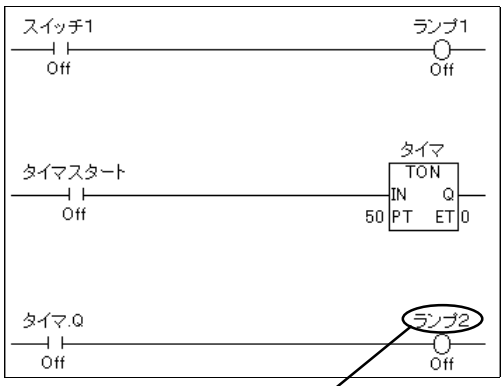
Pro-Control Editorではこのようなデバイスアドレスに任意の名前を付け、**変数**という形でロジックプログラム中で使用します。

PLC



各メーカーごとの
デバイスアドレス

Pro-Control Editor



変数名
(Pro-Control Editor
でユーザーが設定した
任意の名前)

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。
- 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。
  参照→第 5 章 LS エリアリフレッシュ

MEMO
・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、Pro-Control 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	Pro-Control Editor 配列変数	Pro-Control Editor 変数タイプ	PLC デバイス	Pro-Control Editor 配列変数	Pro-Control 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→「Pro-Control Editor オペレーションマニュアル 2.4. 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 3 章 システム変数**

2.2 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。

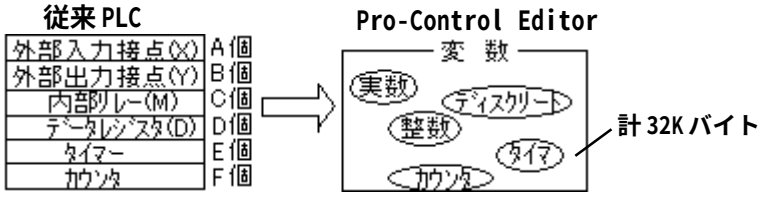
ディスクリート、整数、実数の各変数は、配列指定もできます。配列指定の詳細については、**「2.3 変数へのアクセス」**を参照してください。

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、GLC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

PLCでは各デバイスごとに使用数に制限がありますが、GLC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Pro-Control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、-2147483648～214783647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。

MEMO
- 実数変数をGLCの画面に表示する場合はGP-PRO/PBⅢのEタグで表示データ形式をFloat(32ビット)にします。
- 64ビットの実数から32ビットに変換されるため、誤差が生じます。
- 整数変数は32ビット長のため、GLCの表示機能で16ビット長で使用する場合、下位16ビットが有効になります。

参照→「GP-PRO/PB III タグリファレンスマニュアル Eタグ」

■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。

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

変数名の後にピリオドと専用変数名を付けると、その専用変数を参照できます。

例）タイマ .ET

詳しくは、**4.2 命令詳細**を参照してください。

タイマ変数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。

◆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。

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

変数名の後にピリオドと専用変数名を付けると、その専用変数を参照できます。

例）カウンタ .CV

詳しくは、**4.2 命令詳細**を参照してください。

・カウンタ変数が非保持に指定されていても、専用変数**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。

■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

GLC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→「Pro-Control Editor オペレーションマニュアル　2.11.I/Oの割り付け」**

◆保持

保持型変数は静的なメモリで管理されるため、電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GLC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、CLCの PRW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。GP-PRO/PB Ⅲでタグや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GLCシンボルとして登録され、GP-PRO/PB Ⅲ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。

参照→「Pro-Control Editor オペレーションマニュアル　2.4. 変数の作成」

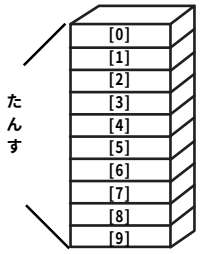
2.3 変数へのアクセス

Pro-Control Editorにて、変数の配列の要素・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。

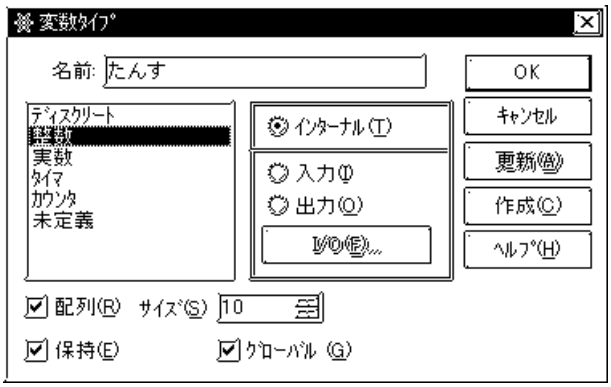
■配列変数

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・整数配列へのアクセス

整数・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。

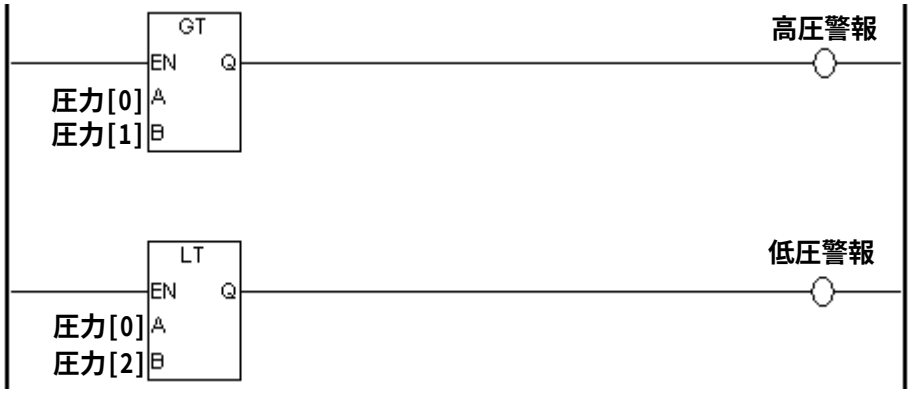
1日目	飲料水売り上げ[0]
2日目	飲料水売り上げ[0]
3日目	飲料水売り上げ[0]
4日目	飲料水売り上げ[0]
	-
	-
	-
28日目	飲料水売り上げ[0]
29日目	飲料水売り上げ[0]
30日目	飲料水売り上げ[0]
31日目	飲料水売り上げ[0]

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。

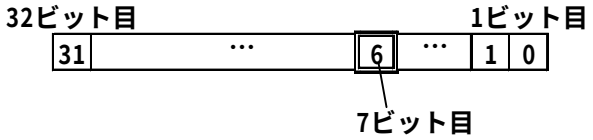
現在の圧力	圧力[0]
圧力上限値	圧力[1]
圧力下限値	圧力[2]



◆整数配列でビット単位のアクセスをする場合

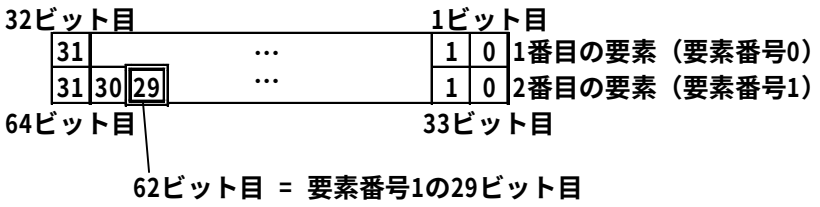
また、整数配列はディスクリート配列変数と同様に、ビット・バイト・ワード単位でのアクセスと組み合わせてアクセスすることもできます。整数配列変数**飲料水売り上げ**の **n+1**番目の要素の **m+1**ビット目にアクセスするには**飲料水売り上げ[n].X[m]**のようになります。

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]**



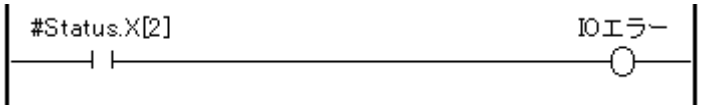
したがって、**飲料水売り上げ .X[61] = 飲料水売り上げ[1].X[29]**となり、どちらも整数配列**飲料水売り上げ**の62ビット目へのアクセスになります。

・ 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、GLC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： GP-PRO/PB Ⅲ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24時間に1回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。

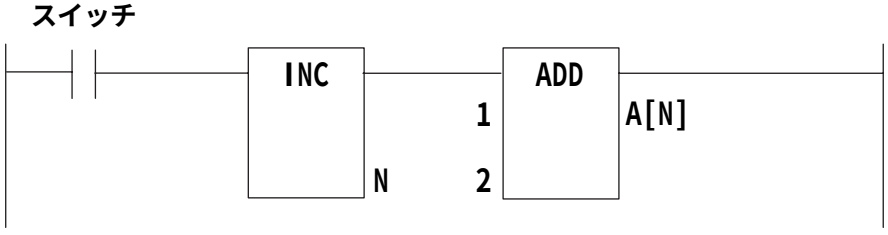
整数要素0は、0:00の温度データに対応します。

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1と2を加算して、A[N]に代入すると、A[N]に3が代入されます。5回スキャンしたとすれば、A[0]、A[1]、A[2]、A[3]、A[4]それぞれに3が代入されます。ただし、これはNの値が初期値(0)のままであった場合にのみ成り立ちます。



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 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。

3.1 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	#AvgLogTime	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	
	#Day	「日」をBCD2桁で格納しています。	-	整数	
	#EditCount	GLC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	
	#Month	「月」をBCD2桁で格納しています。	-	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	GLCでは現在は使用されていません。	-	ディスクリート	
	#Time	「時分」をBCD4桁で格納しています。	-	整数	
	#Version	コントローラのバージョンを示します。	-	整数	
	#WCLScan	GLCでは現在は使用されていません。	-	整数	
	#WCLStatus	GLC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	

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	
	#LadderMonitor	メーカー予約	0	整数	
	#PercentAlloc	パーセントスキャンを設定します。(単位:%)	0	整数	
	#PercentMemCheck	GLCでは現在は使用されていません。	-	整数	
	#RungNo	メーカー予約	0	整数	
	#Screen	画面番号を指定することで、GLCの画面を切り替えます。	-	整数	
	#StopScans	GLC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	
	#WatchdogTime	ウォッチドッグタイマ値を設定します。(単位:ms)	-	整数	

MEMO

- #Year、#Month、#Day、#Time はGLCの時計データを格納しています。時計データの変更はGLC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→各GLCシリーズのユーザーズマニュアル
  GP-PRO/PB III 機器接続マニュアル(PLC接続マニュアル)
- #Clock100ms、#Day、#Month、#Time、#Year、#Screen は、GLC2000シリーズのみ対応です。

3.1.1 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。

3.2 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。

3.2.1 #AvgLogicTime

#AvgLogicTimeは、平均ロジックタイムをms単位で格納します。

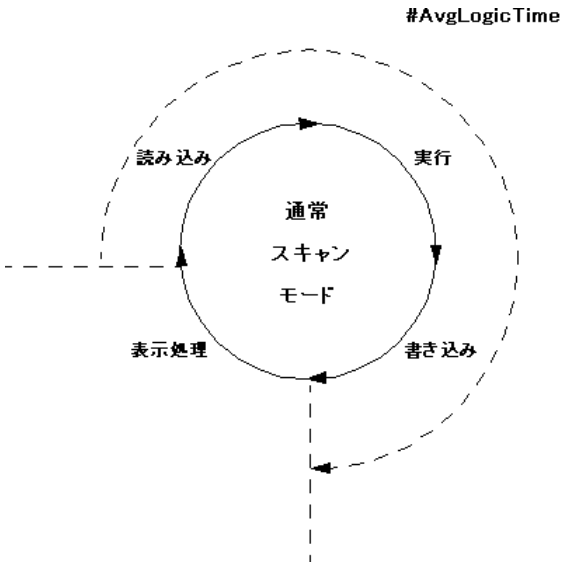
平均ロジックタイムとは、1回のスキャンでI/Oの読み込み、ロジックプログラムを実行し、I/Oの書き込みまでに必要な時間の平均です。

#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



3.2.2 #AvgScanTime

#AvgScanTimeは、平均スキャンタイムをms単位で格納します。

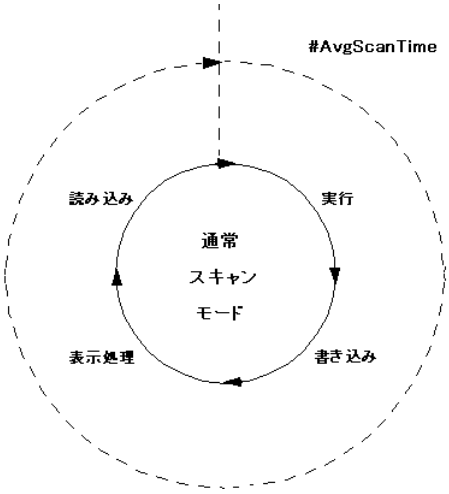
平均スキャンタイムとは、1回のスキャンでI/Oの読み込み、ロジックプログラムの実行、I/Oの書き込み、表示処理までに必要な時間の平均です。

#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



3.2.3 #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。初期値は未定です。

変数タイプ: ディスクリート

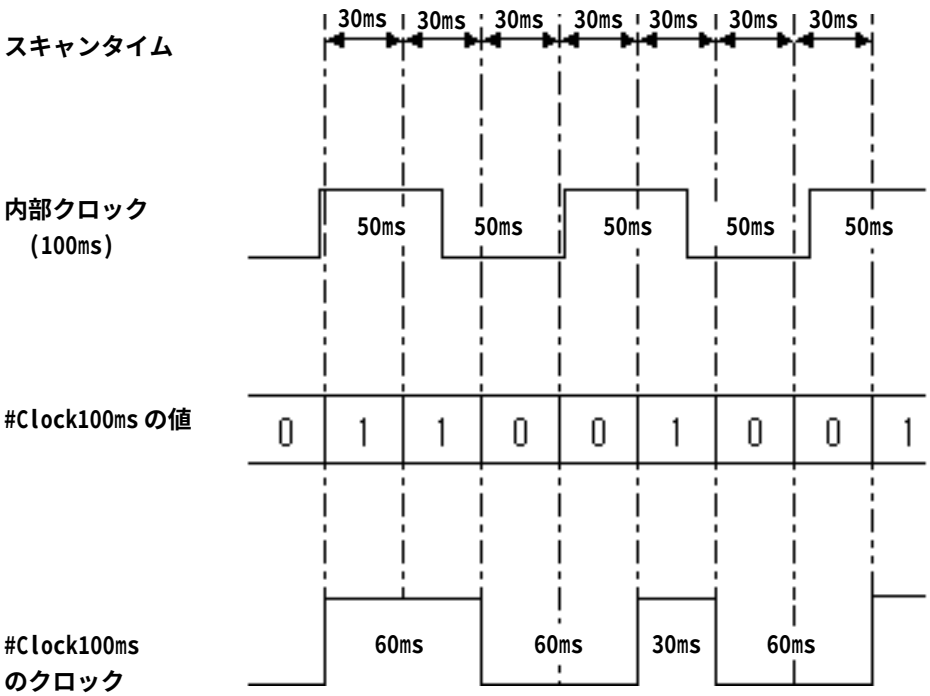
設定: コントローラ

読み込み専用

MEMO

・GLC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、GLC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。
・#Clock100msは、GLC2000シリーズのみ対応です。

◆スキャンタイムが30msの場合



3.2.4 #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	01	07	14	0619

MEMO ・ #Day は、GLC2000 シリーズのみ対応です。

3.2.5 #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、**「Pro-Control Editor オペレーションマニュアル 4.4. ディスクリート変数を強制的に ON/OFF する」**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。

3.2.6 #IOStatus

#IOStatus はI/0 ドライバによりセットされます。I/0 ドライバの現在の状態を格納します。

I/0 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/0 ドライバは正常です。0 以外の値のときは、I/0 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/0 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。

MEMO I/0 ドライバのエラーコードの内容については、
「第6 章I/0 ドライバ」をご覧ください。

3.2.7 #LogicTime

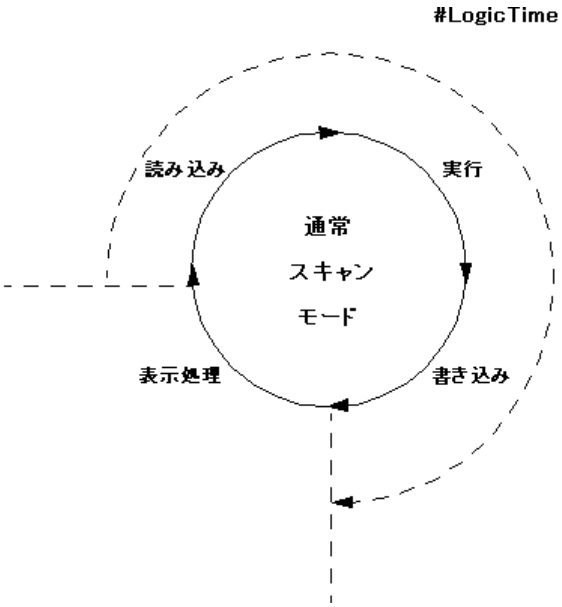
#LogicTime は、前回のスキャンのロジックタイムを表します。

ロジックタイムとは、1 回の スキャンでI/0 の読み込み、ロジックプログラムの実行、I/0 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



3.2.8 #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	01	07	14	0619

・ #Month は、GLC2000 シリーズのみ対応です。

3.2.9 #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
2	GLC100
4	GLC300
11	GLC2400
12	GLC2300
13	GLC2600

3.2.10 #ScanCount

#ScanCount はカウンタで、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 ～ 4294967295 です。最大値(4294967295)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

