SoMachine Basic

汎用ファンクションライブラリーガイド

05/2018

EIO0000001474.JP.00

www.schneider-electric.com

本書の情報には本書に記載された製品についての一般的説明および性能の技術特性が含まれます。本書は、お客様の特定の用途に対する本製品の適合性または信頼性を確約するために作成されたものではありません。お客様またはインテグレーター様は自らの責任で、関連する特定の用途またはその使用に関する本製品のリスク分析、評価、および試験を完全かつ適切に行なってください。シュナイダーエレクトリック社あるいは系列会社は、本書に記載された情報の誤用に対して一切の責任を負いかねますので、あらかじめご了承ください。本書の内容について改善点や修正点の提案がある場合、また何らかの誤りを発見した場合には、弊社までご連絡ください。

本製品を設置して使用する際には、関連する州、地域、地区の安全規定をすべて順守する必要があります。安全のため、また、記録されたシステムデータの適合性を確保するため、部品の修理は製造業者にお任せください。

装置を技術的な安全要件がある用途に使用する場合、関連する指示に従ってください。

シュナイダーエレクトリックのハードウェア製品には必ず、シュナイダーエレクトリック製のソフトウェアまたは承認された ソフトウェアをご使用ください。この指示に従わない場合、人的損害、物的損害、また不適切な動作が生じる可能性があります。

この情報に従わない場合、人的損害や装置の損傷を招くおそれがあります。

目次

安全に関する使用上の注意

重要情報

お断り

本書をよくお読みいただき、装置の正しい取り扱いと機能を十分ご理解いただいた上で、設置、操作、保守を行ってください。本書および装置には以下の表示が使われています。これらは潜在的な危険を警告したり、手順を明確化あるいは簡素化する情報について注意を呼びかけるものです。

この記号が「危険」または「警告」安全ラベルに追加されると、電気的な危険が存在し、指示に従わないと人身傷害の危険があることを示します。

安全警告記号です。人的傷害の危険性があることを警告します。
この記号の後に記載された安全に関する情報に従って、人的傷害や死亡の危険性を回避してください。

⚠ 危険

危険は、危険が生じる可能性のある状況を示します。回避しないと、死亡や重傷を**招きます**。

⚠ 警告

警告は、危険が生じる可能性のある状況を示します。回避しないと、死亡や重傷を**招くおそれ**があります。

⚠ 注意

注意は、危険が生じる可能性のある状況を示します。回避しないと、軽傷を**招くおそれ**があります。

注記

この表示は、指示に従わないと物的損害を負う可能性があることを示します。

注意

電子機器の設置、操作、整備は必ず資格のある人物が行ってください。Schneider Electric は、本資料の使用に起因するいかなる結果についても責任を負わないものとします。

資格のある人物とは、電子機器の構造、操作、設置に関する技術および知識を有し、かつ電子機器に伴う危険性を理解しこれを回避するための安全研修を受けた人物を指します。

ご使用の前に

効果的な作業場所の安全対策がない機械では本製品を使用しないでください。機械の作業場所の安全対策の欠如は、その機械のオペレーターに重大な傷害をもたらす可能性があります。

⚠ 警告

安全対策の無い機器

- 作業場所の安全対策を持たない機器ではこのソフトウェアおよび関連する自動機器を使用しないでください。
- 作業中は機械に手を触れないでください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

この自動機器および関連ソフトウェアは、さまざまな産業プロセスを制御するために使用されます。各アプリケーションに適した自動機器のタイプまたはモデルは、必要な制御機能、必要な保護の程度、生産方法、異常な条件、政府の規制などの要因によって変わります。アプリケーションによっては、バックアップの冗長性が必要な場合など複数のプロセッサーが必要な場合があります、

ユーザー、機械製造者またはシステムインテグレーターのみが、機械の設置、操作、メンテナンス中に存在するすべての条件と要因を認識することができます。また効果的かつ適切に使用することができる自動機器および関連する安全装置やインターロックを選定できます。特定のアプリケーション向けに自動機器、制御機器、および関連するソフトウェアを選択する際は、該当する国、地域の基準や規格を考慮してください。National Safety Council's Accident Prevention Manual (米国では全国的に認知されています) も多くの有用な情報を提供しています。

梱包機械などの一部のアプリケーションでは作業者の保護のために作業場所の安全対策などをさらに追加する必要があります。これは、作業者の手や体のその他の部分が、挟まる場所または危険な領域に入り込み、重大な傷害が発生する可能性がある場合に必要です。ソフトウェアだけでは作業者を怪我から守ることはできません。このためソフトウェアは作業場所の安全対策の代わりに使用することはできません。

作業場所の安全対策として適切な安全装置および機械的 / 電気的インターロックが取り付けられており、それらが作動可能であることを、装置の稼動前に確認してください。作業場所の安全対策としてのすべてのインターロックおよび安全装置は、関連する自動機器およびソフトウェアプログラミングで調整する必要があります。

注記：作業場所の安全対策としての安全装置および機械的 / 電気的インターロックの調整は、このマニュアルで参照されているファンクションブロックライブラリー、システムユーザーガイドまたはその他の実装の範囲外です。

スタートアップとテスト

電気制御および装置の通常運転を開始する前に、有資格者によるスタートアップテストを実施し、装置の動作が正しいことを確認する必要があります。このようなテストの手配および十分な時間をかけて完全かつ満足のいくテストを行うことが重要です。

⚠ 警告

装置操作上の危険

- すべての設置およびセットアップ手順が完了していることを確認します。
- 運用テストを実施する前に、すべてのブロックやその他の運搬用の保護梱包などをすべてのコンポーネントデバイスから取り外します。
- 工具、計器およびゴミ等を機器から取り除きます。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

機器のマニュアルで推奨されているすべてのスタートアップテストに従ってください。後に参照するため、すべての機器のマニュアルを保管してください。

ソフトウェアテストは、シミュレーションと実際の環境の両方で行ってください。

地域の規制 (例えば、米国では National Electrical Code) に従って、一時的な接地や短絡が無いことを確認します。高電位電圧テストが必要な場合、機器の損傷を防ぐため機器のマニュアルに記載されている推奨事項に従ってください。

機器に通電する前に以下のことを行ってください。

- 工具、計器およびゴミ等を機器から取り除きます
- 装置エンクロージャーのドアを閉じます
- 電力供給ラインからすべての一時的な接地を取り除きます
- メーカーの推奨するすべてのスタートアップテストを行います

運転と調整

以下の安全上の注意は NEMA Standards Publication ICS 7.1-1995 (英語版優先) に対応しています。

- 機器の設計、製造または部品の選定、評価過程で注意を払っても、これらの機器が不適切に操作された場合、危険性があります。
- 機器の誤った調整により、不十分な動作または危険な動作が生じることがあります。機能調整のガイドとして、常にメーカーの説明書を使用してください。これらの調整ができる作業者は、機器メーカーの指示書および電気機器に使用されている機械類に精通している必要があります。
- オペレーターが実際に必要とする調整だけがアクセスできるようにしてください。他の制御へのアクセスは、不正な動作特性の変更を防ぐために制限する必要があります。

本書について

概要

本書の適用範囲

このガイドでは、SoMachine Basic ソフトウェアで作成するプログラムでのファンクションブロックと命令の使用方法について説明します。本説明は SoMachine Basic によりサポートされるすべてのロジックコントローラーに適用されます。

有効性に関する注意

本書は、SoMachine Basic 対応製品**のみ**を対象として書かれています。

本書は、SoMachine Basic V1.6 のリリース時に更新されました。

本書に記載された機器の技術特性は、オンラインページにも表示されています。この情報にオンラインでアクセスするには、以下を実行します。

ステップ	アクション
1	シュナイダーエレクトリックのホームページに移動します：*www.schneider-electric.com*。
2	**検索** ボックスに製品の参照番号または製品ライン名を入力します。 ● 参照番号または製品ライン名にはスペースを含めないようにしてください。 ● 類似するモジュールのグループに関する情報を表示するには、アスタリスク（*）を使用します。
3	参照番号を入力した場合は、**製品データシート**検索結果に移動して目的の参照番号をクリックします。 製品ラインを入力した場合は、**製品ライン**検索結果に移動して目的の製品ラインをクリックします。
4	**製品**検索結果に複数の結果が表示された場合は、目的の参照番号を選んでクリックします。
5	画面サイズににによっては、データシート全体を表示するには画面をスクロールダウンしなければならない場合があります。
6	データシートを .pdf ファイルとして保存または印刷するには、**XXX 製品のデータシートをダウンロード**をクリックします。

シュナイダーエレクトリックでは、本マニュアル内に記載された製品特性とオンラインページの記載内容が一致するよう務めていますが、継続的改善を目指す当社の方針に従い、情報をより明確かつ正確なものにするため内容を改訂させていただく場合があります。マニュアルとオンラインページの情報が一致していない場合は、オンラインページの情報を参照してください。

関連マニュアル

マニュアルタイトル	参照番号
SoMachine Basic - オペレーティングガイド	*EIO0000001354 (ENG)* *EIO0000001355 (FRA)* *EIO0000001356 (GER)* *EIO0000001357 (SPA)* *EIO0000001358 (ITA)* *EIO0000001359 (CHS)* *EIO0000001366 (POR)* *EIO0000001367 (TUR)*
Modicon M221 ロジックコントローラーアドバンストファンクション - ライブラリーガイド	*EIO0000002007 (ENG)* *EIO0000002008 (FRE)* *EIO0000002009 (GER)* *EIO0000002010 (SPA)* *EIO0000002011 (ITA)* *EIO0000002012 (CHS)* *EIO0000002013 (POR)* *EIO0000002014 (TUR)*