MEMO #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。

3.2.11 #ScanTime

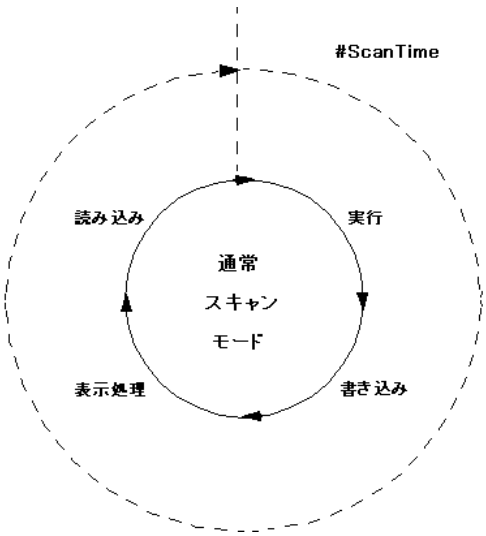
#ScanTime は、最新のスキャンタイムを ms 単位で格納します。

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



3.2.12 #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。

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

3.2.13 #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	01	07	14	0619

・ #Timeは、GLC2000シリーズのみ対応です。

3.2.14 #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.4.0.0の場合
バイト3	メジャーバージョン	04
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	――

3.2.15 #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	01	07	14	0619

・ #Yearは、GLC2000シリーズのみ対応です。

3.2.16 #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。

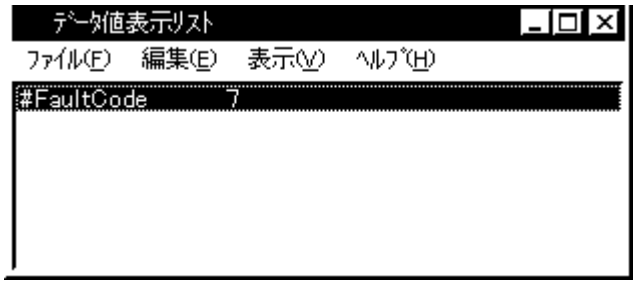
変数タイプ: 整数

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	メジャー	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エラー※1
14		システムで予約
15	マイナー	バックアップメモリ(SRAM)のロジックプログラムが壊れています。FEPROMのロジックプログラムが実行されます。※1

※1 GLC2000シリーズのみで発生するエラーです。



[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。

3.2.17 #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 が設定されています。

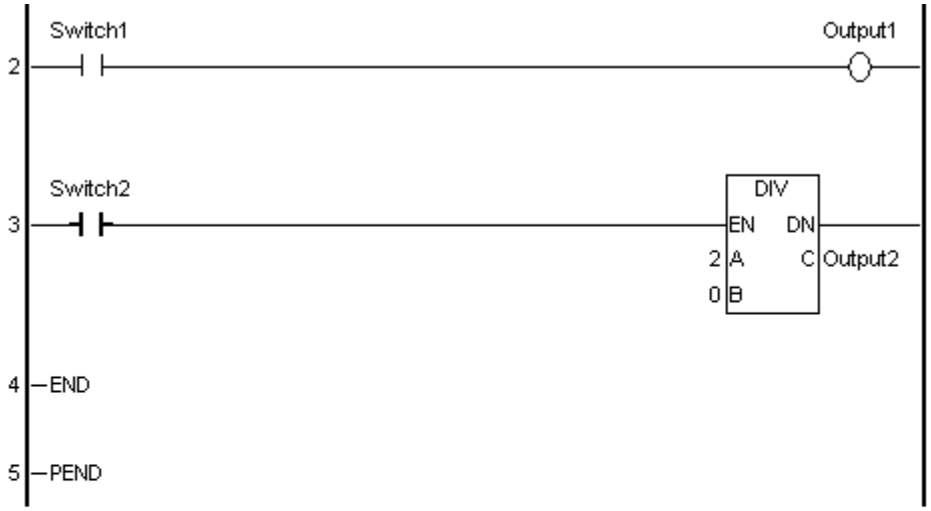
以下の例では、ラング 3 でエラーが発生したことが示されています。

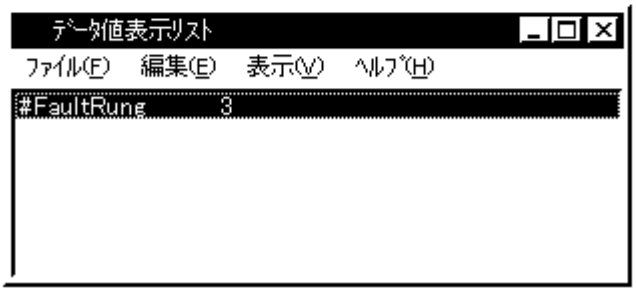
このエラーは、DIV 命令を実行したときに、整数を 0 で除算したことが原因です。次のエラーが発生するか、またはコントローラをリセットするまで残ります。

変数タイプ: 整数

設定: コントローラ

読み込み専用





3.2.18 #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。

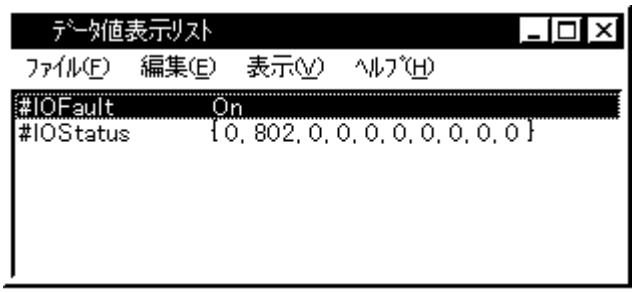
#IOStatus を確認すると、I/O ドライバの詳細な状態を知ることができます。

#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



MEMO I/O ドライバのエラーコードの内容については、「第６章 I/O ドライバ」をご覧ください。

3.2.19 #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。

"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。

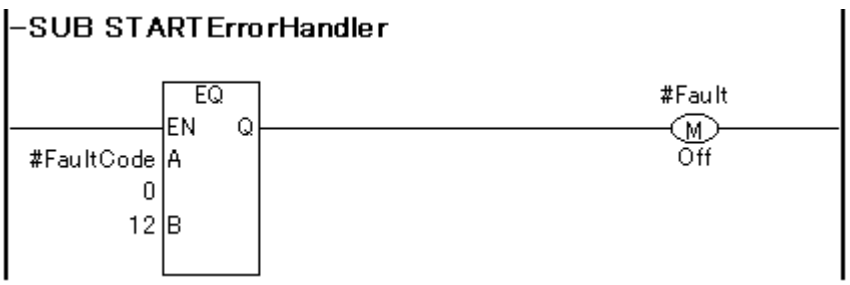
参照→ 3.2.22 #Fault

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。



MEMO 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。

3.2.20 #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

Bit0(=1)	コントローラのSTOP
Bit1(=2)	コントローラのRUN
Bit2(=4)	コントローラのリセット
Bit3(=8)	1スキャン実行
Bit4(=16)	続行
Bit5(=32)	PAUSE
Bit7(=128)	I/O使用可

3.2.21 #DisableAutoStart

#DisableAutoStartは、ディスクリート変数です。

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、GLC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

3.2.22 #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、3.2.19 #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。

3.2.23 #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。"ErrorHandler"サブルーチンについては、 3.2.19 #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

3.2.24 #PercentAlloc

パーセントスキャンモードの場合に使用します。

#PercentAllocで、GLC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→1.1.3 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

3.2.25 #Screen

#Screenに設定された画面番号がコントローラに格納されます。

変数タイプ: 整数

設定: ユーザー

初期値: 0

書き込み可能

MEMO

- #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、GLC2000シリーズのみ対応です。

重要 **#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。**

3.2.26 #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→1.1.3 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

3.2.27 #WatchdogTime

#WatchdogTime で、ウォッチドッグタイマの値を ms 単位で設定します。

#ScanTime がこの値を超えると、メジャー異常が発生します。

#WatchdogTime 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。

変数タイプ: 整数

設定: ユーザー

初期値: 500ms

書き込み可能

第4章 命 令

ここでは、Pro-Control Editorで用いられる命令について説明します。

4.1 命令一覧

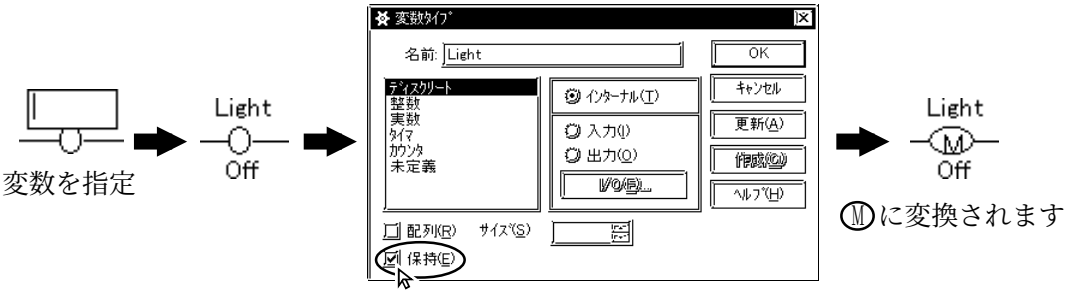
Pro-Control Editorでサポートしている命令を以下に示します。

■ビット操作命令

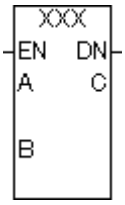
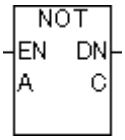
命令	機能	シンボル	処理内容
NO	a接点	⊣ ⊢	論理演算開始（a接点演算開始）
NC	b接点	⊣/⊢	論理演算開始（b接点演算開始）
OUT/M*1	アウト・コイル/ 保持型コイル	-○-/-Ⓜ-	出力/保持型変数への出力
NEG/NM*1	反転コイル/ 保持型反転コイル	-⊘-/-ⓂⓃ-	反転出力/保持型変数への反転出力
SET/SM*1	セット・コイル/ 保持型セット・コイル	-Ⓢ-/-ⓈⓂ-	セット/保持型変数へのセット
RST/RM*1	リセット・コイル/ 保持型リセットコイル	-Ⓡ-/-ⓇⓂ-	リセット/保持型変数へのリセット
PT	立ち上がり接点	⊣P⊢	立ち上がりによる論理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

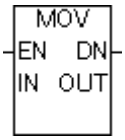
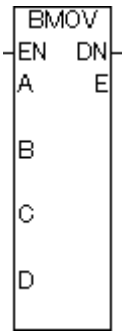
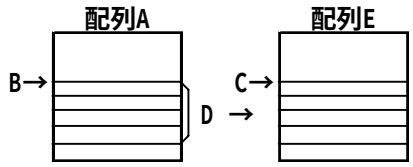
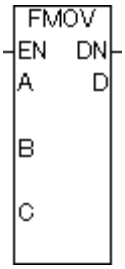
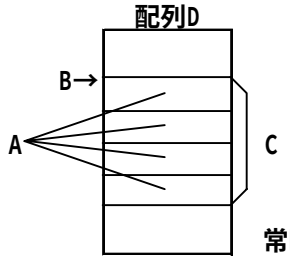
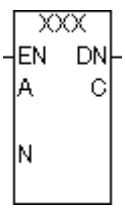
例）下図のようにOUT命令の変数を保持型に指定するとM命令に変換されます。



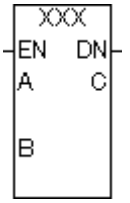
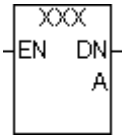
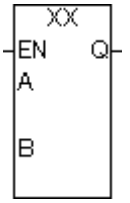
■論理演算命令

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	ビット否定		$\overline{A}$ → C 常時導通

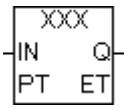
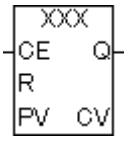
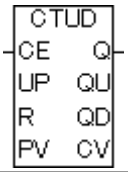
■転送命令

命令	機能	シンボル	処理内容
MOV	転送		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
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

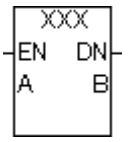
■演算命令

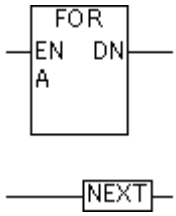
命令	機能	シンボル	処理内容
ADD	加算		A ＋ B → C　常時導通
SUB	減算		A － B → C　常時導通
MUL	乗算		A × B → C　常時導通
DIV	除算		A ÷ B → C　常時導通
MOD	剰余算		A % B → C　常時導通
INC	インクリメント		A ＋ 1 → A　常時導通
DEC	デクリメント		A － 1 → A　常時導通
EQ	比較(=)		A ＝ B　のとき、導通
GT	比較(>)		A ＞ B　のとき、導通
LT	比較(<)		A ＜ B　のとき、導通
GE	比較(>=)		A ≧ B　のとき、導通
LE	比較(<=)		A ≦ B　のとき、導通
NE	比較(<>)		A <> B　のとき、導通

■タイマ命令とカウンタ命令

命令	機能	シンボル	処理内容
TON	オンディレータイマ		参照→4.2.33 TON(オンディレータイマ)
TOF	オフディレータイマ		参照→4.2.34 TOF(オフディレータイマ)
TP	パルスタイマ		参照→4.2.35 TP(パルスタイマ)
CTU	アップカウンタ		参照→4.2.36 CTU(アップカウンタ)
CTD	ダウンカウンタ		参照→4.2.37 CTD(ダウンカウンタ)
CTUD	アップダウンカウンタ		参照→4.2.38 CTUD(アップダウンカウンタ)

■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

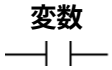
命令	機能	シンボル	処理内容
JMP	ジャンプ	->>ラベル名	ラベルの位置にジャンプ
JSR	ジャンプサブルーチン	->>サブルーチン名<<-	サブルーチンにジャンプ
RET	リターンサブルーチン	-<RETURN>-	呼び出されたJSR命令に戻る
FOR、NEXT	繰り返し		Aで指定された回数、FORとNEXTの間のロジックプログラムを繰り返し実行

MEMO ENCO、DECO、FOR、NEXTはGLC2000シリーズのみ対応です。

4.2 命令詳細

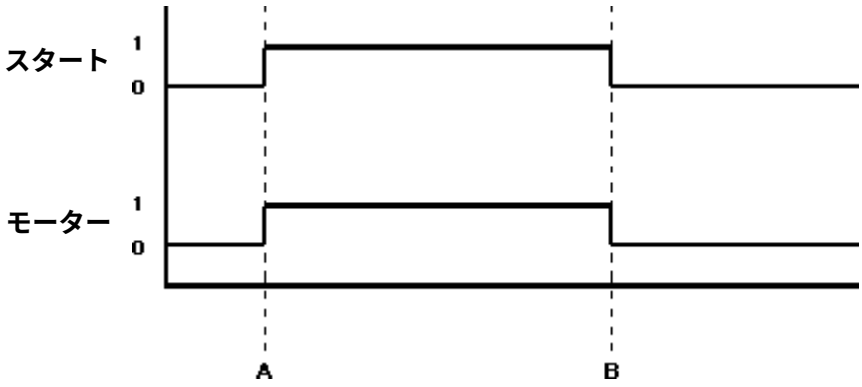
ここでは各命令の詳細を説明します。

4.2.1 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。

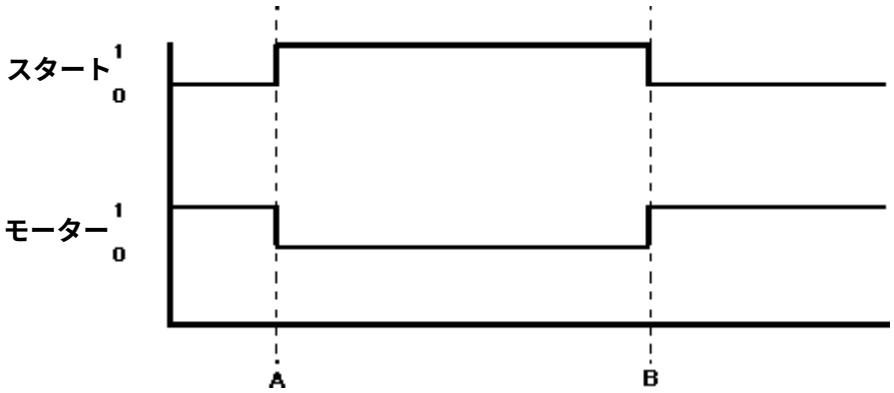
B　変数**スタート**が OFF になると、変数**モーター**が OFF になります。

4.2.2 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。

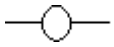


A　変数**スタート**が ON になると、変数**モーター**が OFF になります。

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。

4.2.3 OUT/M(アウト・コイル)

変数

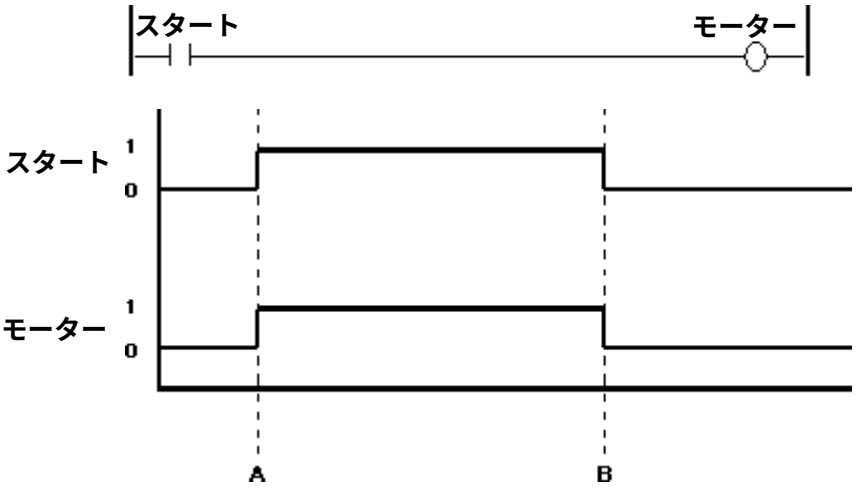


OUT 命令は、I/O に割り付けられた変数の ON/OFF や、内部メモリのディスクリート変数の ON/OFF に使用します。

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、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。

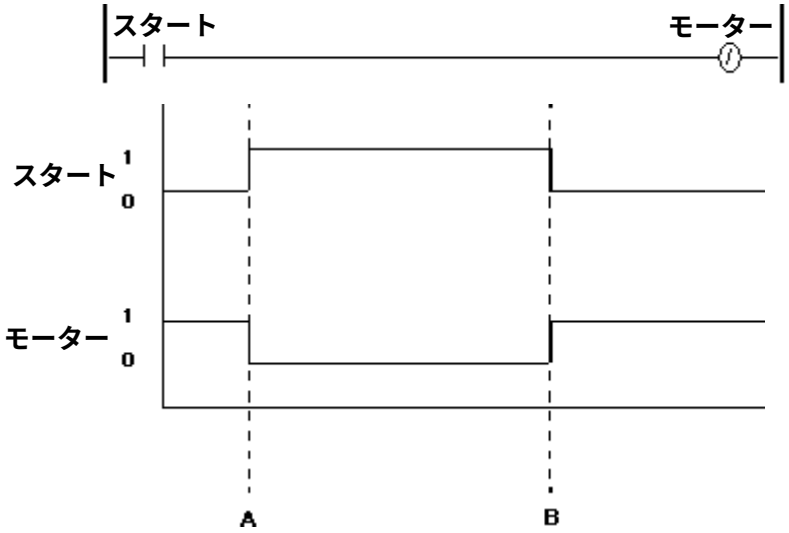
4.2.4 NEG(反転コイル)

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、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。(保持型の NEG 命令は GLC100 では対応していません。)

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。

MEMO NEG 命令は非保持型変数のみ使用できます。

4.2.5 SET(セット・コイル)

変数

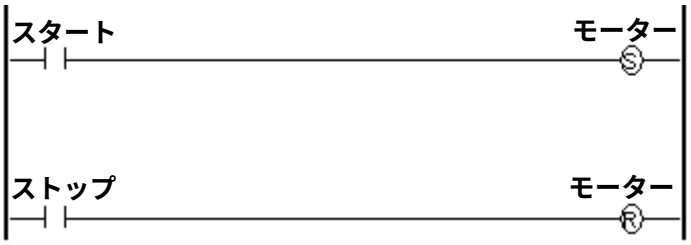
コイルに導通してからSET命令を実行すると、変数がONになります。

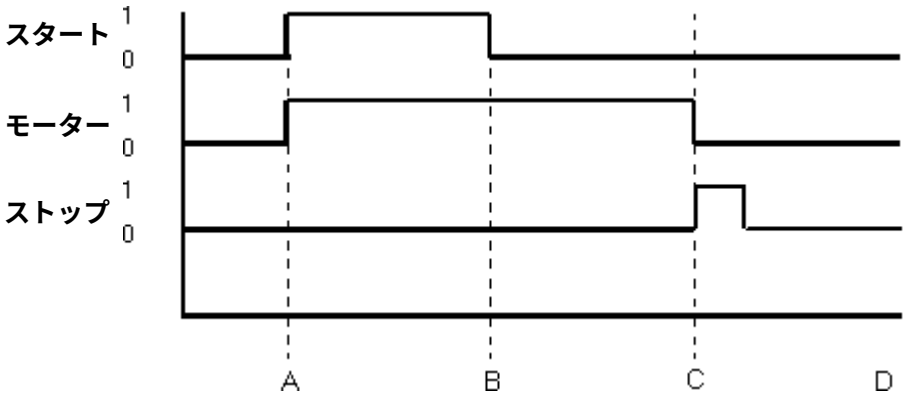
変数は、RST命令のような他の命令で確実にOFFにされるまで、ONのままです。

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、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。

4.2.6 RST(リセット・コイル)

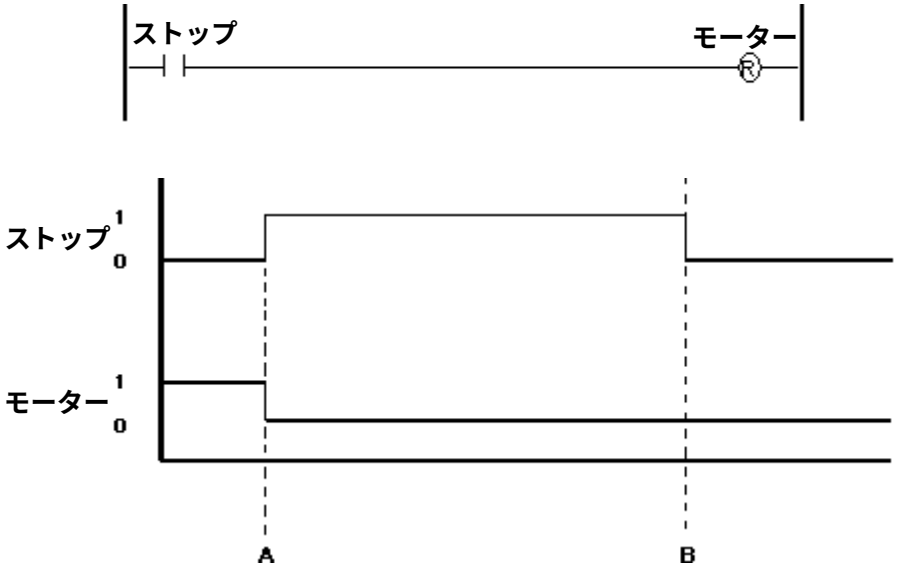
RST 命令を実行して、コイルに導通すると、変数が OFF になります。

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、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。

4.2.7 PT(立ち上がり接点)

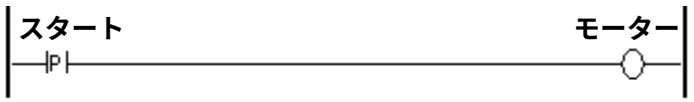
変数

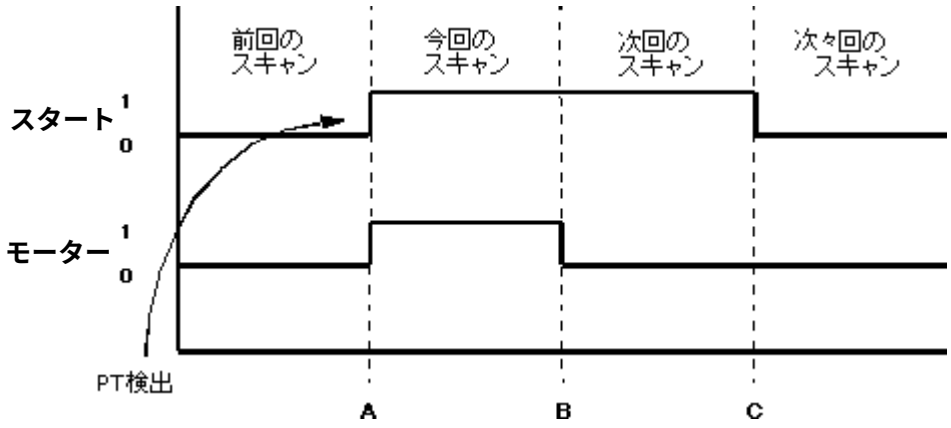
—|P|—

PT 命令を実行すると、前回のスキャンで OFF の変数を、今回のスキャンで ON にしたときに、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。





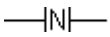
A 変数**スタート**が ON になり、変数**モーター**が ON になります。

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**が今スキャンで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。

4.2.8 NT(立ち下がり接点)

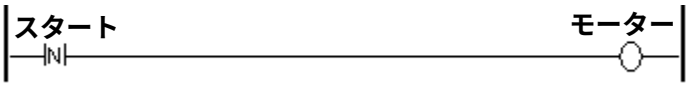
変数

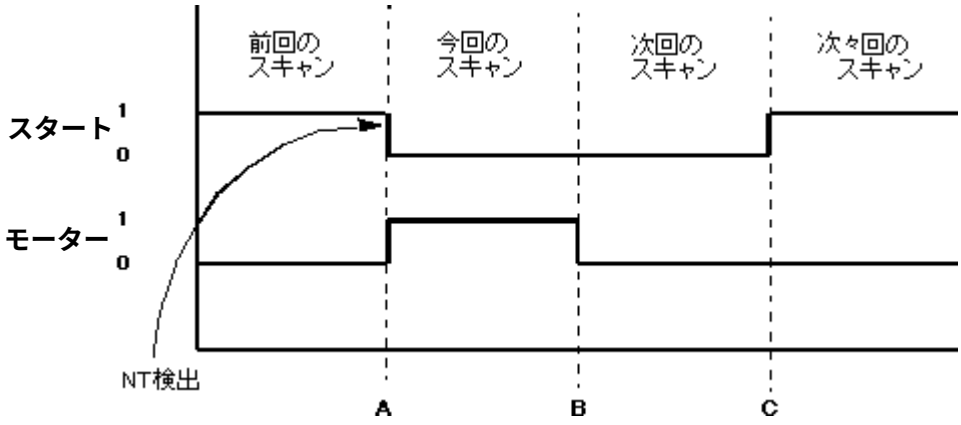


NT 命令を実行すると、前回のスキャンで ON の変数を今回のスキャンで OFF にしたときに、導通します。

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。



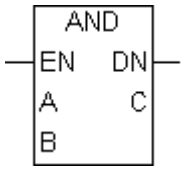


A　変数**スタート**が OFF になり、変数**モーター**が ON になります。

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**が今スキャンで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。

4.2.9 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。

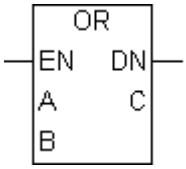
3.3 つの変数が同じサイズの配列のときは、配列 A と配列 B の論理積が演算されます。結果は配列 C に格納されます。

AND 命令は常に導通します。

AND命令が実行できる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	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

4.2.10 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。

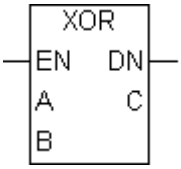
1. **A、B の変数が両方とも整数のときは、32 ビットの単純な論理和が演算されます。**
2. **A と C が配列で、B が整数配列でないとき、A のそれぞれの要素と B を論理和演算し、結果は、C の対応するそれぞれの要素に格納されます。A と C の配列は同じサイズにしてください。**
3. **3 つの変数が同じサイズの配列のときは、配列 A と配列 B の論理和が演算されます。結果は、配列 C に格納されます。**

OR 命令は、常に導通します。

OR命令が実行できる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	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

4.2.11 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。

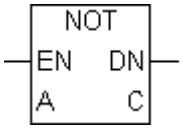
1. **A、B の変数が両方とも整数のときは、32 ビットの単純な排他的論理和が演算されます。**
2. **A と C が配列で、B が整数配列でないとき、A のそれぞれの要素と B を排他的論理和演算し、結果は、C の対応するそれぞれの要素に格納されます。A と C の配列は同じサイズにしてください。**
3. **3 つの変数が同じサイズの配列のときは、配列 A と配列 B の排他的論理和が演算されます。結果は、配列 C に格納されます。**

XOR 命令は、常に導通します。

XOR命令が実行できる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	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

4.2.12 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。

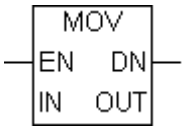
1. **A の変数が整数のときは、32 ビットの単純なビット反転が演算されます。**
2. **A の変数が配列のときは、A の配列全体のビット反転が演算されます。結果は、C に格納されます。このとき、A と C は同じサイズにしてください。**

NOT 命令は、常に導通します。

NOT命令が実行できる変数タイプ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

4.2.13 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 命令に有効な変数タイプの組み合わせは以下の通りです。

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。

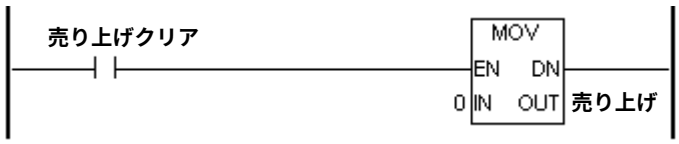
以下の例では、MOV 命令の使用例について説明しています。

1番目の例では、変数をクリアします。

2番目の例では、配列をブロック転送します。

例 1: クリアする

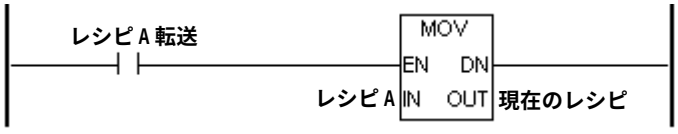
変数をクリアするときは、その変数に 0を転送すると簡単です。

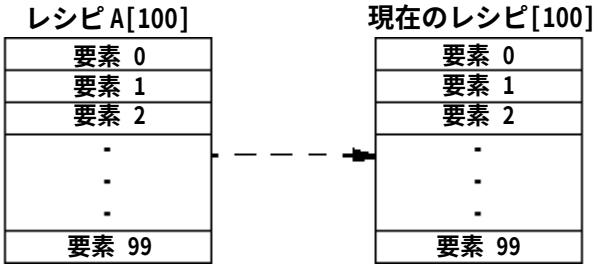


例 2: ブロック転送をする

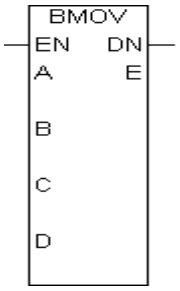
ブロック転送をするときは、同じタイプの 2つの配列を指定して転送すると簡単です。

例えば、配列**レシピ A**を、同じタイプの配列**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。





4.2.14 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:データ数
E:転送先配列変数名

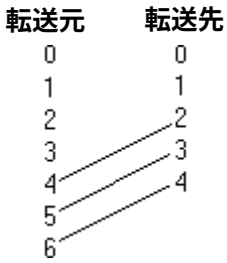
BMOV 命令を実行すると、ある一つの配列の要素のいくつかを、別の配列の要素のいくつかにコピーすることができます。すなわち、配列 A[B]から始まる D 個の要素が、配列 E[C]から始まる D 個の要素にコピーされます。

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 命令が有効な変数タイプの組み合わせは以下の通りです。

A、Eのタイプ	B、C、Dのタイプ
整数配列	整数
	0
	整数定数

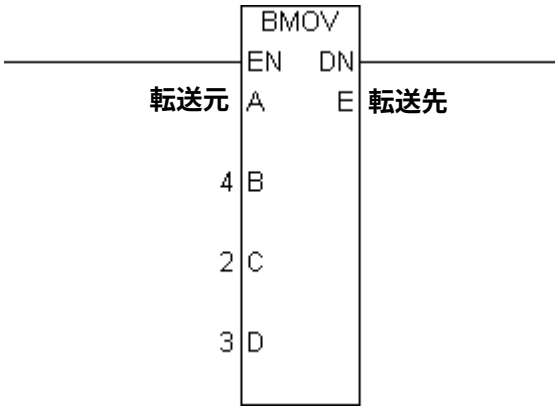
◆動作例

要素数 7 の整数配列**転送元**の一部を要素数 6 の整数配列**転送先**にコピーするときは、以下のようになります。



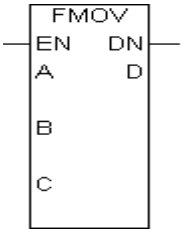
転送元[4]は、**転送先[2]**にコピーされます。
転送元[5]は、**転送先[3]**にコピーされます。
転送元[6]は、**転送先[4]**にコピーされます。

BMOV 命令を、このように定義してください。



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。

4.2.15 FMOV(フィル転送)



A:データ
B:配列D[B]より開始
C:データ数
D:転送先変数名

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。

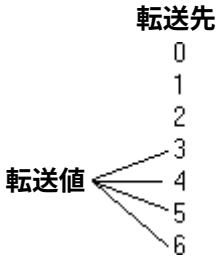
整数配列に対してのみ、有効です。この命令は常に導通します。

FMOV命令が有効な変数タイプの組み合わせは以下の通りです。

A、B、Cのタイプ	Dのタイプ
整数	整数配列
0	
整数定数	

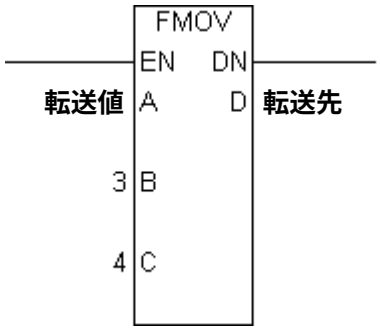
◆動作例

要素数7の整数配列'**転送先**'のいくつかの要素に初期値'**転送値**'をコピーするときは、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。