マニュアルタイトル	参照番号
Modicon M221 ロジックコントローラー - プログラミングガイド	EIO0000001360 (ENG) EIO0000001361 (FRE) EIO0000001362 (GER) EIO0000001363 (SPA) EIO0000001364 (ITA) EIO0000001365 (CHS) EIO0000001369 (TUR) EIO0000001368 (POR)
Modicon M221 ロジックコントローラー - ハードウェアガイド	EIO0000001384 (ENG) EIO0000001385 (FRA) EIO0000001386 (GER) EIO0000001387 (SPA) EIO0000001388 (ITA) EIO0000001389 (CHS) EIO0000001370 (POR) EIO0000001371 (TUR)

マニュアルや技術情報はシュナイダーエレクトリックサポートサイト「おたすけ Pro！」からダウンロードできます。https://www.schneider-electric.com/en/download

製品関連情報

⚠ 警告

制御不能

- 制御手法の設計者は制御パスの障害モードが発生するおそれを考慮する必要があり、特定の重要制御機能については、パス障害の最中および終了後に安全な状態を実現するための方策を準備しておく必要があります。重要制御機能の例としては、緊急停止、オーバートラベル停止、停電、および再起動があります。
- 重要な制御機能に対しては、別のまたは冗長性のある制御パスを用意してください。
- システム制御パスには、データ通信が含まれることがあります。予期しないデータの転送遅れや障害について考慮する必要があります。
- あらゆる事故防止規制および地域の安全性ガイドライン [1] を遵守してください。
- 運用を開始する前に、各実装について、正しく動作するかどうかを個別に十分にテストする必要があります。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

[1] 詳細は、NEMA ICS 1.1 (最新版)、"Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control"、および NEMA ICS 7.1 (最新版)、"Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems"、または該当地域での同等のガイドラインを参照してください。

⚠ 警告

装置の意図しない動作

- 本装置には、Schneider Electric 認定のソフトウェアのみ使用してください。
- ハードウェアの設定を変更した場合は、必ずアプリケーションプログラムも更新してください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

規格から派生した用語

技術用語、専門用語、シンボル、本書の記述、また本製品での表示は、国際規格用語および定義に由来しています。

安全機能システム、ドライブ、一般オートメーションにおいて、用語は、*安全性*、*安全機能*、*安全状態*、*異常*、*異常リセット*、*誤動作*、*障害*、*エラー*、*エラーメッセージ*、*危険*等を含みますが、それに限定されません。

特に以下の規格が含まれます。

規格	詳細
EN 61131-2: 2007	プログラマブルコントローラー、第 2 部 : 機器要件、および試験
ISO 13849-1: 2008	機械類の安全性 : 制御システムの安全関連部 設計の一般原則
EN 61496-1: 2013	機械類の安全性 : 電気的検知保護装置 第 1 部 : 一般要件、および試験
ISO 12100: 2010	機械類の安全性 - 設計の一般原則 - リスク評価とリスク低減
EN 60204-1: 2006	機械類の安全性 - 機械の電気装置 - 第 1 部 : 一般要件
EN 1088: 2008 ISO 14119: 2013	機械類の安全性 - ガードと共同するインターロック装置 - 設計、および選択のための原則
ISO 13850: 2006	機械類の安全性 - 非常停止 - 設計原則
EN/IEC 62061: 2005	機械類の安全性 - 安全関連の電気・電子・プログラマブル電子制御システムの機能安全
IEC 61508-1: 2010	電気・電子・プログラマブル電子安全関連系の機能安全 : 一般要求事項
IEC 61508-2: 2010	電気・電子・プログラマブル電子安全関連系の機能安全 : 電気・電子・プログラマブル電子安全関連系に対する要求事項
IEC 61508-3: 2010	電気・電子・プログラマブル電子安全関連系の機能安全 : ソフトウェア要求事項
IEC 61784-3: 2008	計測制御用デジタルデータ通信 : 機能安全フィールドバス
2006/42/EC	機械指令
2014/30/EU	電磁両立性指令
2014/35/EU	低電圧指令

本書で使われている用語には下記の規格も含まれています。

規格	詳細
IEC 60034 シリーズ	回転電気機械
IEC 61800 シリーズ	可変速電気駆動システム
IEC 61158 シリーズ	計測制御用デジタルデータ通信 – 産業制御システム用のフィールドバス

動作領域 は特定の危険性記述と併せて使われる場合があり、*機械指令* (*2006/42/EC*) と *ISO 12100 : 2010* の *危険区域* と同様に定義されています。

注記：前述の規格は、本書記載の特定の機器には適用されない場合があります。本書に記載されている製品の適用規格についての詳細は製品の特徴が記載された表を参照してください。

第 1 章
概要

概要

この章では、本書内の演算や代入命令の例を実行するために必要なブロックおよびソースコードの例の使用方法について説明します。

この章について

この章には次の項目が含まれています。

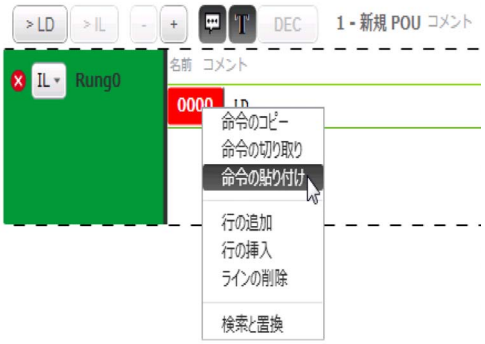
ソースコード例の使用方法

概要

明示的に言及されている場合を除いて、本書に含まれるソースコードの例は、ラダー図および命令リスト (IL) プログラミング言語の両方に有効です。完全な例では複数のラングが必要な場合があります。

可逆性の手順

同等のラダー図ソースコードの取得は次の手順で行います。

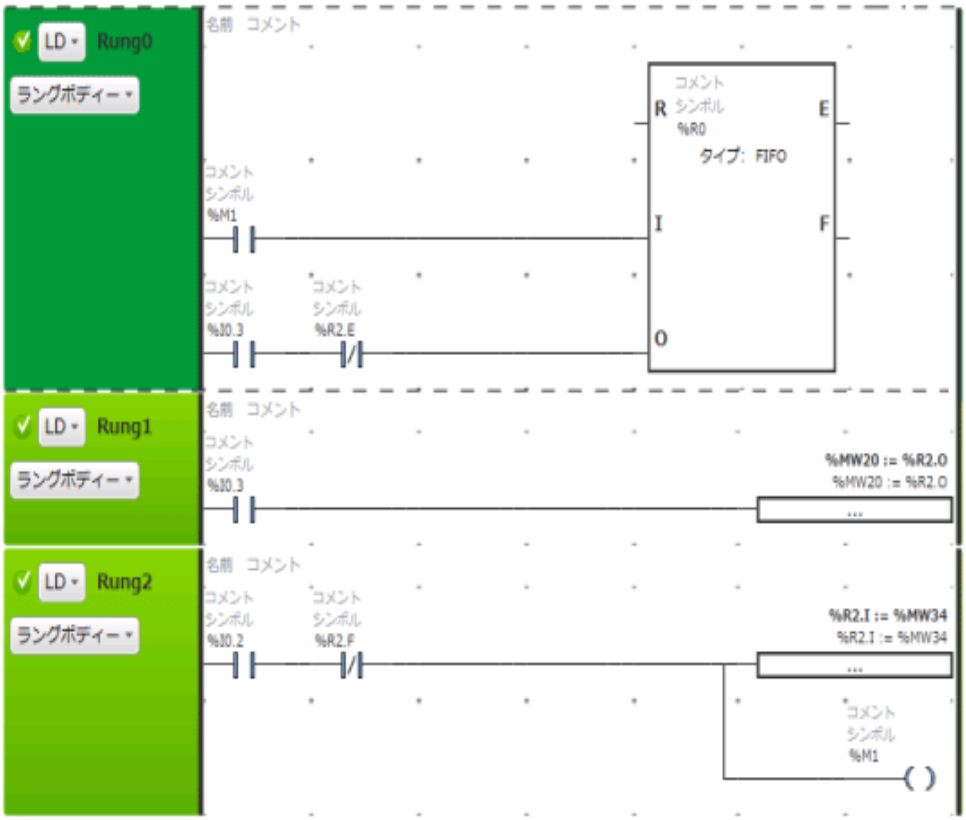
手順	アクション
1	このマニュアルに示すサンプルプログラムの 1 番目のラングのソースコードを選択してコピー (**Ctrl + C**) します。
2	SoMachine Basic で、ツールバー上の  をクリックして新しいラングを作成します。
3	作成したラングで **LD > IL** ボタンをクリックし、命令リスト (IL) ソースコードを表示します。
4	行番号 **0000** を選択し、右クリックで**命令の貼り付け**を選択して、ソースコードをラングに貼り付けます。  **注記**：デフォルトの LD 演算子の前に行を挿入して命令を貼り付けている場合、ラングの最後の行から **LD** 命令を削除することを忘れないでください。
5	**IL > LD** ボタンをクリックすると、ラダー図のソースコードが表示されます。
6	サンプルプログラムの追加のラングについては、前の手順を繰り返します。

例

命令リスト (IL) プログラム

ラング	ソースコード
0	BLK %R0 LD %M1 I LD %I0.3 ANDN %R2.E O END_BLK
1	LD %I0.3 [%MW20:=%R2.0]
2	LD %I0.2 ANDN %R2.F [%R2.I:=%MW34] ST %M1

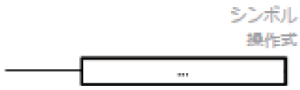
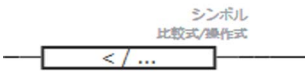
対応するラダー図


LD
Rung0
ラングボディー
名前 コメント
コメント
シンボル
%M1
コメント
シンボル
%I0.3
コメント
シンボル
%R2.E
コメント
シンボル
%R0
タイプ: FIFO
R
E
I
F
O
LD
Rung1
ラングボディー
名前 コメント
コメント
シンボル
%I0.3
%MW20 := %R2.O
%MW20 := %R2.O
...
LD
Rung2
ラングボディー
名前 コメント
コメント
シンボル
%I0.2
コメント
シンボル
%R2.F
%R2.I := %MW34
%R2.I := %MW34
...
コメント
シンボル
%M1
( )

演算ブロック

ラダー図に、代入命令と IL 演算の挿入

演算ブロックのグラフィックシンボルを使って、ラダー図のラングに IL 演算や代入命令を挿入できます。

 ラングの終端にある演算ブロック。	 ラング内の任意の場所にある演算ブロック。

演算ブロックのグラフィックシンボルは、1 列目以外のラダー図の任意の位置に挿入できます (ラングの最初の接点として使用できないため)。

ラダー図のラングで 1 つ以上の**演算ブロック**のグラフィックシンボルを使用する場合は、それらを直列に配置してください。**演算ブロック**命令を並列に使用することはできません。

注記：
アプリケーションが**レベル 5.0** 以上のファンクションレベル で構成されている場合。

- 演算ブロックでは、最大 5 つのオペランドと 入れ子の括弧を最大 3 つまで使用できます。オペランドには同じ型のオブジェクトを使用します (ワード型同士、フロート型同士など)。
- マスタータスクで複数のオペランドを使用するには、最低 20 のワードメモリー (%MW) が利用できる必要があります。周期タスクでも複数のオペランドを使用する場合は、さらに 20 個のワードメモリーを利用できる必要があります。

次の手順で、ラダー図のラングに演算ブロックグラフィックを挿入できます。

手順	手順内容
1	ツールバー上の**演算ブロック**ボタン  をクリックします。
2	ラング上をクリックして、**演算ブロック**を挿入します。
3	ツールバー上の**選択モード**ボタン  をクリックします。
4	**演算式**ラインを、ダブルクリックします。 スマートコーディング (*16* ページ参照) ボタン  が、ラインの最後に表示されます。このボタンをクリックすると、ファンクションと命令の構文が選択できます。
5	IL 演算または代入命令を入力し、ENTER を押します。 例 : %MF10 := ((SIN(%MF12 + 60.0) + COS(%MF13)) + %MF10) + 1.2 オンラインモードで式を編集できます。オンライン編集 を参照してください。

注記：マルチオペランド式はイベントタスクでは使えません。

OPER 命令の構文

OPER 命令は、ラングの任意の場所に配置された演算ブロックに対応します。

同等の OPER 命令は、IL ラングで直接使用できます。

OPER [*式*] は、*式*が有効で、最大 5 つのオペランドと最大 3 つの括弧の入れ子が含まれます。例 :

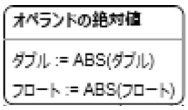
```
OPER [ %MF10 := ((SIN( %MF12 + 60.0 ) + COS( %MF13 )) + %MF10 ) + 1.2]
```

ラダー図のスマートコーディングツールチップ

SoMachine Basic では、演算ブロックでファンクション名を入力時、ファンクションを選択するためにツールチップが表示されます。

ツールチップは、2 種類あります。

- ファンクション名のリスト。リストは入力された文字によって、動的に更新されます。例えば、“AS” と入力すると、ASCII_TO_FLOAT、ASCII_TO_INT、ASIN が表示されます。
- ファンクションの構文のヘルプ。括弧開きを入力したときに表示されます。例えば、“ABS(“ と入力すると、以下が表示されます。



スマートコーディングアシスタントの使用

スマートコーディングアシスタントは、演算式ラインでスマートコーディングボタン  をクリックすると、表示されます。

以下の手順に沿って進めます。

手順	手順内容
1	(オプション) ファンクションのカテゴリーでリストをフィルターします。 ● すべてのタイプ ● ASCII ● 浮動小数点 ● I/O オブジェクト ● 浮動小数点 ● 数値処理 ● テーブル ● PID ● ユーザー定義ファンクション
2	式に追加するファンクションを選択します。
3	**ファンクションの挿入**をクリックします。

構文のヘルプを取得

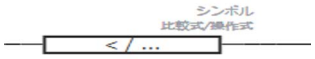
IL 演算の構文または代入命令が正しくない場合、**演算式**ボックスの縁が、赤色になります。次のいずれかでヘルプを参照してください。

- マウスを **演算式**ラインへ移動する。
- **ツール → プログラムメッセージ**を選択する。

比較ブロック

ラダー図に IL 比較式を挿入

比較ブロックのグラフィックシンボルを使って、ラダー図のラングに IL 比較式を挿入できます。



オペランドには同じ型のオブジェクトを使用します (ワード型同士、フロート型同士など)。

以下の手順に沿って進めます。

手順	手順内容
1	ツールバー上の**比較ブロック**ボタン  をクリックします。
2	ラング上をクリックして、**比較ブロック**を挿入します。
3	**比較式**ラインを、ダブルクリックします。
4	IL 比較演算を入力し、ENTER を押します。 オンラインモードで式を編集できます。オンライン編集 *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を参照してください。

注記：
アプリケーションが**レベル 6.0** 以上のファンクションレベル *(EcoStruxure Machine Expert - Basic, オペレーティングガイド参照)* で構成されている場合。

- 比較ブロックでは、最大 5 つのオペランドと最大 3 つの括弧の入れ子が使用できます。
- マスタータスクで複数のオペランドを使用するには、最低 20 のワードメモリー (%MW) が利用できる必要があります。周期タスクでも複数のオペランドを使用する場合は、さらに 20 個のワードメモリーを利用できる必要があります。

注記：マルチオペランド式はイベントタスクでは使えません。

構文のヘルプを取得

IL 比較演算の構文が正しくない場合、**比較式**ボックスの縁が、赤色になります。次のいずれかでヘルプを参照してください。

- マウスを**比較式**ラインへ移動する。
- **ツール → プログラムメッセージ**を選択する。

第 2 章
オブジェクト

この章について

この章には次の項目が含まれています。

オブジェクト

概要

SoMachine Basic では、*オブジェクト*は、アプリケーションで使用するために確保したロジックコントローラーのメモリーエリアです。オブジェクトは以下のような形態をとることができます。

- ビットメモリーやワードメモリーなどの、ソフトウェアの変数
- デジタルまたはアナログの入出力のアドレス
- システムワードやシステムビットなどの、コントローラーの内部変数
- タイマーやカウンターなどの、定義済みシステムファンクションやファンクションブロック

コントローラーメモリーは、事前に特定のオブジェクトタイプに割り当てられるか、ロジックコントローラーにアプリケーションをダウンロードするときに、自動的に割り当てられます。

メモリーが割り当てられると、プログラムでオブジェクトのアドレス指定ができます。オブジェクトのアドレス指定には、先頭に % を付けます。例えば、%MW12 はワードメモリーのアドレス、%Q0.3 は標準デジタル出力のアドレス、%TM0 はタイマーファンクションブロックのアドレスです。

ビットメモリーオブジェクト

概要

ビットメモリーオブジェクトはビット型のソフトウェア変数です。オペランドとして使用でき、ブール命令でテストができます。

ビットオブジェクトの例
- ビットメモリー
- システムビット
- ステップビット
- ワードから抽出されたビット

有効なオブジェクトの範囲は、0 からアプリケーションで使用されている最大数までです (ロジックコントローラーの *プログラミングガイド* を参照してください)。

構文

メモリービット、システムビット、およびステップビットのオブジェクトのアドレス指定は次の形式を使用します。

次の表で、アドレス指定形式の要素について説明します。

グループ	項目	説明
シンボル	%	パーセント記号は常にソフトウェア変数の先頭につきます。
オブジェクトタイプ	M	ビットメモリーはプログラムの実行中の値を格納します。
	S	システムビットはコントローラーのステータスと制御情報を示します。
	X	ステップビットはグラフセステップアクティビティのステータスを示します。
オブジェクトインスタンス識別子	i	メモリー内の連続したインスタンスを表すオブジェクトの識別子。オブジェクトの最大数は、使用可能なメモリーに設定されたオブジェクトの数によって異なります。使用可能なメモリーの最大値については、ロジックコントローラーの *プログラミングガイド* を参照してください。

I/O ビットのアドレス指定については I/O オブジェクト (*23* ページ参照) を参照してください。

ワードから抽出されたビットのアドレス指定については ワードオブジェクトよりビットを抽出 (*27* ページ参照) を参照してください。

説明

ブール命令のオペランドとして使用されるメモリー、システム、およびステップビットオブジェクトを次の表に示します。

タイプ	説明	アドレスまたは値	書き込みアクセス (1)
即値	0 または 1 (False または True)	0 または 1	–
メモリー	ビットメモリーは 2 進値の格納に使用する内部メモリー領域です。 **メモ :** 未使用の I / O オブジェクトはビットメモリーとして使用できません。	%Mi	可
システム	システムビット %S0 ～ %S127 により、コントローラーが正しく動作しているか、およびアプリケーションプログラムが正しく実行されているかを監視したり、特定のシステムレベル機能を制御したりすることができます。	%Si	i により異なる
グラフセステップ	ビット %X1 ～ %Xi はグラフセステップに関連付けられています。対応するステップが有効になると、ステップビット Xi は 1 に設定されます。ステップが無効になると、0 に設定されます。	%Xi	可
(1) プログラムによる書き込みまたはアニメーションテーブルを使用した書き込み			