FMOV命令を、このように定義してください。

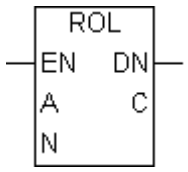


コントローラはRUN中にFMOV命令で配列AやEの存在しない要素を参照していないかどうかをチェックします。無効な配列の参照が行われた場合、メジャー異常が発生し、

#Faultcodeに2が設定されます。

参照→3.2.16 #Faultcode

4.2.16 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。

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。**

この命令は常に導通します。

ROL命令が有効な変数タイプの組み合わせは以下の通りです。

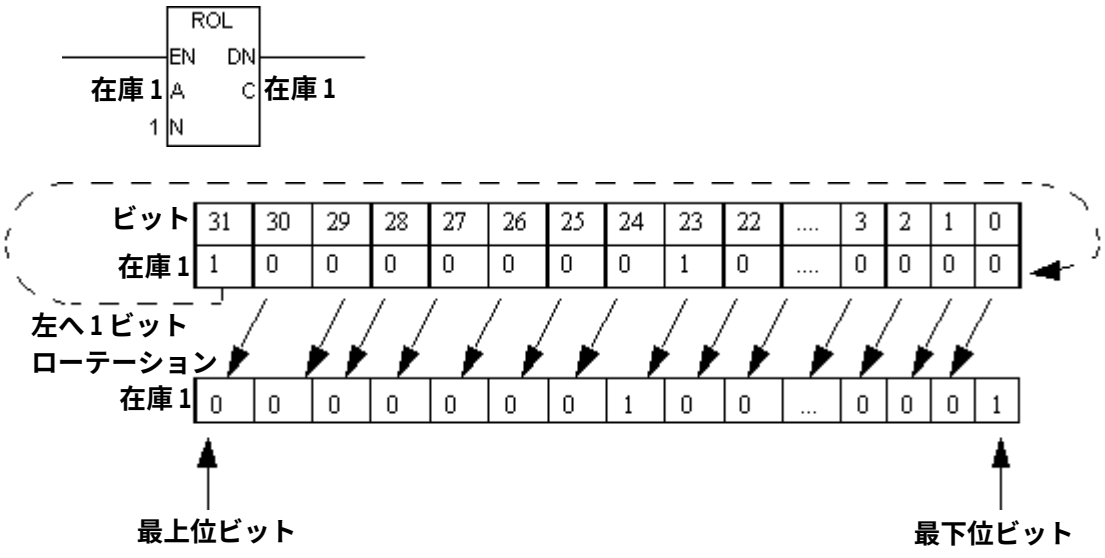
Aのタイプ	Nのタイプ	Cのタイプ
整数	整数または整数定数	整数
整数配列	整数または整数定数	Aと同じサイズの配列
整数定数	整数または整数定数	整数

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。

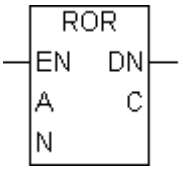
参照→ 3.2.19 #Overflow

◆動作例

以下の例では、整数**在庫1**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。



4.2.17 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。

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。**

この命令は常に導通します。

ROR命令が有効な変数タイプの組み合わせは以下の通りです。

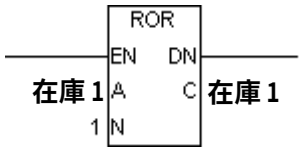
Aのタイプ	Nのタイプ	Cのタイプ
整数	整数または整数定数	整数
整数配列	整数または整数定数	Aと同じサイズの配列
整数定数	整数または整数定数	整数

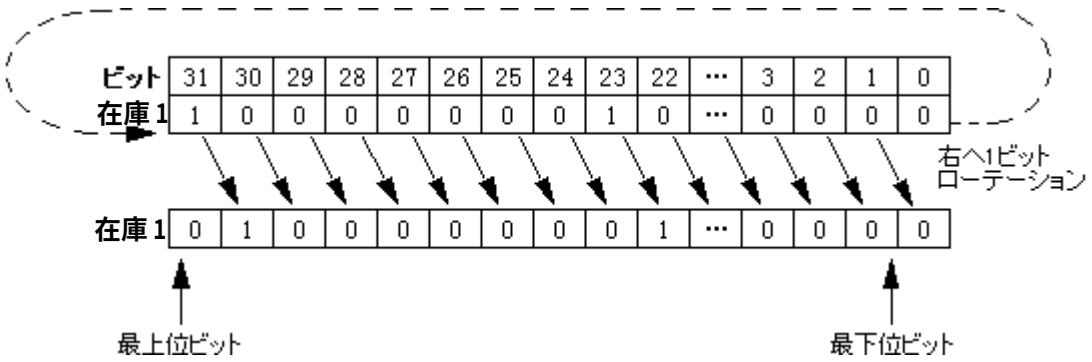
MEMO N が範囲外のときは、#Overflow が ON になります。このとき、結果は不定です。

参照→ 3.2.19 #Overflow

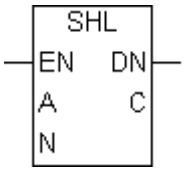
◆動作例

以下の例では、整数**在庫 1** 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。





4.2.18 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。

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。**

この命令は常に導通します。

SHL命令が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	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

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→ 3.2.19 #Overflow

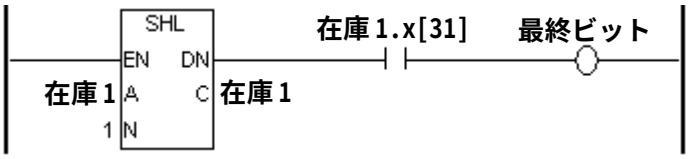
◆動作例

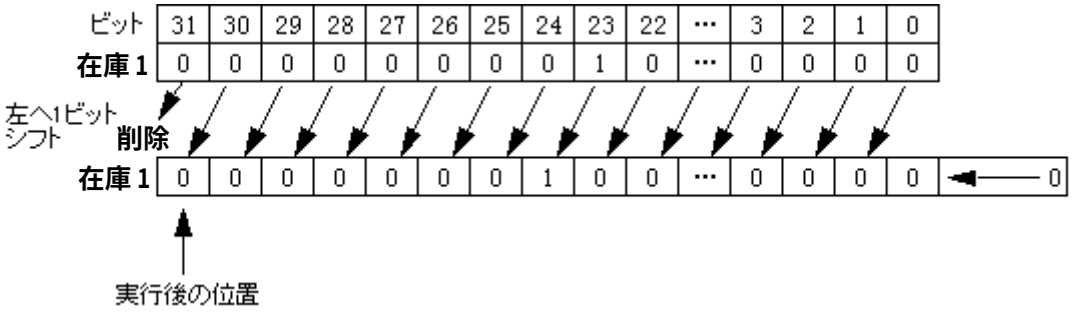
以下の例では、1ビット左シフトを実行してビットの位置を把握する方法について説明しています。

整数**在庫1**のそれぞれのビットは、実際の位置を表します。

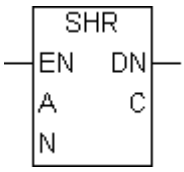
プログラムのスキャンのたびに（または一定時間ごとに)、ビットは次の位置へ左シフトします。

ビットが変数の最終ビット位置（31）にくると、変数**最終ビット**が ON になり、動作が終了したことがわかります。


SHL
EN
DN
在庫1
A
C
在庫1
1
N
在庫1.x[31]
最終ビット


ビット
31 30 29 28 27 26 25 24 23 22 … 3 2 1 0
在庫1
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
在庫1
0 0 0 0 0 0 0 1 0 0 … 0 0 0 0
0
実行後の位置

4.2.19 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。

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。**

この命令は常に導通します。

SHR命令が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	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

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→ 3.2.19 #Overflow

◆動作例

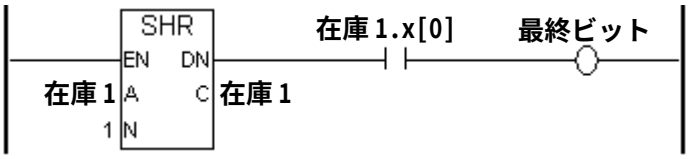
・ビットを扱う場合

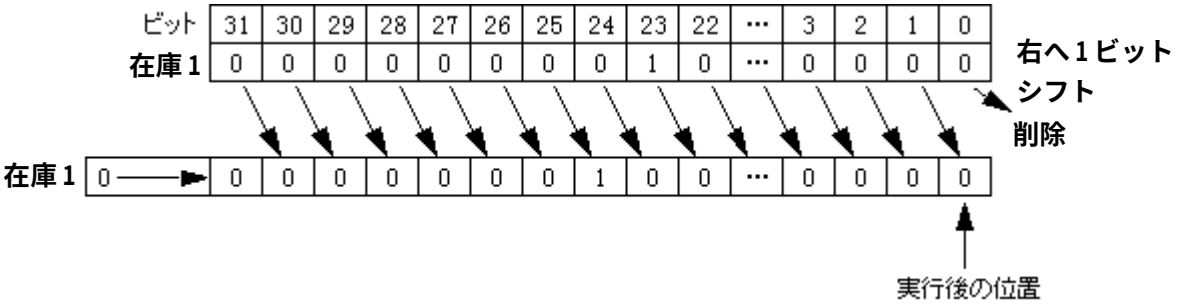
1ビット右シフトを実行してビットの位置を把握する方法について説明しています。

整数**在庫1**のそれぞれのビットは、実際の位置を表します。

プログラムのスキャンのたびに（または一定時間ごとに）、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、変数**最終ビット**が ON になり、動作が終了したことがわかります。





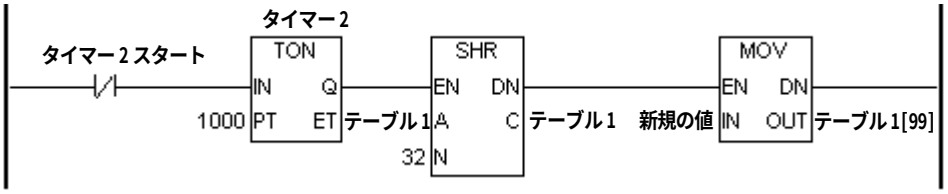
・配列を扱う場合

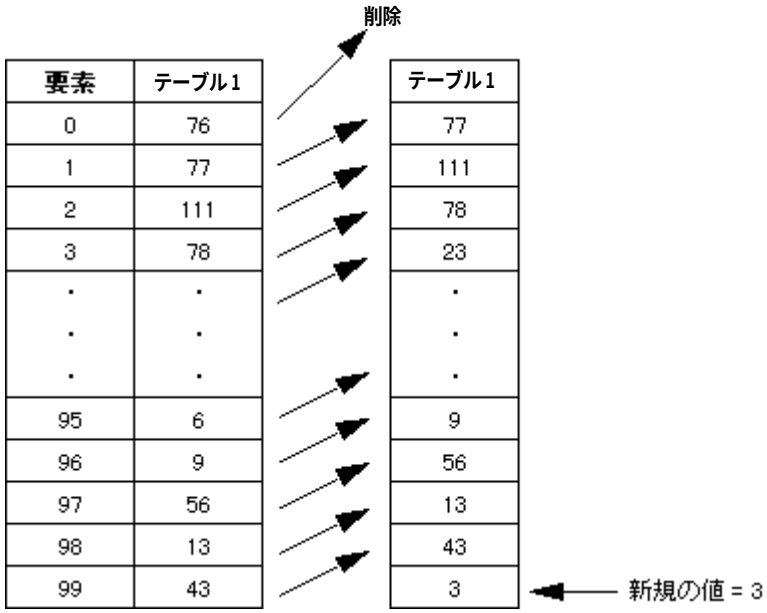
SHR命令を使用して整数配列の中で各要素の値を移動することについて説明しています。

32 ビットシフトでは、32 ビット整数全体がシフトします。

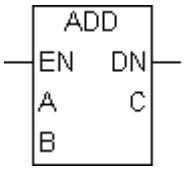
１秒ごとに、整数配列**テーブル 1** の整数値が要素 0 の方向に 1 つずつ移動します。

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。





4.2.20 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。

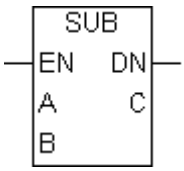
Aのタイプ	Bのタイプ	Cのタイプ
整数	整数	整数または実数
整数定数	整数定数	整数または実数
実数	実数	整数または実数
実数定数	実数定数	整数または実数

MEMO

・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→ 3.2.19 #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。

4.2.21 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。

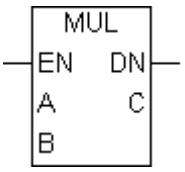
Aのタイプ	Bのタイプ	Cのタイプ
整数	整数	整数または実数
整数定数	整数定数	整数または実数
実数	実数	整数または実数
実数定数	実数定数	整数または実数

MEMO

・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→ 3.2.19 #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。

4.2.22 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。

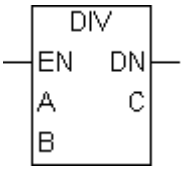
Aのタイプ	Bのタイプ	Cのタイプ
整数	整数	整数または実数
整数定数	整数定数	整数または実数
実数	実数	整数または実数
実数定数	実数定数	整数または実数

MEMO

・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→ 3.2.19 #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。

4.2.23 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。

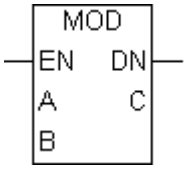
Aのタイプ	Bのタイプ	Cのタイプ
整数	整数	整数または実数
整数定数	整数定数	整数または実数
実数	実数	整数または実数
実数定数	実数定数	整数または実数

MEMO

・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→ 3.2.19 #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。

4.2.24 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。

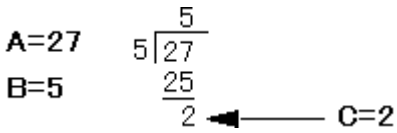
この命令は常に導通します。MOD 命令が有効な変数タイプの組み合わせは以下の通りです。

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

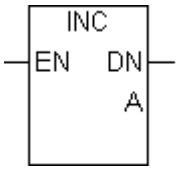
0 で除算したときは、#0verflow が ON になり、剰余算の結果は不定です。

参照→ 3.2.19 #0verflow

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。



4.2.25 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。

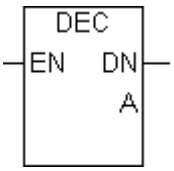
INC命令が有効な変数タイプの組み合わせは以下の通りです。

Aのタイプ
整数

A が 0x7FFFFFFF のとき 0x80000000 となり、#0verflow がセットされます。

参照→ 3.2.19 #0verflow

4.2.26 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。

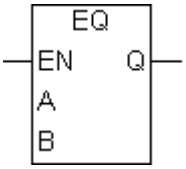
DEC命令が有効な変数タイプは以下の通りです。

Aのタイプ
整数

A が 0x80000000 のとき 0x7FFFFFFF となり、 #Overflow がセットされます。

参照→ 3.2.19 #Overflow

4.2.27 EQ(比較：=)



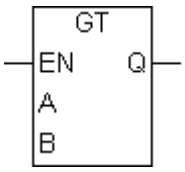
A:データ
B:データ

A = B のときに導通します。EQ 命令が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。

4.2.28 GT(比較：>)



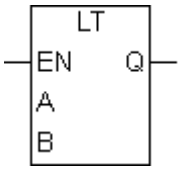
A:データ
B:データ

A > Bのときに導通します。GT命令が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より大きいです。

4.2.29 LT(比較：<)



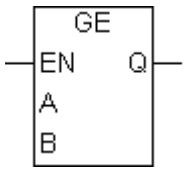
A:データ
B:データ

A < Bのときに導通します。LT命令が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。

4.2.30 GE(比較：>=)



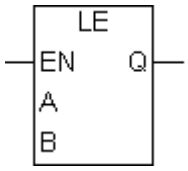
A:データ
B:データ

A ≧ Bのときに導通します。GE命令が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。

4.2.31 LE(比較：<=)



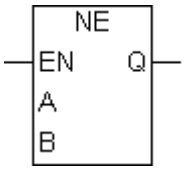
A:データ
B:データ

A ≦ Bのときに導通します。LE命令が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。

4.2.32 NE(比較：<>)



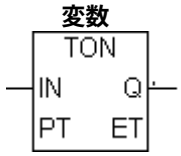
A:データ
B:データ

A ≠ Bのときに導通します。NE命令が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。

4.2.33 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
PT	設定値	整数
ET	現在値	整数
Q	タイマ出力ビット	ディスクリート
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

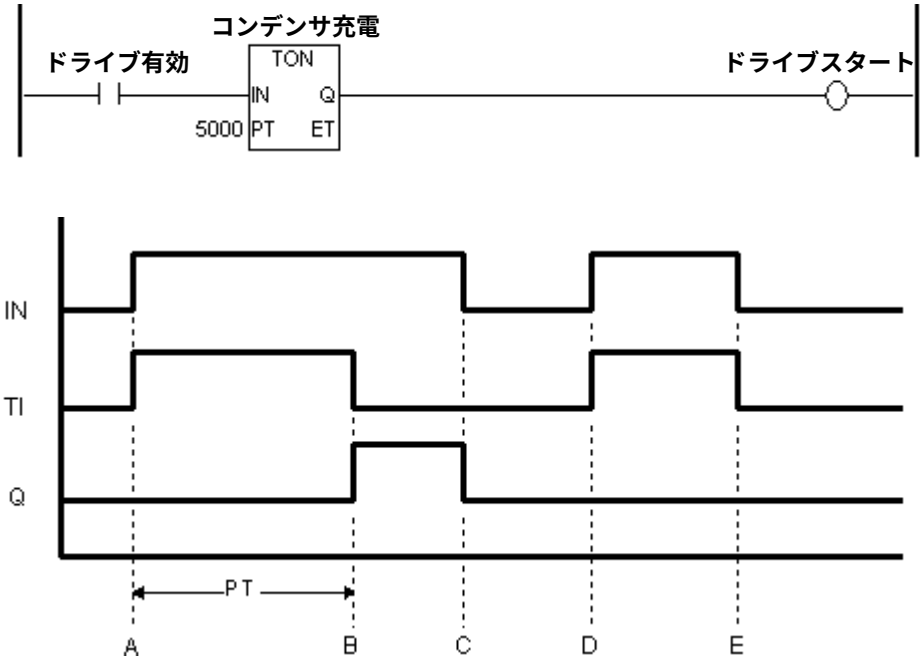
・経過時間**変数 .ET**は0にリセットされます。

・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、電源を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。

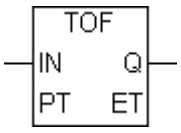
C　タイマ入力ビットINがOFFになり、タイマ出力ビットQはOFFになります。経過時間ETが0にリセットされます。

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。

4.2.34 TOF(オフディレータイマ)

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

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
PT	設定値	整数
ET	現在値	整数
Q	タイマ出力ビット	ディスクリート
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

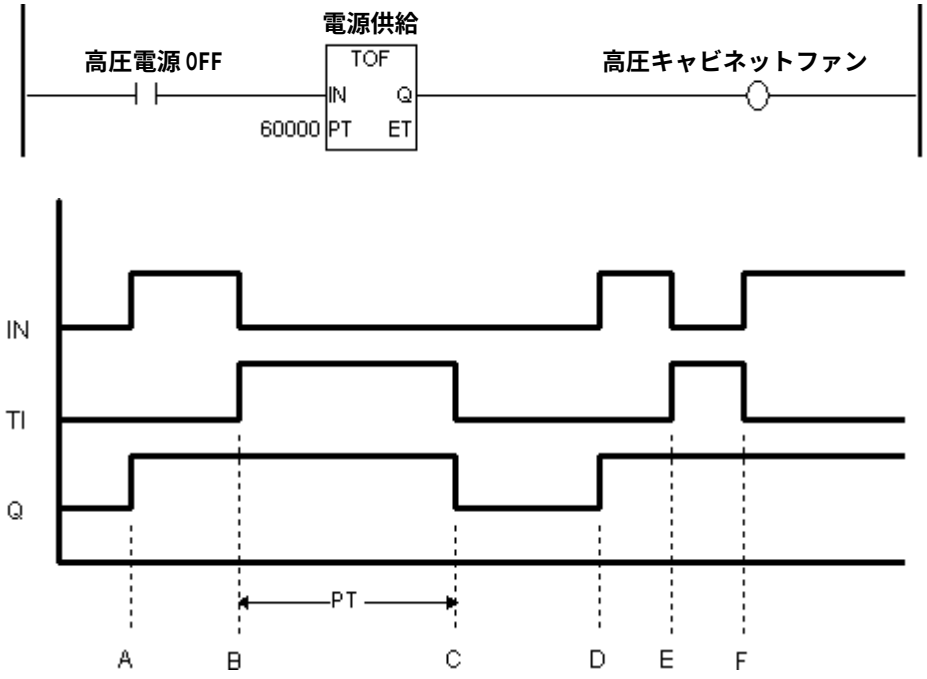
・経過時間**変数 .ET** は、設定値を保持します。

・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）。

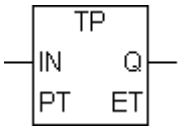
D　タイマ入力ビット IN が ON になります。タイマ計測ビット TI は OFF のままです。タイマ出力ビット Q は ON になります。経過時間 ET は 0 にリセットされます。

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。

4.2.35 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
PT	設定値	整数
ET	現在値	整数
Q	タイマ出力ビット	ディスクリート
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。

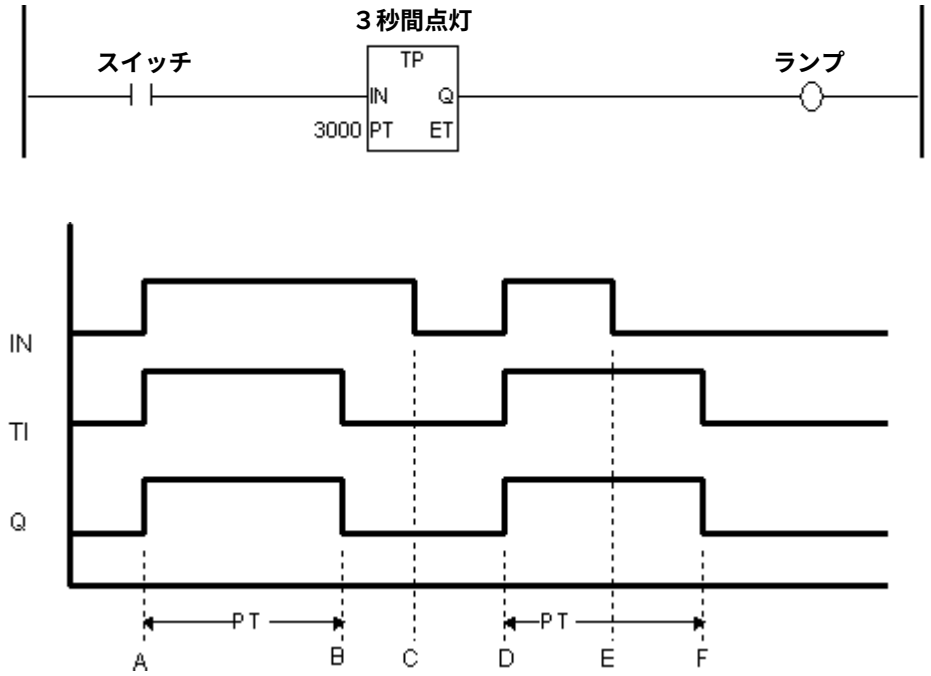
・タイマ出力ビット**変数 .Q** が OFF になり、導通を終了します。

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）。

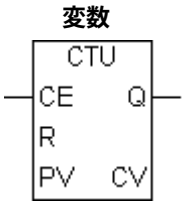
C　タイマ入力ビット IN が OFF になります。経過時間 ET は 0 にリセットされます。

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。

4.2.36 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
PV	設定値	整数
CV	現在値	整数
R	カウンタリセット	ディスクリート
UP	アップカウンタ	ディスクリート
QU	アップカウンタ出力	ディスクリート
QD	ダウンカウンタ出力	ディスクリート
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。

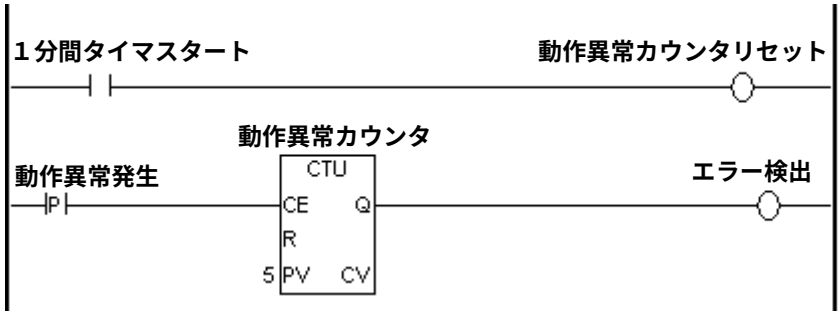
現在値**変数.CV**が設定値**変数.PV**と等しくなると、カウンタ出力ビット**変数.Q**がONになり、導通します。

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。

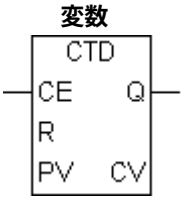
◆動作例

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。

4.2.37 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
PV	設定値	整数
CV	現在値	整数
R	カウンタリセット	ディスクリート
UP	アップカウンタ	ディスクリート
QU	アップカウンタ出力	ディスクリート
QD	ダウンカウンタ出力	ディスクリート
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。

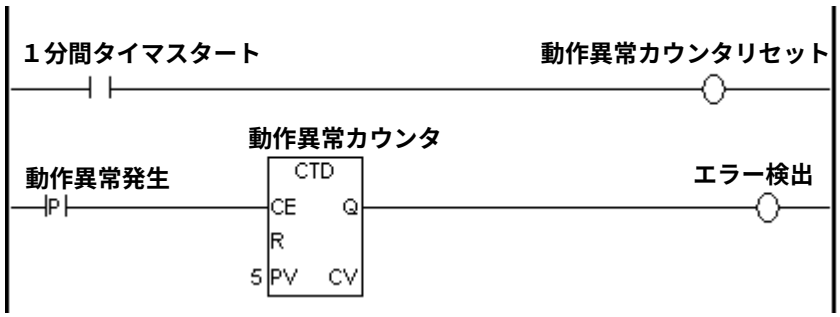
現在値**変数.CV**が1減算され0以下になると、カウンタ出力ビット**変数.Q**はONになり、導通します。

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

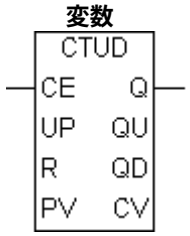
以下の例では、1分間に**動作異常**を5つカウントすると、エラーを知らせます。

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。

4.2.38 CTUD(アップダウンカウンタ)



CE:カウンタ起動用ビット
UP:カウンタアップフラグ
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(以上)
QD カウンタアップフラグ(以下)
CV:カウンタ現在値

◆動作概要

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

CTUD 命令は、カウンタアップ有効フラグ**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。

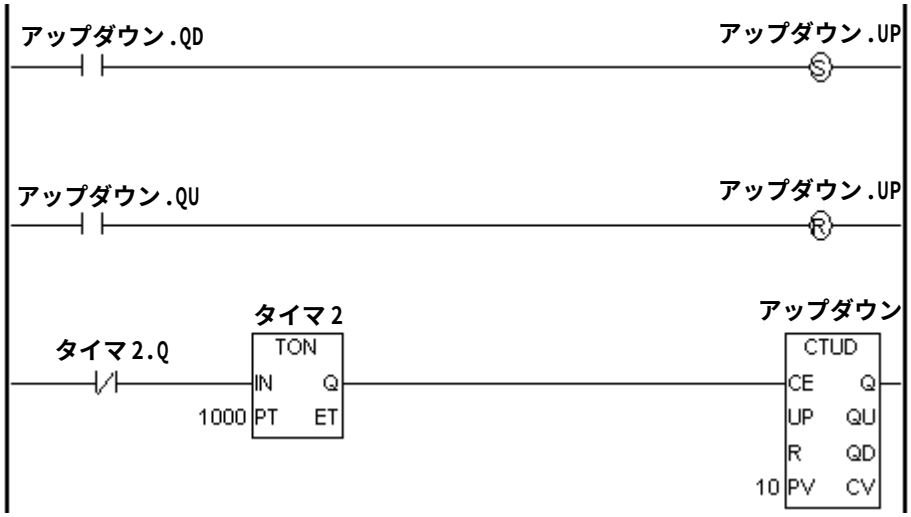
実行後に現在値**変数 .CV** が 0 以下になったときは、**変数 .Q** と**変数 .QD** が ON になります。

◆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。

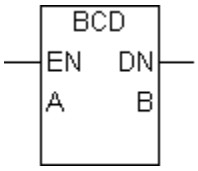
カウンタ**アップダウン**が 0 になると UP ビットが ON になり、カウンタ**アップダウン**が 10（プリセット値）になると UP ビットが OFF になります。



MEMO

カウンタアップ有効フラグ**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。カウンタアップ有効フラグ**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。

4.2.39 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。

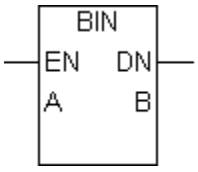
Aのタイプ	Bのタイプ
整数	整数
0	
整数定数	

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→3.2.16 #Faultcode、3.2.19 #Overflow

MEMO 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。

4.2.40 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。

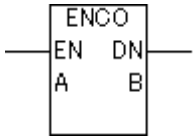
Aのタイプ	Bのタイプ
整数	整数
0	
整数定数	

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→3.2.16 #Faultcode、3.2.19 #Overflow

MEMO 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。

4.2.41 ENCO(エンコード)



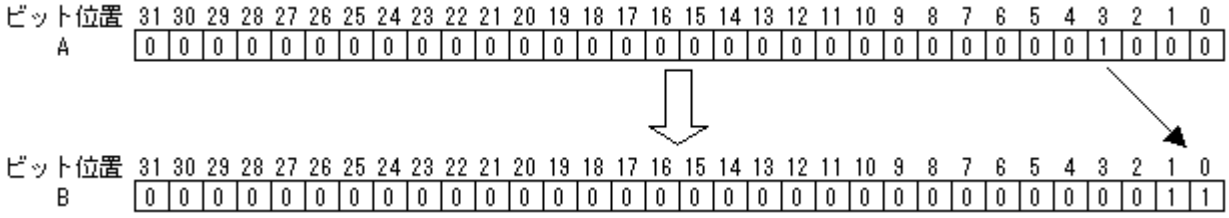
A:データ
B:結果格納先

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

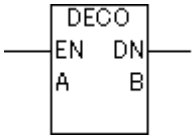
例：Aに0x00000008を入力した場合、出力Bは0x00000003になります。



MEMO

・入力Aに0が入っているとマイナー異常#OverFlowにし、#FaultCodeにエラーコード ”13”をセットします。**参照→3.2.19 #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)
・ENCO命令はGLC2000シリーズのみ対応です。

4.2.42 DECO(デコード)



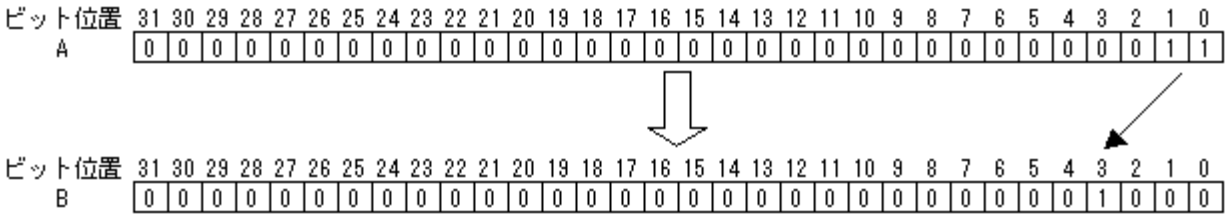
A:データ
B:結果格納先

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。



MEMO

・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→3.2.19 #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)
・DECO命令はGLC2000シリーズのみ対応です。

4.2.43 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。

定期的に END ラングまで実行し、ウォッチドッグタイマをリセットしてください。

JMP 命令は、ラングの最後に置いてください。

4.2.44 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。1 回のサブルーチンジャンプでスタックを 1 使用します。FOR ～ NEXT 命令を使用する場合は、スタックサイズを合計してください。

・ロジックプログラムで使用できるスタック数は128です。
スタックを使用する命令は FOR ～ NEXT 命令と JSR 命令のみです。
参照→4.2.46 FOR ～ NEXT 命令

4.2.45 RET(リターンサブルーチン)

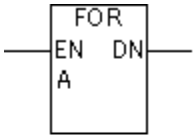
—<RETURN>—

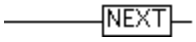
RET 命令に導通すると、サブルーチンから強制的に呼び出し元の JSR 命令に戻り、次のラングから命令の実行が続けられます。

RET 命令は必ずしも使用する必要はありません。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からです。

RET 命令は、ラングの最後に置いてください。

4.2.46 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。JSR命令を使用する場合は、スタックサイズを合計してください。

MEMO

・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**「Pro-Control Editorオペレーションマニュアル 付録A:エラーと警告」**を参照してください。

・＃FaultCodeのエラーコードについては **3.2.25 ＃FaultCode** を参照してください。

・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→3.2.27 #WatchdogTime

・ロジックプログラムで使用できるスタック数は128です。スタックを使用する命令はFOR～NEXT命令とJSR命令のみです。

参照→4.2.44 JSR命令

・FOR～NEXT命令はGLC2000シリーズのみ対応です。

第5章 LSエリアリフレッシュ

5.1 LSエリアリフレッシュの概要

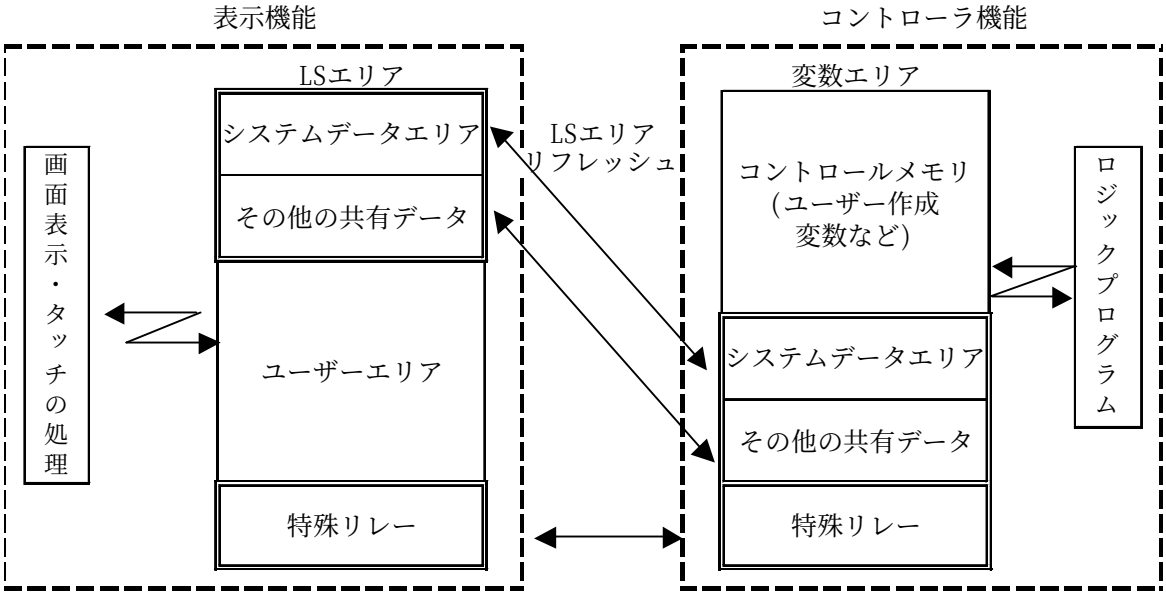
■LSエリアリフレッシュ機能

GLCでは、画面切り替えや内部ブザーなどを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ラ機能上で画面切り替えや内部ブザー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ラ機能間で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ラ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。