例

ビットオブジェクトのアドレス指定の例を次の表に示します。

ビットオブジェクト	説明
%M25	ビットメモリー番号 25
%S20	システムビット番号 20
%X4	グラフセステップ番号 4

I/O オブジェクト

概要

I / O オブジェクトにはビットとワードの両方が含まれます。各物理入出力は内部メモリー内のこれらのオブジェクトにマップされます。I / O ビットオブジェクトはオペランドとして使用でき、ブール命令でテストができます。I / O ワードオブジェクトは、算術演算子を含むファンクションや命令などのほとんどの非ブール型命令で使用できます。

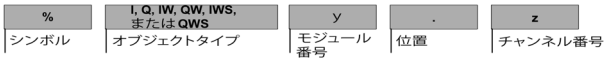
I / O オブジェクトの例

- デジタル入力
- デジタル出力
- アナログ入力
- アナログ出力
- 通信入力および出力

有効なオブジェクトの範囲は、0 からコントローラーで設定および対応している最大値までです (ロジックコントローラーのハードウェアガイドおよびプログラミングガイドを参照してください)。

構文

入出力アドレス形式を次の図に示します。



アドレス指定形式の要素について次の図に示します。

コンポーネント	項目	値	説明
シンボル	%	–	パーセント記号は常に内部アドレスの先頭につきます。
オブジェクトタイプ	I	–	デジタル入力 (ビットオブジェクト)
	Q	–	デジタル出力 (ビットオブジェクト)
	IW	–	アナログ入力値 (ワードオブジェクト)
	QW	–	アナログ出力値 (ワードオブジェクト)
	IWS	–	アナログ入力チャンネルステータス (ワードオブジェクト)
	QWS	–	アナログ出力チャンネルステータス (ワードオブジェクト)
モジュール番号	y	0	ロジックコントローラーまたはロジックコントローラーに挿入されたカートリッジの内蔵 I / O チャンネル。
		1...*m*(1)	コントローラーに直接接続された拡張モジュールの I / O チャンネル。
		m+1...*n*(2)	TM3 送信機 / 受信機モジュールを使用して接続された拡張モジュールの I / O チャンネル。
チャンネル番号	z	0...31	ロジックコントローラーまたは拡張モジュールの I/O チャンネル番号。使用可能なチャンネルの数は、ロジックコントローラーのモデルまたは拡張モジュールのタイプによって異なります。
		*p*0*q*(3)	ロジックコントローラーに挿入されたカートリッジの I/O チャンネル。使用可能なチャンネルの数は、カートリッジのタイプによって異なります。

(1) *m* は設定されたローカルモジュールの番号です (最大 7)。
(2) *m* は設定されたリモートモジュールの番号です (最大 n+7)。最大番号は 14 です。
(3) *p* はコントローラー内のカートリッジの番号です。*q* はカートリッジのチャンネル番号です。

説明

命令のオペランドとして使用されるすべての I/O オブジェクトを次の表に示します。

タイプ	アドレス または値	書き込み アクセス (1)	説明
入力ビット	%Iy.z(2)	不可 (3)	これらのビットは物理デジタル I/O の電気的状態の論理イメージです。それらはデータメモリーに格納され、プログラムロジックの各スキャンの合間に更新されます。
出力ビット	%Qy.z(2)	可	
入力ワード	%IWy.z(2)	不可	これらのワードオブジェクトには、対応するチャンネルのアナログ値が含まれています。
出力ワード	%QWy.z(2)	可	
入力ワードステータス	%IWSy.z(2)	不可	これらのワードオブジェクトには、対応するアナログチャンネルのステータスが含まれます。
出力ワードステータス	%QWSy.z(2)	不可	

(1) プログラムによる書き込みまたはアニメーションテーブルを使用した書き込み
(2) y はモジュール番号で、z はチャンネル番号です。y および z の詳細については I/O のアドレス指定構文 (*23* ページ参照) を参照してください。
(3) 入力ビットに書き込むことはできませんが、強制書き込みすることはできます。

例

I/O アドレス指定の例を次の表に示します。

I/O オブジェクト	説明
%I0.5	コントローラーのデジタル入力チャンネル番号 5 (内蔵 I/O はモジュール番号 0)。
%Q3.4	アドレス 3 にある拡張モジュールのデジタル出力チャンネル番号 4 (拡張モジュール I/O)。
%IW0.1	コントローラーのアナログ入力 1 (内蔵 I/O)。
%QW2.1	アドレス 2 にある拡張モジュールのアナログ出力 1 (拡張モジュール I/O)。
%IWS0.101	ロジックコントローラー第 1 カートリッジの入力チャンネル 1 のアナログ入力チャンネルステータス。
%QWS1.1	アドレス 1 にある拡張モジュールの出力チャンネル 1 のアナログ出力チャンネルステータス (拡張モジュール I/O)。

ワードオブジェクト

概要

16 ビットワードの形式でアドレス指定されたワードオブジェクトは、データメモリーに格納され -32768 ～ 32767 の整数値を含むことができます (**高速カウンター (FC)** ファンクションブロックは 0 ～ 65535 です)。

ワードオブジェクトの例
- 即値
- ワードメモリー (%MWi)
- ワード型定数 (%KWi)
- I/O 交換ワード (%IWi, %QWi, %IWSi, %QWSi)
- システムワード (%SWi)
- ファンクションブロック (設定およびランタイムデータ)

有効なオブジェクトの範囲は、0 からアプリケーションで使用されている最大数までです (ロジックコントローラーのプログラミングガイドを参照してください)。

例えば、ワードメモリーのアプリケーションの最大数が %MW9 の場合、%MW0 から %MW9 が割り当てられるスペースとなります。この例の %MW10 は無効で、内部的または外部的にアクセスすることはできません。

構文

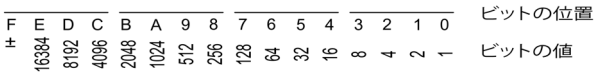
ワードメモリー、ワード型定数、およびシステムワードのアドレス指定は次の形式を使用します。

次の表で、アドレス指定形式の要素について説明します。

グループ	項目	説明
シンボル	%	パーセント記号は常に内部アドレスの先頭につきます。
オブジェクトタイプ	M	ワードメモリーはプログラムの実行中に値を格納します。
	K	ワード型定数は定数値または英数字メッセージを格納します。 SoMachine Basic のみ内容の書き込みまたは編集が可能です。
	S	システムワードはコントローラーのステータスと制御情報を示します。
形式	W	16 ビットワード。
オブジェクトインスタンス識別子	i	メモリー内の連続したインスタンスを表すオブジェクトの識別子。オブジェクトの最大数は、使用可能なメモリーに設定されたオブジェクトの数によって異なります。使用可能なメモリーの最大値については、ロジックコントローラーの *プログラミングガイド* を参照してください。

形式

ワードまたは値の内容は次の規則に従って 16 ビットバイナリコード (2 の補数形式) でユーザーメモリーに格納されます。



符号付き 2 進表記では、ビット 15 が符号化値の符号に割り当てられます。
- ビット 15 は 0 に設定されると、ワードの内容は正の値です。
- ビット 15 は 1 に設定されると、ワードの内容は負の値です (負の値は 2 の補数ロジックで表されます)。

ワードおよび即値 (符号なし整数の例外リスト (*26* ページ参照) を参照) は、次の形式で入力または取得できます。
- 10 進数
  最小 : -32768、最大 : 32767 (例えば、1579)
- 16 進数
  最小 : 16#0000、最大 : 16#FFFF (例えば、16#A536)
  代替構文 : #A536

- ASCII 形式の規則は次のとおりです。
  - ファクションは常に最上位バイトを最初に読み込みます。
  - [0 - 9] ([16#30 - 16#39]) の範囲にない ASCII 文字は終了文字とみなされます。ただしマイナス記号 '-' (16#2D) が先頭の文字の場合を除きます。
  - オーバーフロー (> 32767 または < -32768) の場合、システムビット %S18 (算術オーバーフローまたは検出されたエラー) が 1 に設定され、32768 または -32768 が返されます。
  - オペランドの最初の文字が終了文字の場合、値 0 が返され、ビット %S18 が 1 に設定されます。

  例えば、"HELLO":
  - %MW0:="HE"
  - %MW1:="LL"
  - %MW2:="0"

例外リスト

符号なし整数オブジェクトの値の範囲を次の表に示します。

オブジェクト	値
%SW	0...65535
%FC.V および %FC.P	0...65535
%FC.VD および %FC.PD	0...4294967295
%HSC.V、%HSC.P、%HSC.SO、%HSC.S1、および %HSC.C	0...65535
%HSC.DV、%HSC.PD、%HSC.SOD、%HSC.S1D、および %HSC.CD	0...4294967295
%HSC.T	100...1000
%PWM.P	0...32767
%PWM.R	0...100
%PLS.P	0...32767
%PLS.N	0...32767
%PLS.ND	0...2147483647

例外リストのオブジェクト以外のすべてのデータは、次の値の範囲を持ちます。
- ワード : -32768...32767
- ダブルワード : -2147483648...2147483647