5.2 LSエリアリフレッシュの設定

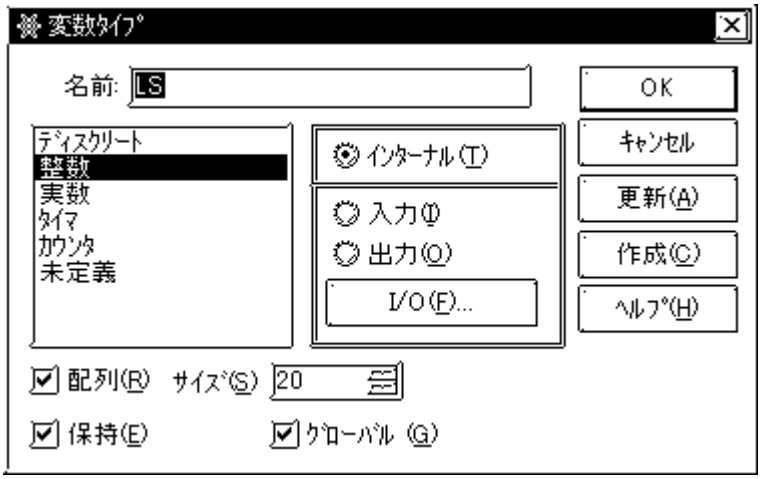
ロジックプログラムにてLSエリアを指定するには、Pro-Control Editorにて変数を登録する必要があります。ここではPro-Control Editorにて登録する方法を説明します。

■登録方法

Pro-Control 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。**

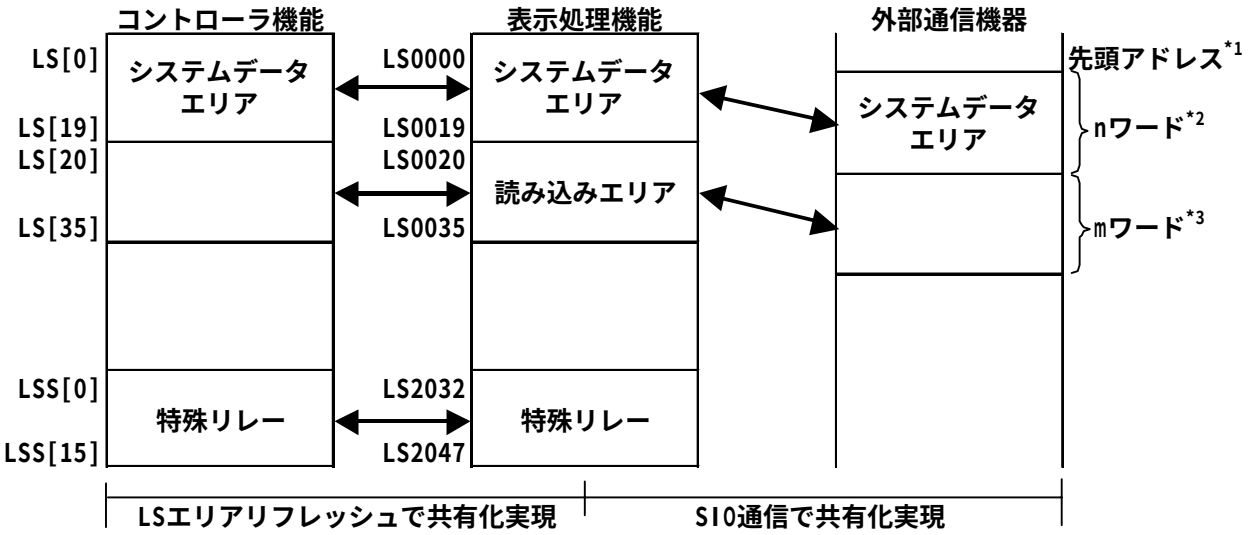
変数とアドレスとの関係は下表に示した通りです。

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	
・ ・	・ ・	・ ・	
LS[275]	275	LS0275	その他の共有データ
LSS[0]	2032	LS2032	特殊リレー
LSS[1]	2033	LS2033	
・ ・	・ ・	・ ・	
LSS[15]	2047	LS2047	

「LSエリア」、「特殊リレー」の詳細については機器接続マニュアル(PLC接続マニュアル)を参照してください。**※1　変数名：GLCのロジックプログラムで扱うシステム変数**

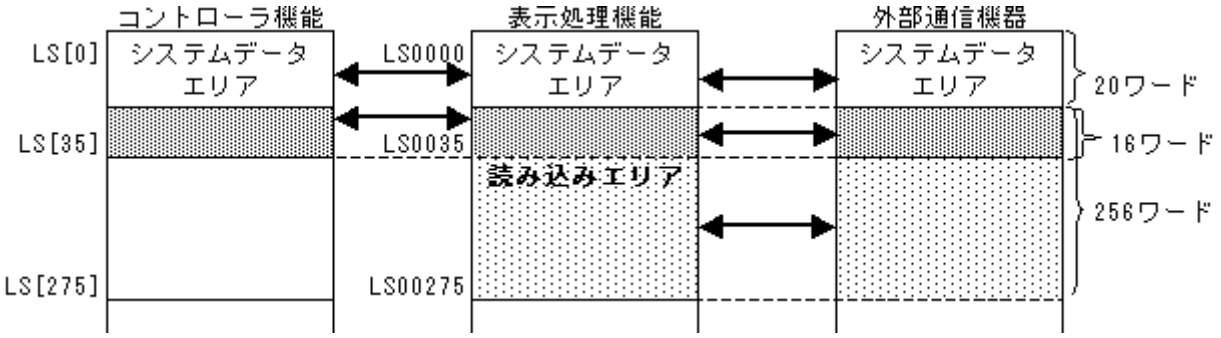
5.3 GLC と外部通信機器のデータ共有について

コントロール機能で外部通信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外部通信機器のデータレジスタのデータ共有が16ワードを超えた場合、画面表示機能が著しく低下する場合があります。

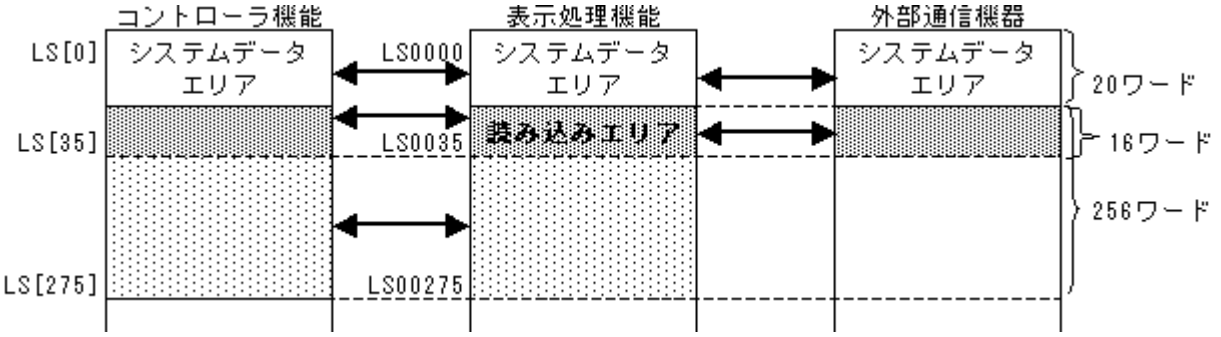


読み込みエリアまたは変数LSを16ワードを超えて設定したい場合、読み込みエリアは256ワード、変数LSのサイズは276ワードまで設定可能です。コントローラ機能、表示処理機能、外部通信機器で共有するデータは16ワードまでに設定することを推奨いたします。

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



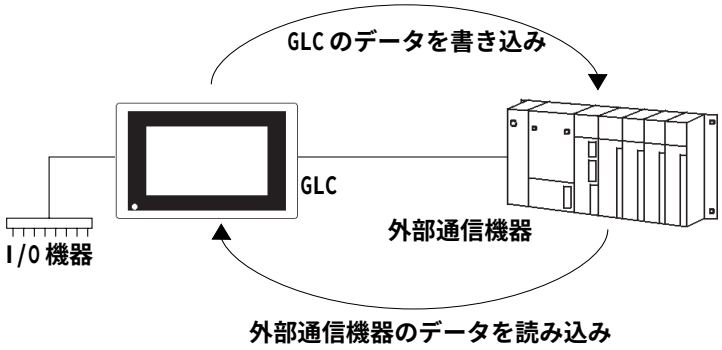
※1 GLCの初期設定で指定したシステム先頭アドレスのことです。

※2 n=0～20　GLCの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　GLCの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
- **コントロール機能のロジックプログラムと表示処理機能のタグ、外部通信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。**
- **GLC 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、タグ設定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 GLC と外部通信機器のデータ共有を行うと、GLC を外部通信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。



5.3.1 GLC と外部通信機器のデータ共有時の注意点

GLC と外部通信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外部通信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、「外部通信機器との通信異常」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6章 I/Oドライバ

ここでは、GLCにてI/Oユニット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。

6.1 I/Oドライバについて

Pro-Control Editorでは、外部入出力を扱う場合、GLCに装着するI/Oユニットとそれに対応したI/Oドライバが必要になります。I/Oドライバの選択・設定方法は、**「Pro-Control Editor オペレーションマニュアル 2.11.I/Oの割り付け」**を参照してください。

対応ドライバは以下の通りです。

機種	対応ドライバ
GLC100シリーズ	DIOドライバ ユニワイヤI/Fドライバ Flex Networkドライバ
GLC300シリーズ	
GLC2300シリーズ	Flex Networkドライバ
GLC2400シリーズ	
GLC2600シリーズ	

MEMO.

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→3.2.18 #IOFault、3.2.20 #Command

6.2 Flex Network ドライバ

GLC 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 Pro-Control 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。また、GLC100、GLC300 の場合は Flex Network I/F ユニットが装着されていることも確認してください。GLC2300、GLC2400、GLC2600 は Flex Network I/F が内蔵されています。

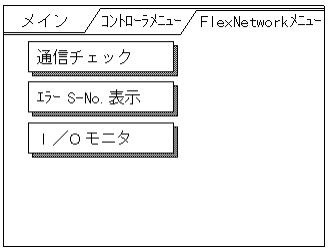
オフラインモードに移る方法は、参照→「**各 GLC シリーズユーザーズマニュアル」(別売)**。

6.2.1 Flex Network I/F ユニットの自己診断

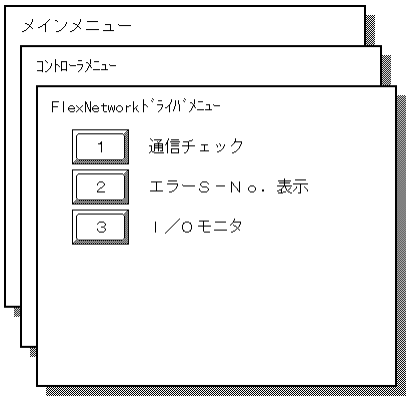
ここでは Flex Network ユニットの自己診断の方法について説明します。GLC 本体の自己診断については、参照→「**GLC シリーズユーザーズマニュアル」(別売)**。
コントローラメニューで[Flex Network ドライバ]を選択すると、下の画面が表示されます。

＜通信チェックを選択する場合＞

GLC100/GLC2300 シリーズ

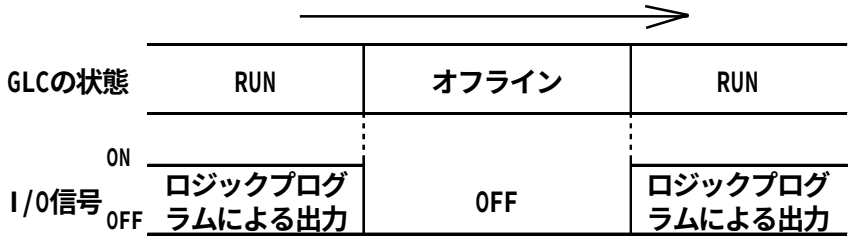


GLC300/GLC2400/GLC2600 シリーズ



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 GLC 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。

Flex Network I/Fユニットに接続されている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。

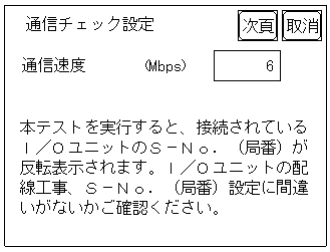
- 接続されているI/Oユニットの確認
- 故障しているI/Oユニット（通信部）の確認

以下に通信チェックの手順を示します。

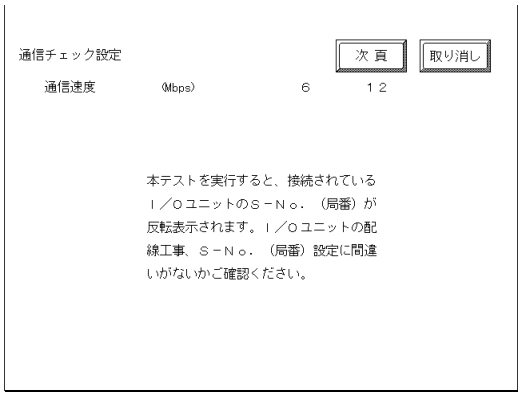
[通信チェック]を押すと以下の**［通信チェック設定］画面**が表示されます。

[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。

GLC100/GLC2300シリーズ



GLC300/GLC2400/GLC2600シリーズ

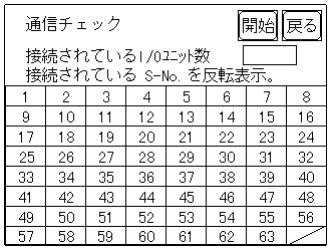


[次頁]ボタンを押すと、以下の**［通信チェック］画面**に切り替わります。

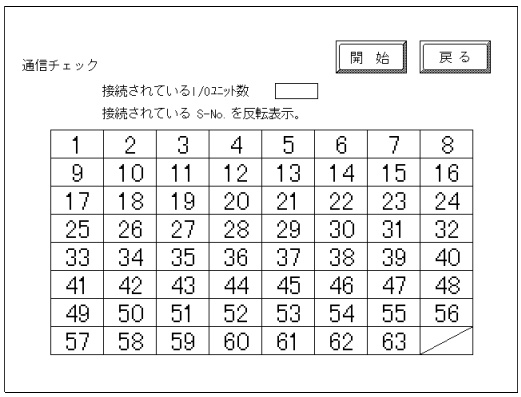
[開始]ボタンを押すと、通信チェックが開始されます。

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。

GLC100/GLC2300シリーズ



GLC300/GLC2400/GLC2600シリーズ



[戻る]ボタンを押すと、**［Flex Networkメニュー］画面**に戻ります。

＜エラーS-No.表示を選択する場合＞

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。

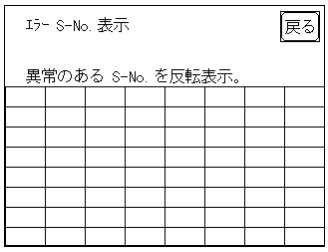
参照→6.4.3 Flex Network I/Fユニット使用時のトラブルシューティング

[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。

[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。

GLC100/GLC2300シリーズ



GLC300/GLC2400/GLC2600シリーズ

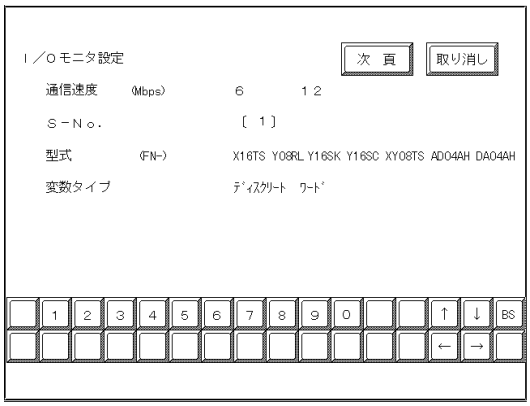
6.2.2 I/O モニタ（I/O 工事接続チェック）

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。

◆ I/O モニタ設定

GLC100/GLC2300 シリーズ

GLC300/GLC2400/GLC2600 シリーズ



・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

「FN-X16TS」、「FN-XY08TS」、「FN-Y08RL」、「FN-Y16SK」、「FN-Y16SC」、「FN-AD04AH」、「FN-DA04AH」の中から選択します。

FN-XY16SK および FN-XY16SC を使用する場合は、入力用として「FN-X16TS」を選択し、出力用として「FN-Y16SKS」、または「FN-Y16SC」を選択してください。