説明

この表ではワードオブジェクトについて説明します。

ワード	説明	アドレスまたは値	書き込みアクセス (1)
即値	16 ビットワードと同じ形式の整数値です。これらのワードに値を割り当てることができます。	-	不可
	10 進法 (10 進数)	-32768 ～ 32767	
	16 進法 (16 進数)	16#0000 ～ 16#FFFF	
メモリー	データメモリーで演算中に値を格納するための作業用のワードとして使用されます。	%MWi	可
定数	定数または英数字メッセージを格納します。オンラインモードおよび設定中でも SoMachine Basic を使用して書き込みおよび編集ができます。(*(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* 参照)	%KWi	オンラインモードおよび設定中でもワード型定数プロパティを使用した場合は、可。(*(EcoStruxure Machine Expert - Basic, オペレーティングガイド)*)
(1) プログラムによる書き込みまたはアニメーションテーブルを使用した書き込み			

ワード	説明	アドレスまたは値	書き込みアクセス (1)
システム	これらの 16 ビットワードには複数のファンクションがあります。 ● ワード %SWi を読み込むことで、コントローラーから直接受け取るデータへのアクセスができます。 ● アプリケーションの演算を実行します (例えば、スケジュールブロックの調整など)。	%SWi	i により異なる
ファンクションブロック	これらのワードは現在のパラメーターまたはファンクションブロックの値に対応しています。	%TM2.P、%Ci.P など。	可
(1) プログラムによる書き込みまたはアニメーションテーブルを使用した書き込み			

使用可能なオブジェクトの最大数は、ロジックコントローラーによって異なります。オブジェクトの最大数については、ロジックコントローラーの *プログラミングガイド* を参照してください。

例

ワードオブジェクトのアドレス指定の例を次の表に示します。

ワードオブジェクト	説明
%MW15	ワードメモリー番号 15
%KW26	ワード型定数番号 26
%SW30	システムワード番号 30

ワードオブジェクトからビットを抽出します

次のワードオブジェクトから 16 ビットのうち 1 ビットを抽出する方法を次に示します。

ワードオブジェクト	アドレスまたは値	書き込みアクセス (1)
メモリー	%MWi:Xk	可
システム	%SWi:Xk	i により異なる
定数	%KWi:Xk	不可
入力値	%IWy.z:Xk(2)	不可
出力値	%QWy.z:Xk(2)	可
入力ステータス	%IWSy.z:Xk(2)	不可
出力ステータス	%QWSy.z:Xk(2)	可
(1) プログラムによる書き込みまたはアニメーションテーブルを使用した書き込み **(2)** I/O ワードオブジェクトについては I/O オブジェクトのアドレス指定 (*23* ページ参照) を参照してください。 **Xk は、**ワードオブジェクトから抽出されるビット番号を示します。例えば、%MW0.X3 の場合、ワードメモリー %MW0 の 3 番目に格納されたビットが抽出されます。		

浮動小数点およびダブルワードオブジェクト

概要

浮動小数点オブジェクトは実数です。小数部分がある数値です (例えば、3.4E+38、2.3、および 1.0)。

ダブルワードはデータメモリーに 4 バイトで格納され、2 の補数 -2147483648 ~ +2147483647 を含みます。

すべてのロジックコントローラーで浮動小数点の演算およびダブルワードの演算は対応していません。

詳細は、ロジックコントローラーの *プログラミングガイド* を参照してください。

浮動小数点形式と値

浮動形式は IEEE STD 734-1985 (IEC 559 相当) です。ワードの長さは 32 ビットで単精度浮動小数点数に対応します。

次の表に浮動小数点値の形式を示します。

ビット 31 :	ビット {30...23}	ビット {22...0}
指数の符号	指数	仮数

表現精度は 2...24 で浮動小数点数を表示します。小数点以下 6 桁以上を表示する必要はありません。

注記：値 1285 は値全体として解釈されます。浮動小数点値として認識されるためには、次のように記述してください。1285.0

浮動小数点オブジェクトの算術ファンクションの範囲制限

この表では、浮動小数点オブジェクトの算術ファンクションの範囲制限について説明します。

算術ファンクション		**範囲制限と無効な演算**	
タイプ	**構文**	**NaN (数字以外)**	**無限**
オペランドの平方根	SQRT(x)	x < 0	x> SQRT(3.402824E+38) は取得可能な最大数です
実数による整数のべき乗 EXPT(%MF,%MW)	EXPT(y, x) (例えば、x^y = %MW^%MF)	x < 0 および y = 分数	X ^ Y > 3.402824E+38
基数 10 の対数	LOG(x)	x < 0	x の最大値が得られるまで計算が可能 (3.402824E+38)
自然対数	LN(x)	x < 0	x の最大値はありません。LN(3.402824E+38) は取得可能な最大数です
自然指数	EXP(x)	実際の範囲に制限はありません	x > 88.72283 x < -103.973 の場合、結果は 0

有効性チェック

結果が有効範囲内にない場合、システムビット %S18 は 1 に設定されます。

ステータスワード %SW17 は浮動小数点演算で検出されたエラーの原因を示します。

ワード %SW17 の各ビット

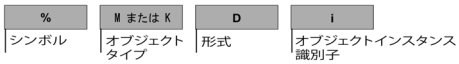
%SW17:X0	無効な演算。結果が数値ではありません (NaN)。
%SW17:X1	予約済
%SW17:X2	ゼロ除算。結果が無効です (無限大またはマイナス無限大)。
%SW17:X3	+3.402824E+38 よりも絶対値が大きい結果は無効です (無限大またはマイナス無限大)。
%SW17:X4 to X15	予約済

このワードはコールドスタート後にシステムによって 0 にリセットされます。再使用の目的でプログラムによってリセットすることもできます。

構文

浮動小数点メモリーおよび浮動小数点定数のオブジェクトのアドレス指定は次の形式を使用します。

ダブルワードメモリーおよびダブルワード定数のオブジェクトのアドレス指定は次の形式を使用します。



次の表で、アドレス指定形式の要素について説明します。

グループ	項目	説明
シンボル	%	パーセント記号は常に内部アドレスの先頭につきます。
オブジェクトタイプ	M	メモリーオブジェクトはプログラムの実行中に中間値を格納します。
	K	定数は、定数値または英数字メッセージを格納するために使用されます (ダブルワードの場合のみ)。
形式	F	32 ビット浮動小数点オブジェクト
	D	32 ビットダブルワードオブジェクト
オブジェクトインスタンス識別子	i	メモリー内のオブジェクトのインスタンス (連続した位置) を表す識別子。オブジェクトの最大数については、ロジックコントローラーの *プログラミングガイド*を参照してください。

浮動小数点オブジェクトおよびダブルワードオブジェクトの説明

浮動小数点オブジェクトとダブルワードオブジェクトについて説明します。

オブジェクトの型	説明	アドレス	書き込みアクセス
即値	32 ビットオブジェクトと同じ形式の整数 (ダブルワード) または 10 進数 (浮動小数点)。	-	不可
浮動小数点メモリー	データメモリーで演算中に値を格納するために使用されるオブジェクト。	%MFi	可
ダブルワードメモリー		%MDi	可
フロート定数値	定数の格納に使用します。	%KFi	設定中およびオンラインモード中にワード型定数プロパティ *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を使用した場合は、可。
ダブル定数		%KDi	設定中およびオンラインモード中にワード型定数プロパティ *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を使用した場合は、可。

注記：最大オブジェクト数は、ロジックコントローラーによって決まります。詳細については、ご使用のハードウェアプラットフォームのプログラミングガイドを参照してください。

例

浮動小数点およびダブルワードオブジェクトのアドレス指定の例を次の表に示します。

オブジェクト	説明
%MF15	浮動小数点メモリーオブジェクト番号 15
%KF26	浮動小数点定数オブジェクト番号 26
%MD15	ダブルワードメモリー番号 15
%KD26	ダブルワード型定数番号 26

オブジェクト間の重複の可能性

シングルワード、ダブルワード、およびフローティングワードは、1 つのメモリー領域のデータスペースに格納されます。フローティングワード %MFi およびダブルワード %MDi は、シングルワードの %MWi と %MWi+1 に相当します。フローティングワード %MFi の最下位ビットはシングルワード %MWi に含まれ、フローティングワード %MFi の最上位ビットはシングルワード %MWi+1 に含まれます。

次の表で、フローティングワードメモリーとダブルワードメモリーの重複を示します。

フローティングとダブル	奇数アドレス	ワードメモリー
%MF0 / %MD0		%MW0
	%MF1 / %MD1	%MW1
%MF2 / %MD2		%MW2
	%MF3 / %MD3	%MW3
%MF4 / %MD4		%MW4
	…	%MW5
…		…
	%MFi / %MDi	%MWi
%MFi+1 / %MDi+1		%MWi+1

次の表で、フローティングワード型定数とダブルワード型定数の重複を示します。

フローティングとダブル	奇数アドレス	ワードメモリー
%KF0 / %KD0		%KW0
	%KF1 / %KD1	%KW1
%KF2 / %KD2		%KW2
	%KF3 / %KD3	%KW3
%KF4 / %KD4		%KW4
	…	%KW5
…		…
	%KFi / %KDi	%KWi
%KFi+1 / %KDi+1		%KWi+1

例：

%MF0 は %MW0 と %MW1 に相当します。%KF43 は %KW43 と %KW44 に相当します。

構造体オブジェクト

概要

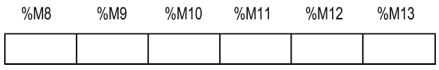
構造体オブジェクトは隣接するオブジェクトを組み合わせたものです。SoMachine Basic は次の構造体オブジェクトに対応しています。

- ビット文字列
- ワードテーブル
- ダブルワードテーブル
- フローティングワードテーブル

ビット文字列

ビット文字列は同じタイプと長さ (L) のオブジェクトビットが連なったものです。ビット文字列は、バイト境界から開始して参照されます。

例：ビット文字列 %M8:6



注記：%M8:6 は有効です (8 は 8 の倍数です) が、%M10:16 は無効です (10 は 8 の倍数ではありません)。

ビット文字列は、代入命令 (*42* ページ参照) で使用できます。

使用可能なビットのタイプ

ビット文字列として使用可能なビットのタイプ

タイプ	アドレス	書き込みアクセス
デジタル入力ビット	%I0.0:L または %I1.0:L(1)	不可
デジタル出力ビット	%Q0.0:L または %Q1.0:L(1)	可
システムビット	%Si:L i は 8 の倍数	i により異なる
グラフセステップビット	%Xi:L i は 8 の倍数	可 (プログラムによる)
ビットメモリー	%Mi:L i は 8 の倍数	可

(1) I/O ビット 0 ～ 16 のみをビット文字列に読み込むことができます。24 または 32 の I/O チャンネルを持つロジックコントローラーの場合、ビット 16 以降はビット文字列に読み込むことができません。

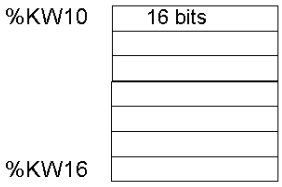
L 構造体オブジェクト (ビット文字列、ワードテーブル、ダブルワードテーブル、およびフローティングワードテーブル) の長さを表します。

ビット数は、ロジックコントローラーによって決まります。詳細については、ご使用のハードウェアプラットフォームのプログラミングガイドを参照してください。

ワードテーブル

ワードテーブルは同じタイプと長さ (L、最大値は 255) のワードが連なったものです。

例：ワードテーブル %KW10:7



ワードテーブルは代入命令 (*42* ページ参照) で使用できます。

使用可能なワードのタイプ

ワードテーブルとして使用可能なワードのタイプ

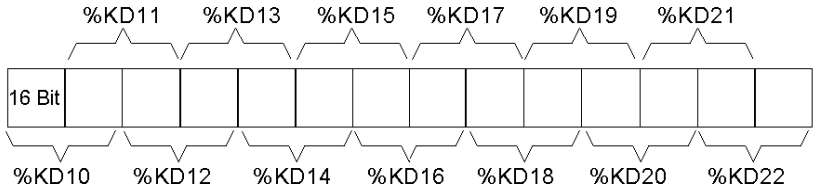
タイプ	アドレス	書き込みアクセス
ワードメモリー	%MWi:L	可
ワード型定数	%KWi:L	不可
システムワード	%SWi:L	i により異なる

ワード数は、ロジックコントローラーによって決まります。詳細については、ご使用のハードウェアプラットフォームのプログラミングガイドを参照してください。

ダブルワードテーブル

ダブルワードテーブルは同じタイプと長さ (L、最大値は 255) のワードが連なったものです。

例 : ダブルワードテーブル %KD10:7



ダブルワードテーブルは代入命令 (*42* ページ参照) で使用できます。

使用可能なダブルワードのタイプ

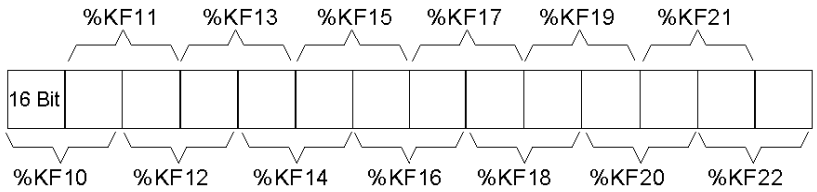
ダブルワードテーブルとして使用可能なワードのタイプ

タイプ	アドレス	書き込みアクセス
ワードメモリー	%MDi:L	可
ワード型定数	%KDi:L	不可

フローティングワードテーブル

フローティングワードテーブルは同じタイプと長さ (L、最大値は 255) のワードが連なったものです。

例 : 浮動小数点テーブル %KF10:7



浮動小数点テーブルは代入命令 (*42* ページ参照) で使用できます。

使用可能なフローティングワードのタイプ

フローティングワードテーブルとして使用可能なワードのタイプ

タイプ	アドレス	書き込みアクセス
ワードメモリー	%MFi:L	可
ワード型定数	%KFi:L	不可

インデックス付きオブジェクト

概要

インデックス付きオブジェクトは、インデックス付きオブジェクトアドレスを持つシングルワード、ダブルワード、または浮動小数点オブジェクトです。オブジェクトアドレス指定には 2 つ方法があります。

- 直接アドレス指定
- インデックス付きアドレス指定

直接アドレス指定

オブジェクトの直接アドレスは、プログラムの記述時に設定および定義されます。

例 : %M26 は直接アドレス 26 のビットメモリーです。

インデックス付きアドレス指定

オブジェクトのインデックス付きアドレスは、オブジェクトの直接アドレスにインデックスを追加することにより、オブジェクトのアドレスを変更できます。インデックスの内容がオブジェクトの直接アドレスに追加されます。インデックスはワードメモリー %MWi で定義されます。

例 : ワード %MW108[%MW2] のアドレスは直接アドレス 108 にワード %MW2 の内容を加算したアドレスです。

ワード %MW2 の値が 12 の場合、%MW108[%MW2] への書き込みは %MW120 (108+12) への書き込みと同等です。

インデックス付きアドレス指定に使用可能なオブジェクト

インデックス付きアドレス指定のオブジェクトとして使用可能なタイプについて次の表に示します。

タイプ	アドレス	書き込みアクセス
ワードメモリー	%MWi[MWj]	可
ワード型定数	%KWi[%MWj]	不可
ダブルワードメモリー	%MDi[MWj]	可
ダブルワード型定数	%KDi[%MWj]	不可
浮動小数点メモリー	%MFi[MWj]	可
浮動小数点型定数	%KFi[%MWj]	不可
i メモリー内のオブジェクトのインスタンス (連続した位置) を表すオブジェクトインスタンス識別子。オブジェクトの最大数については、ロジックコントローラーのプログラミングガイドを参照してください。 **j** オブジェクトの直接アドレスに追加されるインデックスオブジェクトのオブジェクトインスタンス識別子。		

インデックス付きオブジェクトは、代入命令 (*54* ページ参照) および 比較命令 (*50* ページ参照) で使用できます。

このタイプのアドレス指定は、プログラム内のインデックスオブジェクトの内容を変更することによって、同じタイプ (ワードメモリーや定数など) の一連のオブジェクトを連続してスキャンすることができます。

インデックスオーバーフローシステムビット %S20

インデックス付きオブジェクトのアドレスが、同じタイプのオブジェクトを含むメモリー領域の範囲を超えると、インデックスのオーバーフローが発生します。以下にまとめると

- オブジェクトのアドレスとインデックスの内容の合計が 0 未満。
- オブジェクトのアドレスとインデックスの内容の合計が、アプリケーションで直接参照されている最大ワードよりも大きい。

インデックスがオーバーフローした場合、システムビット %S20 は 1 に設定され、オブジェクトにはインデックス値 0 が割り当てられます。

注記 : ユーザーの責任においてオーバーフローの監視を行ってください。使用するプログラムで処理するために %S20 を読み込んでください。それが 0 にリセットされることを確認してください。

%S20 (初期状態 = 0):

- インデックスオーバーフロー : コントローラーによって 1 に設定されます。
- オーバーフローの確認 : インデックスを変更後、プログラムで 0 に設定します。

⚠ 警告

装置の意図しない動作

- 算術演算で使用するオペランドの有効性を確認するためのプログラミング命令を書いてください。
- 算術演算で異なるデータ型のオペランドを使用しないでください。
- 無効な算術結果を表示するために割り当てられたシステムビットを常に監視してください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

ファンクションブロックオブジェクト

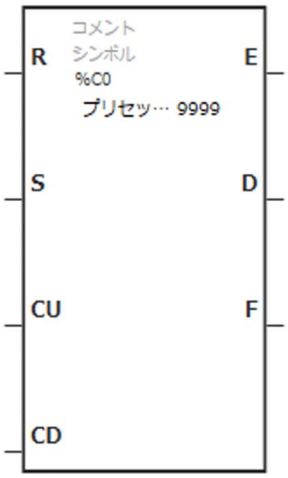
概要

ファンクションブロックは再利用可能なオブジェクトです。1 つ以上の入力値を受けつけ、1 つ以上の出力値を返します。ファンクションブロックは常にインスタンス (専用の名前と変数を持つファンクションブロックのコピー) を介して呼び出されます。各ファンクションブロックインスタンスはある呼び出しから別の呼び出しへの保持状態 (出力および内部変数) を持ちます。

注記：ファンクションブロック (%FC、%HSC、%PLS、および %PWM) とステータスアラームは、コントローラーのサイクルとは関係なく入力および出力 (設定に影響を受ける　%I0.x　と %Q0.x) を直接運転します。イメージビット (%I0.x および %Q0.x) はコントローラーによって更新されないため、これらの入出力ビットはユーザープログラムで直接使用はできません。また、アニメーションテーブルで使用しているイメージビットの入出力も現在の状態を示すことはできません。

例

次の図は、カウンターファンクションブロックを示しています。



ビットオブジェクト

ビットオブジェクトはファンクションブロックの出力に対応します。これらのビットは、次のいずれかの方法を使用してブールテスト命令でアクセスできます。

- 可逆プログラミング (*119* ページ参照) でブロックに配線されている場合は直接 (例えば、LD E)。
- ブロックタイプを指定 (例えば、LD %Ci.E)。