FN-X32TS を使用する場合は、「FN-XY16TS」を選択してください。下位 16 ビットは I/O ユニットで設定した S-No. を指定してください。上位 16 ビットは I/O ユニットで設定した S-No. に 1 加算した値を設定してください。

※「FN-XY16SK」、「FN-XY16SC」、「FN-X32TS」は GLC2000 シリーズでのみ使用可能です。

・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。

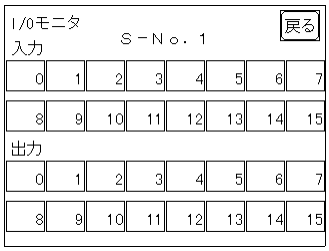
＜FN-X16TS ／ FN-XY08TS ／ FN-Y08RL ／ FN-Y16SK ／ FN-Y16SC ／ FN-XY16SK ／ FN-XY16SC ／ FN-X32TS の場合＞

◆ I/O モニタ（[変数タイプ] が「ディスクリート」の場合）

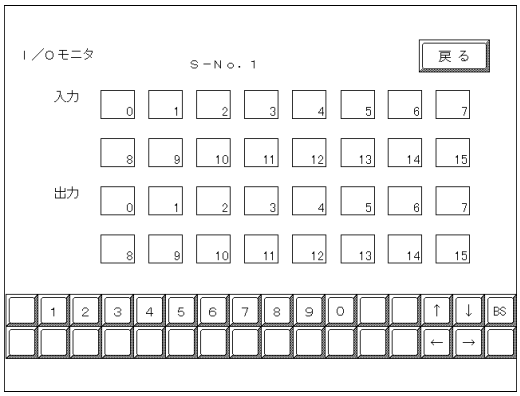
入力部分は入力のあった端子番号が反転表示されます。出力部分は端子番号をタッチして反転表示させると出力されます。

[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。

GLC100/GLC2300 シリーズ



GLC300/GLC2400/GLC2600 シリーズ



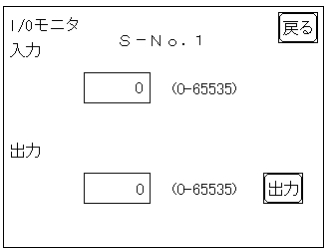
上記画面は、Flex Network システムの 1 つの I/O ユニットの最大入出力点数を表示しています。I/O ユニットの機種により、入力点数、出力点数は異なります。0 を先頭に各 I/O ユニットの持つ点数範囲内で使用してください。

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。

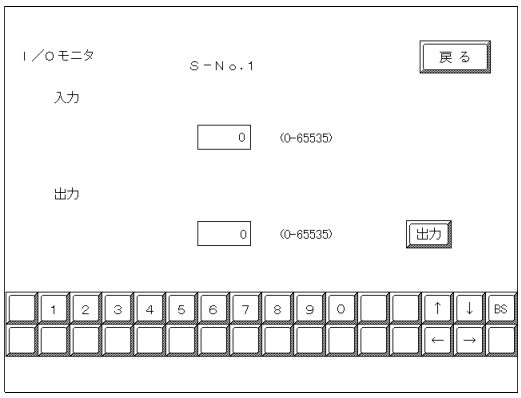
◆ I/O モニタ（[変数タイプ] が「ワード」の場合）

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。GLC100、GLC2300 シリーズ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。

GLC100/GLC2300 シリーズ



GLC300/GLC2400/GLC2600 シリーズ



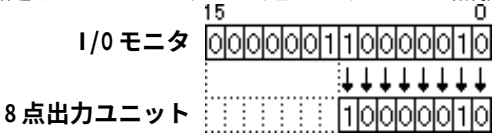
重要 ・ 各 I/O ユニットの I/O 点数に応じて、出力できる範囲のデータを入力してください。

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。

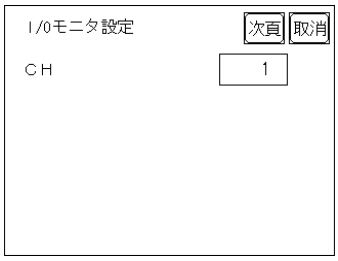


＜ FN-AD04AH ／ FN-DA04AH の場合＞

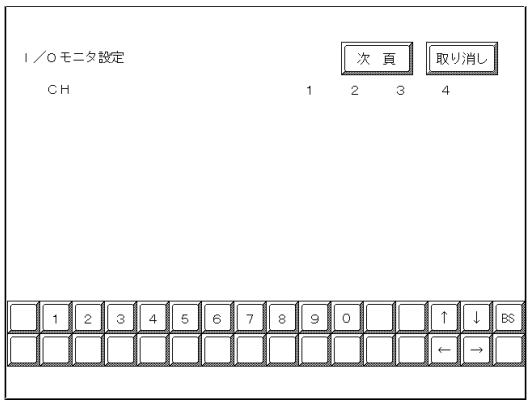
◆ I/O モニタ設定（チャンネル設定）

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。

GLC100/GLC2300 シリーズ



GLC300/GLC2400/GLC2600 シリーズ



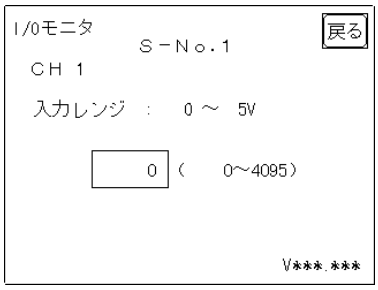
[次頁]ボタンを押すと、次の**[I/O モニタ]画面**に切り替わります。FN-AD04AH と FN-DA04AH では画面が異なります。

＜FN-AD04AH の場合＞

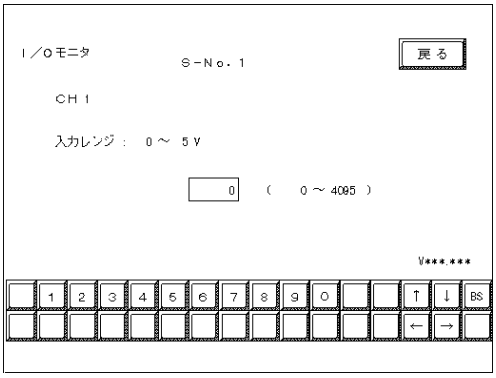
◆ I/O モニタ

入力データを表示します。

GLC100/GLC2300 シリーズ



GLC300/GLC2400/GLC2600 シリーズ



[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

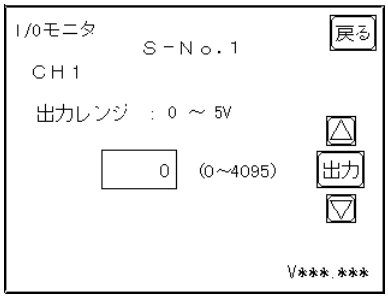
重要
- フィルタタイプ、A/D 変換サンプル回数、最大／最小除外設定は、I/O ユニット側に保存されている設定内容で動作します。I/O ユニット側に保存されている設定内容を変更するには、Pro-Control Editor から設定内容を変更し、GLC にロジックプログラムをダウンロードします。その後、ロジックプログラムを RUN モードにして有効になります。
- レンジ切り替えスイッチの設定内容は、I/O ユニットの電源投入時のみユニット内部に読み込まれます。レンジ切り替えスイッチの設定を変更する時は、必ず I/O ユニットの電源を一度切ってから再投入してください。
- I/O ユニット側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＞

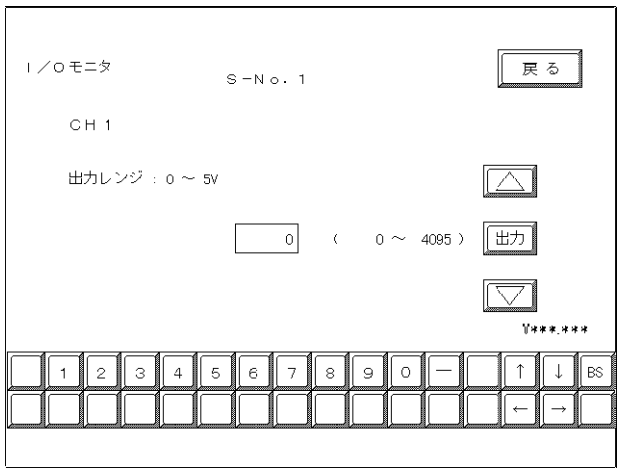
◆ I/O モニタ

テンキーでデータを入力してください。GLC100 シリーズ、GLC2300 シリーズ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。

GLC100/GLC2300 シリーズ



GLC300/GLC2400/GLC2600 シリーズ



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。

6.2.3 Flex Network ユニット使用時のトラブルシューティング

ここでは、Flex Network ユニット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 ユニットの入力 / 出力異常

Flex Network ユニット使用時の入力 / 出力異常につきましては、各 Flex Network ユニットのユーザーズマニュアル（別売）を参照してください。

■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。ここでは Flex Network ユニットを接続したときに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	プロジェクトファイルが破損しているか、プロジェクト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ユニットが同じ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	プロジェクトファイルが破損しているか、プロジェクト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ユニットが同じ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ユニットが同じ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	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ユニットが同じ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	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。このエラーは、ほとんどの場合、Flex Network I/Fユニットがないときに表示されます。Flex Network I/Fユニットが正しく装着されているか確認してください。
822	イニシャル異常 イニシャル処理でFlex Networkドライバとの同期が取れていません	Flex Network I/Fユニットの異常が考えられます。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	GLC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、GLC本体の異常が考えられます。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。

6.3 DIOドライバ

GLC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、予めPro-Control 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。また、DIOユニットが装着されていることを確認してください。

オフラインモードに移る方法は、**「GLC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。

6.3.1 DIOユニットの自己診断

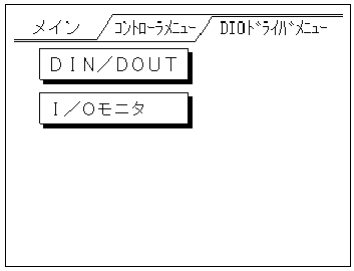
ここではDIOユニットの自己診断の方法について説明します。

GLC本体の自己診断については、
参照→「GLCシリーズユーザーズマニュアル」（別売）

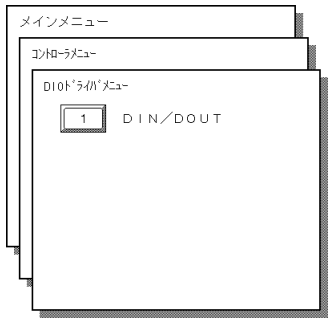
コントローラメニューで[DIOドライバ]を選択すると、下の画面が表示されます。

＜DIOドライバを選択する場合＞

GLC100シリーズ

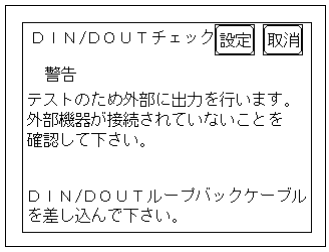


GLC300シリーズ

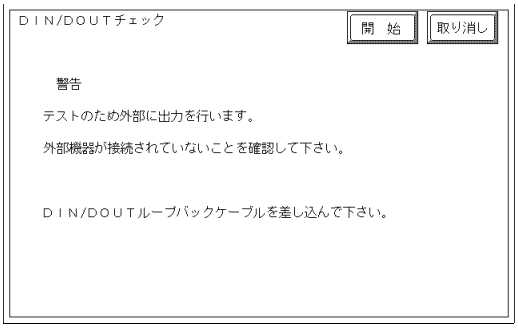


[DIN/DOUT]を押すと下の画面が表示されます。

GLC100シリーズ



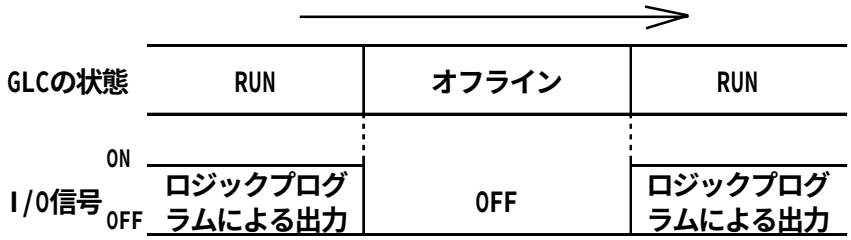
GLC300シリーズ



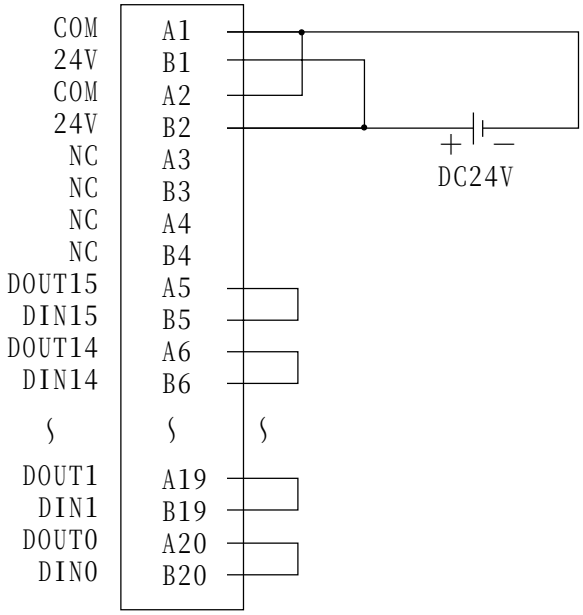
[設定]/[開始]を押すとチェックが開始されます。

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や、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。**



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＞

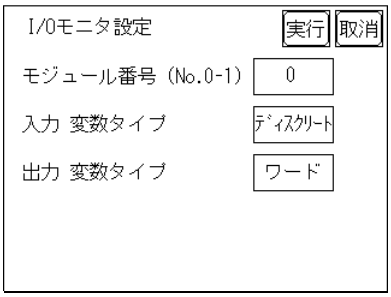
接続方法	メーカー	型番
ハンダ付けタイプ	富士通（株）	FCN-361J040-AU (コネクタ) FCN-360C040-B (カバー)
圧着タイプ	富士通（株）	FCN-363J040 FCN-363J-AU/S FCN-360C0404-B
端子台ユニットタイプ	三菱電機（株）	A6TBX36 (端子台ユニット) AC＊＊TB (ケーブル) (＊＊は、ケーブル長を表します。)
	横河電気（株）	TA40-ON

6.3.2 I/O モニタ（I/O 工事接続チェック）

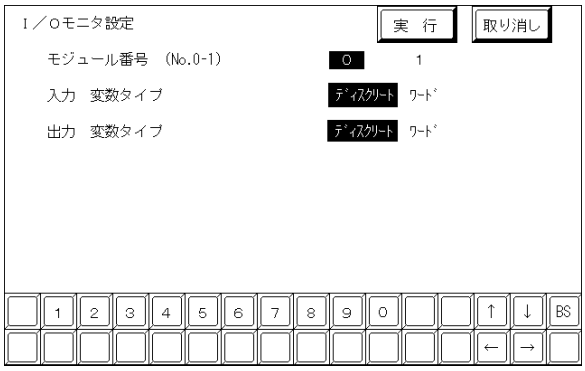
DIO ドライバメニューで[I/O モニタ]を選択すると、下の画面が表示されます。

＜I/O モニタを選択する場合＞

GLC100 シリーズ



GLC300 シリーズ



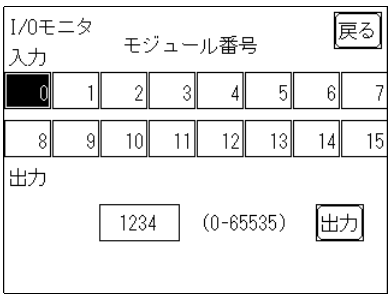
[モジュール番号]は「0」、「1」より選択します。（0、1 はそれぞれ GLC 側のユニット、外側のユニットを表しています。）

[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。

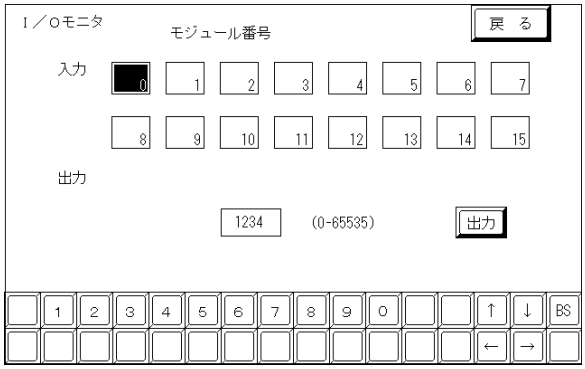
[出力変数タイプ]は「ディスクリート」、「ワード」より選択します。

例えば、[I/O モニタ設定]画面で「モジュール番号 0」、「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。

GLC100 シリーズ



GLC300 シリーズ



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。GLC100 シリーズ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。

6.3.3 DIOユニット使用時のトラブルシューティング

ここでは、DIOユニット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ユニットの入力異常

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ユニットの不良	DIOユニットの交換
	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ユニットの不良	DIOユニットの交換
	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
	DIOユニットの取り付け異常	DIOユニットを確実にネジで取り付ける
	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ユニットの不良	DIOユニットの交換
特定の入力がONしない。	DIOユニットの不良	DIOユニットの交換
	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ユニットの不良	DIOユニットの交換
	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ユニットの出力異常