入力は命令形式でアクセスできます。

ワードオブジェクト

Word オブジェクトは次のような指定されたパラメーターと値に対応します。

- ブロック設定パラメーター : プログラムによってアクセス可能なパラメーター (事前選択パラメーターなど)、プログラムによってはアクセスできないパラメーター (Time base など) もあります。
- 現在の値 : 例えば、カウントの現在値 %Ci.V

ダブルワードオブジェクト

ダブルワードオブジェクトは、高速カウンター (%FC)、高速カウンター (%HSC)、パルスジェネレーター (%PLS, %PWM) などのシステムファンクションの実行中にロジックコントローラーの計算能力を向上させます。

ファンクションブロックで使用される 32 ビットのダブルワードオブジェクトのアドレス指定は、文字 D を標準ワードオブジェクトの構文に追加するだけです。

標準形式とダブルワード形式で高速カウンター (FC) の現在値を指定する方法を次の例に示します。

- %FCi.V は標準形式の高速カウンター (FC) の現在値です。
- %FCi.VD はダブルワード形式の高速カウンター (FC) の現在値です。

第 3 章
命令

この章について

この章には次のセクションが含まれています。

3.1
ブール演算

このセクションの目的

このセクションでは、ブール演算命令の概要について説明します。

このセクションについて

このセクションには次の項目が含まれています。

ブール命令

概要

ブール命令はラダー図言語の要素と比較できます。これらの命令を以下の表に示します。

項目	演算子	命令の例	説明
テスト要素	ロード (LD) 命令は、ラダー図の電源レールに接続されている最初の A 接点に相当します。論理 AND および OR 命令は、ラダー図の電源レールに接続されている最初の接点の後の A 接点に相当します。	LD %I0.0	ビット %I0.0 が状態 1 のとき、接点は閉じています。
アクション要素	ストア (ST) 命令はコイルに相当します。	ST %Q0.0	関連するビットオブジェクトは、ビットアキュムレーターの論理値をとります (直前の論理の結果)。

テスト要素のブール結果は、次の手順に示すようにアクション要素に適用されます。

ラング	命令
0	LD %I0.0 AND %I0.1 ST %Q0.0

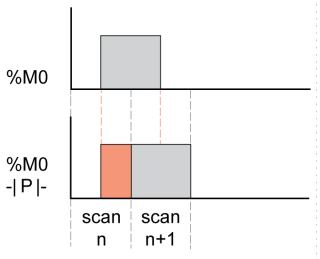
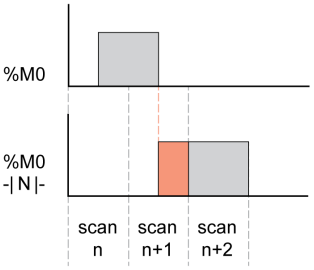
注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

コントローラー入力のテスト

ブールテスト命令を使用してコントローラー入力の立上がりまたは立下がりを検出することができます。入力の状態が「スキャン n-1」と現在の「スキャン n」の間で変化したときに立上がりまたは立下がりが検出されます。現在のスキャン中は、このエッジの検出は保持されます。

エッジ検出

次の表に、立上がりおよび立下がりを検出するための命令とタイミングを示します。

エッジ	IL 演算	ラダー図	タイミングチャート
立上がり接点	LDR %M0		
立下り接点	LDF %M0		
 オブジェクトはエッジ検出に続く次のマスターサイクルの開始時にのみ更新されます。ビットメモリー (例えば、%M0) の状態変化は 1 スキャン遅れて表示されます。			

注記：立上がりおよび立下り接点は入力ビット (%I) およびビットメモリー (%M) オブジェクトでのみ使用できます。

RISING  および FALLING  ファンクション (*49* ページ参照) を使用してエッジ検出ができます。それらを使用するにはアプリケーションのファンクションレベル を **レベル 6.0** 以上に設定してください。

立上がり / 立下り接点と RISING / FALLING ファンクションの違いの 1 つはプログラムに適用されるスキャンです。

- LDR / LDF 命令は 1 スキャン遅れで立上がり / 立下がりを示します。
- RISING / FALLING ファンクションは同じサイクルで発生した立上がり / 立下りを示します。

立上がり検出

Load Rising Edge (LDR) 命令は、立上がり接点検出と同等です。立上がりは 0 から 1 への入力値の変化を検出します。

この例に示すように、立上がりを検出するには立上がり接点検出を使用します。

ラング	命令
0	LDR %I0.0

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

立下がり検出

Load Falling Edge (LDF) 命令は、立下がり接点検出と同等です。立下がりは 1 から 0 への入力値の変化を検出します。

この例に示すように、立下がりを検出するには立下がり接点検出を使用します。

ラング	命令
0	LDF %I0.0

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

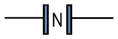
ロード演算子 (LD, LDN, LDR, LDF)

概要

ロード演算子 LD、LDN、LDR、および LDF は、A 接点、B 接点、立上がり接点、および立下り接点に対応します。LDR および LDF はロジックコントローラーの入力およびワードメモリーでのみ使用できます。

構文

次の表にロード演算子の種類ごとに相当するラダー図と使用できるオペランドを示します。

演算子	相当するラダー図	使用できるオペランド
LD		O/1 %I, %Q, %M, %S, %X, %BLK.x %IW:Xk, %QW:Xk, %IWS:Xk, %QWS:Xk, %MW:Xk, %SW:Xk, %KW:Xk
LDN		
LDR		%I, %M
LDF		

コーディングの例

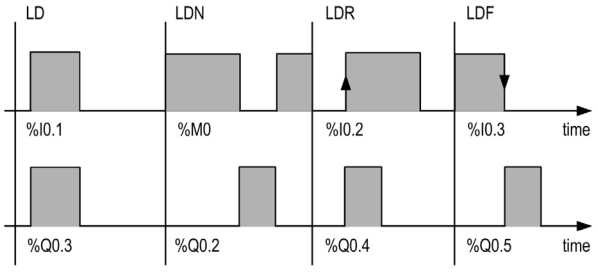
ロード命令の例 :

ラング	命令	
0	LD	%I0.1
	ST	%Q0.3
1	LDN	%M0
	ST	%Q0.2
2	LDR	%I0.1
	ST	%Q0.4
3	LDF	%I0.3
	ST	%Q0.5

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

タイミングチャート

次の図は、コーディング例によるコードタイミングと出力への影響を示しています。



注記：ビットメモリー (%M) の ビットエッジ検出は、マスタータスクスキャンの間で実行されます。

代入演算子 (ST, STN, R, S)

概要

代入演算子 ST、STN、S、および R は、正転、反転、コイルのセットおよびリセットを行います。

構文

次の表に代入演算子の種類ごとに相当するラダー図と使用できるオペランドを示します。

演算子	相当するラダー図	使用できるオペランド
ST	—( )	%Q, %M, %BLK.x %QW:Xk, %MW:Xk, %S(1), %SW:Xk(1)
STN	—(/)	
S	—(S)	%Q, %M, %S, %X, %BLK.x %QW:Xk, %MW:Xk, %SW:Xk(1)
R	—(R)	
(1) %S または %SW:Xk は読み取り専用オブジェクトではありません。		

コーディングの例

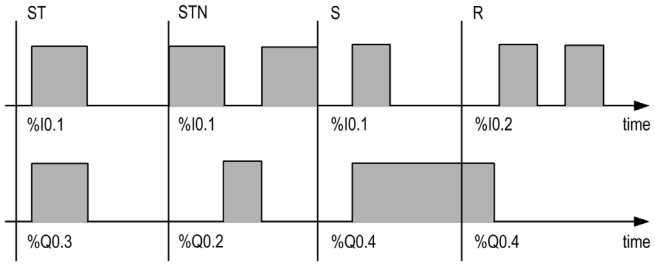
代入命令の例 :

ラング	命令
0	LD %I0.1 ST %Q0.3 STN %Q0.2 S %Q0.4
1	LD %I0.2 R %Q0.4

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

タイミングチャート

次の図は、コーディング例によるコードのタイミングと出力への影響を示しています。



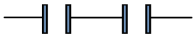
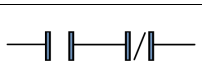
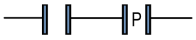
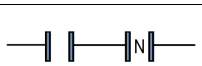
論理積演算子 (AND, ANDN, ANDR, ANDF)

概要

AND 演算子はオペランド (または、オペランドの反転、立上がりまたは立下がり) と直前の命令のブール結果の間の論理積演算を実行します。

構文

次の表に AND 演算子の種類ごとに相当するラダー図と使用できるオペランドを示します。

演算子	相当するラダー図	使用できるオペランド
AND		0/1 %I, %Q, %M, %S, %X, %BLK.x %IW:Xk, %QW:Xk, %IWS:Xk, %QWS:Xk, %MW:Xk, %SW:Xk, %KW:Xk
ANDN		
ANDR		%I, %M
ANDF		

コーディングの例

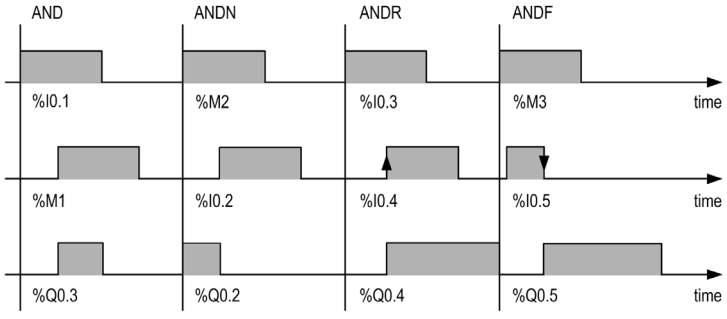
論理 AND 命令の例 :