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ユニットの不良	DIOユニットの交換
	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ユニットの不良	DIOユニットの交換
	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
	DIOユニットの取り付け異常	DIOユニットを確実にネジで取り付ける
出力すべてがOFFしない。	DIOユニットの不良	DIOユニットの交換
特定の出力がONしない。	DIOユニットの不良	DIOユニットの交換
	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ユニットの不良	DIOユニットの交換
	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。ここではDIOユニットを接続したときに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。プロジェクト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	2台のDIOユニットが同じモジュール番号を使用しています。モジュール番号が重複しないように設定し直してください。
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か1に設定してください。
804	ユニット番号が1から始まっています	GLCに近い方のDIOユニットを0に設定してください。

◆初期化エラー

エラーコード	内容	対処方法
821	WLLファイルに記述されているDIOユニットの数と実際に接続されているDIOユニットの数が一致しません。	接続するDIOユニットの数を設定し直してください。
822	モジュール 0がありません。 GLCに近い方のDIOユニットがありません。	ユニットが確実に装着されていることを確認し、DIOドライバの設定を見直してください。
823	モジュール 1がありません。 GLCに遠い方のDIOユニットがありません。	ユニットが確実に装着されていることを確認し、DIOドライバの設定を見直してください。

◆ランタイムエラー

エラーコード	内容	対処方法
840	モジュール 0 読み出しデータが不正です。GLCに近い方のDIOユニットから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	モジュール 1 読み出しデータが不正です。GLCに遠い方のDIOユニットから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。
842	モジュール 0 出力データが不正です。内部ループバックチェックにてGLCに近い方のDIOユニット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。
843	モジュール 1 出力データが不正です。内部ループバックチェックにてGLCに近い方のDIOユニット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。

◆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、（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。

6.4 ユニワイヤ I/F ドライバ

GLC のオフラインモードにあるユニワイヤドライバメニューについて説明します。ユニワイヤドライバメニューを実行するには、予め Pro-Control Editor よりユニワイヤ I/F ドライバをダウンロードしておいてください。また、ユニワイヤ拡張 I/F ユニットが装着されていることを確認してください。

オフラインモードに移る方法は、**「GLC 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。

6.4.1 ユニワイヤ拡張 I/F ユニットの自己診断

ここではユニワイヤ拡張 I/F ユニットの自己診断の方法について説明します。

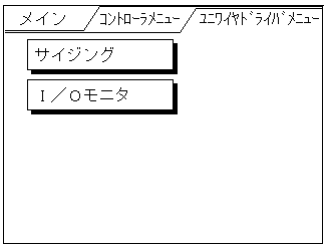
GLC 本体の自己診断については、

参照→「GLC シリーズユーザーズマニュアル」（別売）

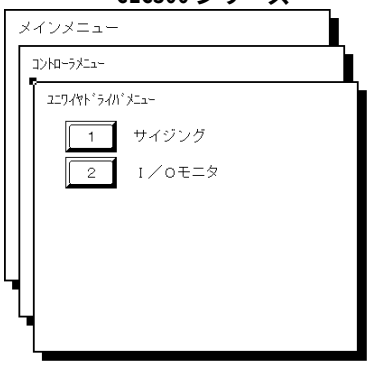
コントローラメニューで[ユニワイヤドライバ]を選択すると、下の画面が表示されます。

＜サイジングを選択する場合＞

GLC100 シリーズ

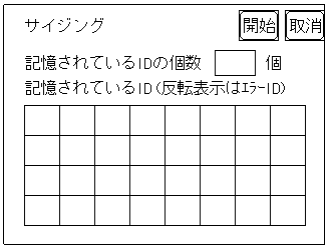


GLC300 シリーズ

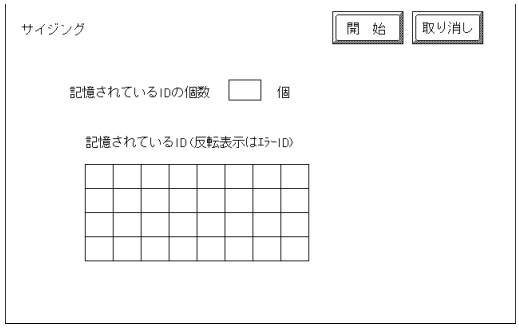


[サイジング]を押すと下の画面が表示されます。

GLC100 シリーズ



GLC300 シリーズ

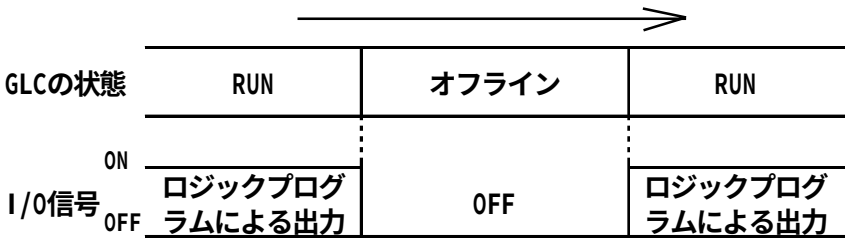


[開始]を押すとサイジングが開始されます。

サイジングとは接続されているターミナルの ID(ターミナル番号)を GLC に記憶させることです。この処理により伝送ラインの断線を検知できるようになります。

- サイジングを行うには、ユニワイヤ拡張 I/F ユニット側面の CERR LED が点滅している必要があります。点滅しない場合は、伝送点数・伝送距離を再設定し再起動してください。
- エラーコードはサイジングを行うと消去されます。
- ID(ターミナル番号)が記憶されていない場合は、断線検知は行われません。

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や、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。**

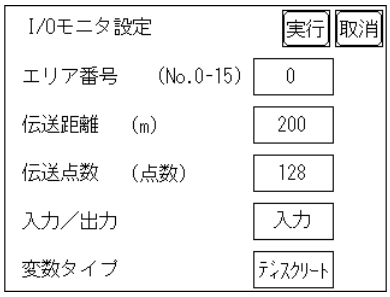


6.4.2 I/O モニタ（I/O 工事接続チェック）

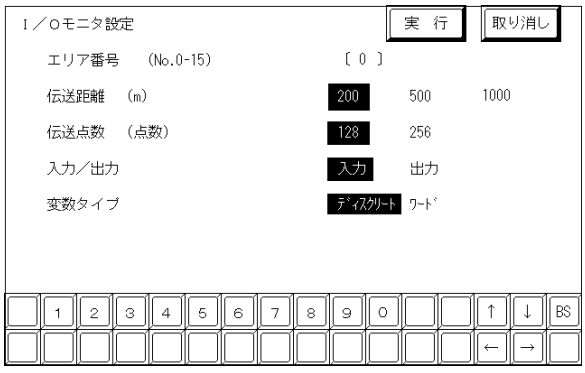
ユニワイヤドライバメニューで[I/Oモニタ]を選択すると、下の画面が表示されます。

＜I/O モニタを選択する場合＞

GLC100 シリーズ



GLC300 シリーズ



[エリア番号]は「0-15」より選択します。

[伝送距離]は「200m」、「500m」、「1000m」から選択します。

[伝送点数]は「128 点」、「256 点」より選択します。

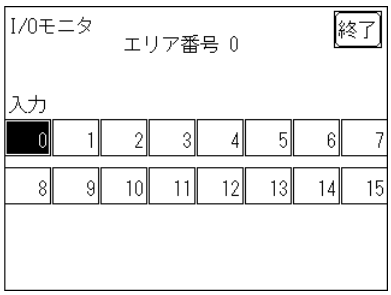
[入力 / 出力]は「入力」、「出力」より選択します。

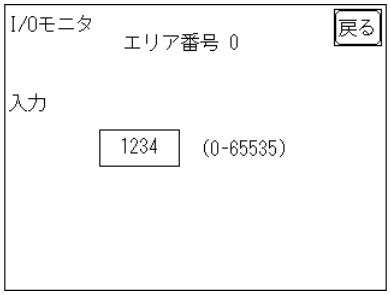
[変数タイプ]は「ディスクリート」、「ワード」より選択します。

各項目を入力、または設定した後、画面右上の実行ボタンをタッチすると確定し、以下の画面が表示されます。

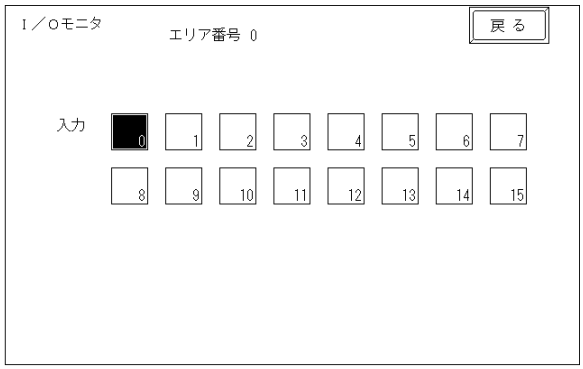
「I/O モニタ設定」画面で「入力」を選択する場合

GLC100 シリーズ



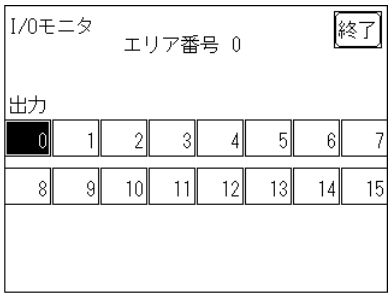


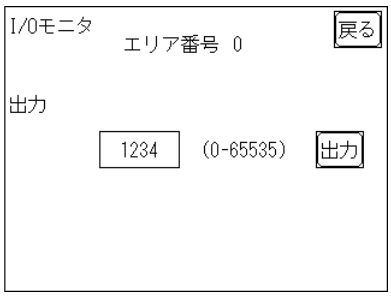
GLC300 シリーズ



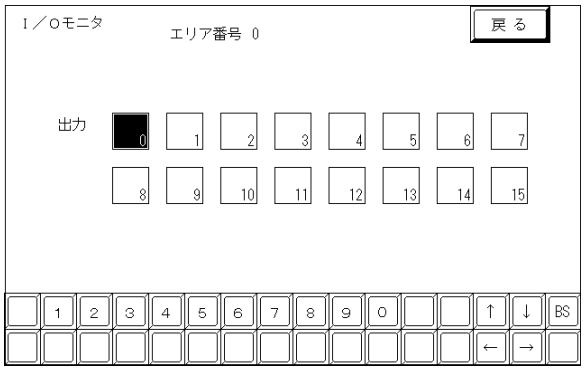
「I/O モニタ設定」画面で「出力」を選択する場合

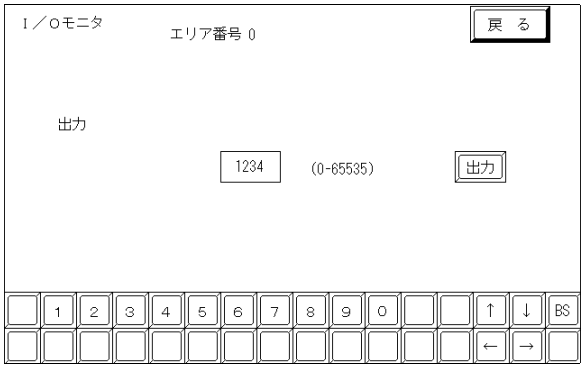
GLC100 シリーズ





GLC300 シリーズ





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。GLC100 シリーズ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。

データ入力後、「出力」ボタンを押すとデータを出力されます。データ表示は 10 進数です。

6.4.3 ユニワイヤ拡張 I/F ユニット使用時のトラブルシューティング

ここでは、ユニワイヤ拡張 I/F ユニット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。

■ユニワイヤ拡張 I/F ユニットの入力 / 出力異常

ユニワイヤ拡張 I/F ユニット使用時の入力 / 出力異常につきましては、ユニワイヤ拡張 I/F ユニットのユーザーズマニュアルをご覧ください。

■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。ロジックプログラムの実行は続けられます。ここではユニワイヤ拡張 I/F ユニットを接続したときに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/0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。プロジェクトファイルを再ダウンロードしてください。
802	エリア番号が重複しています	2つ以上のエリアが同じエリア番号を使用しています。エリア番号が重複しないように設定し直してください。
803	エリア番号が範囲を超えています	伝送点数128点の場合は0～7Fまで、256点の場合は0～FFまで入出力エリアを設定できます。範囲内になるように設定し直してください。
804	スキャンタイム設定エラー	伝送距離・伝送点数による適切なスキャンタイム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	ユニワイヤ拡張I/Fユニットがありません	ユニワイヤ拡張I/Fユニットから読み出したID番号が不正です。ユニワイヤ拡張I/Fユニットがないときに表示されます。
822	イニシャル異常 イニシャル処理でユニワイヤI/Fドライバとユニワイヤ拡張I/Fユニットの同期が取れていません	ハード異常が考えられます。 **参照→ユニワイヤ拡張I/Fインターフェイスのユーザーズマニュアル**
823	内蔵メモリ異常 ユニワイヤ拡張I/Fユニットのデュアルポートメモリのリード/ライトチェックで異常が発生しました	設定を見直してプロジェクトファイルを再転送してください。問題が解決されない場合は、ハード異常が考えられます。

◆ランタイムエラー

エラーコード	内容	対処方法
841	ユニワイヤ拡張I/Fユニットがリセットされました	GLCを再起動してください。
842	I/Oリフレッシュ異常	設定を見直してプロジェクトファイルを再転送してください。問題が解決されない場合は、ハード異常が考えられます。
843	D-G間の短絡	D-G間が短絡していないか確認してください。
844	D、Gラインの断線	D、Gラインが断線していないか確認してください。
845	D-24V間に電源が供給されていない	電源が供給されているか確認してください。

◆内部エラー

エラーコード	内容	対処方法
850 ⋮ 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、（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。

第7章 エラーと異常処理

7.1 エラーメッセージ

ここでは、GLC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Pro-Control 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→「GLCシリーズ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"	GLC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GLC上のロジックプログラムファイルが壊れています。	Pro-Control 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"	GLC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記述されているI/OドライバとGLC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、（株）デジタル・サポートダイヤルまでご連絡ください。
"Can't Set Priority"	GLC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。	GP-PRO/PBⅢで「機種タイプ」にGLCを選択されていることを確認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 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（株）デジタル・サポートダイヤルまでご連絡ください。
"SRAM checksum error"	SRAMに保存されているプロジェクトファイルが壊れています。(GLC2000シリーズのみ)	Pro-Control 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。
"SRAM data broken"	SRAMのバックアップ用バッテリーが切れた可能性があります。このメッセージは警告メッセージです。（GLC2000シリーズのみ）	FEPROMのプロジェクトファイルから実行します。オンラインエディットでロジックプログラムを変更していないか確認してください。
"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されていない。 ・GP-PRO/PBⅢ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る。	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 Array: xxx"	GP-PRO/PBⅢのインポートした配列変数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。	ロジックプログラムファイルを保存した後、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"	GLCの変数「xxx」の変数タイプがGP-PRO/PBⅢの変数タイプと異なります。	ロジックプログラムファイルを保存した後、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。	ロジックプログラムファイルを保存した後、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"	使用しているタグや部品が多すぎます。	使用しているタグや部品を減らして、プロジェクトファイルを再ダウンロードしてください。
"Logic Program is Empty"	GLC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GLC上（FEPROM)のロジックプログラムファイルが壊れています。（GLC2000シリーズのみ）	Pro-Control Editorからロジックプログラムファイルをダウンロードし直してください。
"No backup logic program in FEPROM"	オンラインエディットをした後、FEPROMにプロジェクトファイルをコピーしていない。このメッセージは警告メッセージです。（GLC2000シリーズのみ）	GLCのオフラインメニューでプロジェクトファイルをFEPROMへコピーしてください。
"Exception 65532 [xxxx:xxx]" "Exception 65533 [xxxx:xxx]" "Exception 65534 [xxxx:xxx]" "Exception 65535 [xxxx:xxx]"	GLC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GP-PRO/PBⅢでGLC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。
"Exception 137 [xxxx:xxx]"	対応していないI/Oが設定されています。	I/Oコンフィギュレーションを確認して、I/Oを割り付け直してください。

7.2 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	メジャー	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 ※1
14	-	システムで予約
15	マイナー	バックアップメモリ(SRAM)のロジックプログラムが壊れています。 FEPROMのロジックプログラムが実行されます。 ※1

※1　GLC2000シリーズでのみ発生するエラーです。

「メジャー異常」と「マイナー異常」

メジャー異常は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。

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。

7.3 プログラムの動作異常

ここではPro-Control Editorの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	本機交換
	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	GP-PRO/PBⅢで、プロジェクトファイルを再ダウンロードする。**参照→Pro-Control Editorオペレーションマニュアル 5.2 GLCへのダウンロード**
STOPモードになっているのに、I/O出力されている。	出力データRUN/STOP切り換え時、I/O出力の保持が有効になっている。	当機能を無効にする。**参照→Pro-Control Editorオンラインヘルプ**
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。	システム変数 #FaultCodeの内容を確認し、プログラムを修正する。**参照→Pro-Control Editorオペレーションマニュアル 3.4 システム変数の表示** プログラムを見直し、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。**参照→3.2.25 #FaultCode、3.2.29 #Command**
Pro-Control Editorでモニタリングモードに入れない。	GP-PRO/PBⅢ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、GLCやパソコンの電源が落ちたなどの電気的ノイズによりGLC内のシステムファイルが壊れた可能性があります。	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。
Pro-Control 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。		
GP-PRO/PBⅢ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GLC本体や、I/Oユニットへの入出力を可能にする動作です。GLCでは、ロジックプログラムのダウンロードを行った後、GLC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「Pro-Control Editor オペレーションマニュアル 3.1.コントローラの設定、 3.2.コントローラのRUN/STOP」を参照してください。

索引

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。