ラング	命令	
0	LD	%I0.1
	AND	%M1
	ST	%Q0.3
1	LD	%M0
	ANDN	%I0.0
	ST	%Q0.2
2	LD	%I0.3
	ANDR	%I0.4
	S	%Q0.4
3	LD	%M3
	ANDF	%I0.5
	S	%Q0.5

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

タイミングチャート

次の図は、コーディング例によるコードのタイミングと出力への影響を示しています。



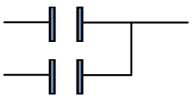
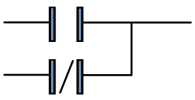
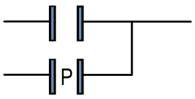
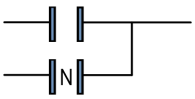
論理和演算子 (OR, ORN, ORR, ORF)

概要

OR 演算子はオペランド (または、オペランドの反転、立上がりまたは立下がり) と直前の命令のブール結果の間の論理和演算を実行します。

構文

次の表に OR 演算子の種類ごとに相当するラダー図と使用できるオペランドを示します。

演算子	相当するラダー図	使用できるオペランド
OR		O/1 %I, %Q, %M, %S, %X, %BLK.x %IW:Xk, %QW:Xk, %IWS:Xk, %QWS:Xk, %MW:Xk, %SW:Xk, %KW:Xk
ORN		
ORR		%I, %M
ORF		

コーディングの例

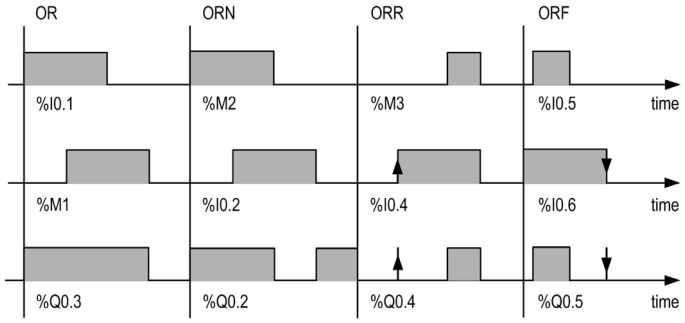
論理 OR 命令の例 :

ラング	命令	
0	LD	%I0.1
	OR	%M1
	ST	%Q0.0
1	LD	%I0.2
	ORN	%M2
	ST	%Q0.1
2	LD	%M0
	ORR	%I0.3
	S	%Q0.5
3	LDF	%I0.5
	ORF	%I0.6
	S	%Q0.0

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

タイミングチャート

次の図は、コーディング例によるコードのタイミングと出力への影響を示しています。



排他的論理和演算子 (XOR, XORN, XORR, XORF)

概要

XOR 演算子はオペランドと演算命令のブール結果の排他的論理和演算を実行します。

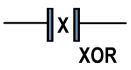
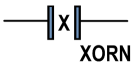
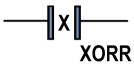
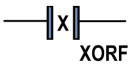
XORN 演算子はオペランドの逆数と直前の命令のブール結果の排他的論理和演算を実行します。

XORR 演算子はオペランドの立上がりと直前の命令のブール結果の排他的論理和演算を実行します。

XORF 演算子はオペランドの立下がりと直前の命令のブール結果の排他的論理和演算を実行します。

構文

次の表に XOR 演算子の種類と使用できるオペランドを示します。

演算子	相当するラダー図	使用できるオペランド
XOR		%I, %Q, %M, %S, %X, %BLK.x %IW:Xk, %QW:Xk, %IWS:Xk, %QWS:Xk, %MW:Xk, %SW:Xk, %KW:Xk
XORN		
XORR		%I, %M
XORF		

コーディングの例

XOR 命令の使用

ラング	命令
0	LD %I0.1 XOR %M1 ST %Q0.3

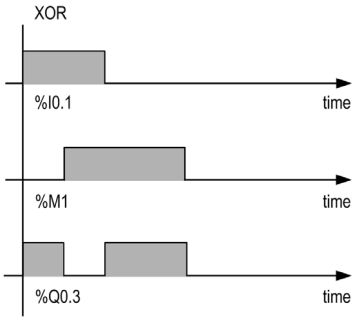
XOR 演算子の論理命令に相当します。

ラング	命令
0	LD %I0.1 ANDN %M1 OR(%M1 ANDN %I0.1) ST %Q0.3

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

タイミングチャート

次の図は、コーディング例によるコードのタイミングと出力への影響を示しています。

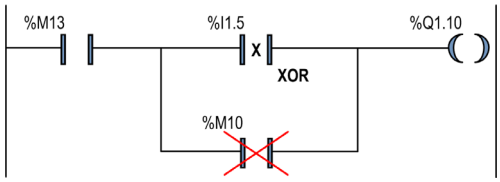


特殊な場合

禁止事項を次に示します。

- ラングの最初に XOR 接点
- その他のラダー図要素と XOR 接点の並列接続 (次の例を参照してください)。

例に示すように、要素を XOR 接点と並列接続すると、コンパイルエラーが発生します。



NOT 演算子 (N)

概要

NOT (N) 演算子は暗黙のオペランドを持ちます。つまり、結果はブール型アキュムレーターに格納されます。NOT はアキュムレーターの値を無効にします。

構文

N 演算子を以下の表に示します。

演算子	相当するラダー図	使用できるオペランド
N		適用できません

コーディングの例

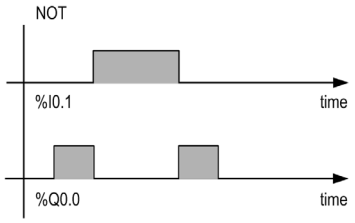
NOT 命令の例 :

ラング	命令
0	LD %I0.1 N ST %Q0.0

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

タイミングチャート

次の図は、コーディング例によるコードのタイミングと出力への影響を示しています。



立上がりおよび立下りファンクション (RISING、FALLING)

概要

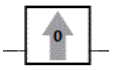
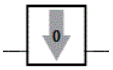
RISING および FALLING ファンクションは、直前の式の立上がりと立下りをそれぞれ評価します。

これらのファンクションには暗黙のオペランドがあります。つまり、ブール型アキュムレーターに格納される直前の式の結果です。

これらのファンクションはラダー言語のラングまたはブランチの最初の列に配置することはできません。また IL ラングの最初の命令にすることもできません。

注記: RISING および FALLING ファンクションを使用するにはアプリケーションのファンクションレベル *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を **レベル 6.0** 以上に設定してください。

構文

ファンクション	相当するラダー図	使用できるオペランド
RISINGn (1)		適用できません
FALLINGn (1)		適用できません
(1) n は、立上がりまたは立下りが挿入されるたびにインクリメントされる整数です。 この整数は次の場合に自動的に計算されます。 ● インデックスを定義しない場合。 ● 間違ったインデックスを入力した場合。 ● インデックスを削除した場合。 ● インデックスを変更した場合。		

コーディングの例

ファンクション	ラング	命令
RISING	0	LD %M0 RISING0 ST %Q0.0
FALLING	1	LD %I0.1 FALLING0 ST %Q0.7

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

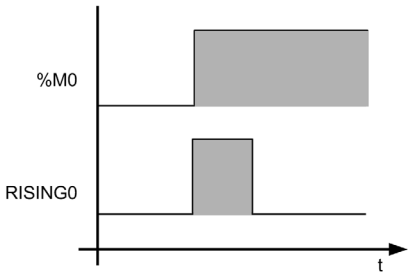
各タイプごとに最大 32 の命令を挿入できます。

次のいずれかの命令の直後に RISING および FALLING ファンクションを使用することはできません。

- AND
- ANDN
- OR
- ORN

タイミングチャート

1 つのマスタータスクのスキャン時間に対する上記の例のタイミングを次の図に示します。



比較命令

概要

比較演算子は、3 つの括弧の入れ子をもつオペランドを最大 5 つまで比較するために使用します。

比較命令の種類を以下の表に示します。

演算子	ファンクション
>	Op1 が Op2 より大きいかテストします
>=	Op1 が Op2 以上かをテストします
<	Op1 が Op2 より小さいかテストします
<=	Op1 が Op2 以下かをテストします
=	Op1 が Op2 と等しいかをテストします
<>	Op1 が Op2 と異なるかをテストします

構文

IL 構文は次のとおりです。**比較ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 比較式を挿入 (*18* ページ参照) できます。

比較命令の構文

演算子	構文
>, >=, <, <=, =, <>	LD [Op1 演算子 Op2] AND [Op1 演算子 Op2] OR [Op1 演算子 Op2]

次の表にオペランドの詳細を示します。

タイプ	Op1	Op2
ワード	%MWi, %KWi, %IW, %QWi, %SWi, %BLK.x	即値 ,%MWi, %KWi, %IW, %QW, %IWSi, %QWSi, %SWi, %BLK.x, %MWi[%MWi], %KWi[%MWi]
ダブルワード	%MDi, %KDi	即値 ,%MDi, %KDi, %MDi[%MWi], %KD[%MWi]
浮動小数点ワード	%MFi, %KFi	浮動小数点型即値 ,%MFi, %KFi, %MFi[%MWi], %KFi[%MWi]

注記：比較命令は括弧内で使用できます。

コーディングの例

比較は、命令 LD、AND、OR に続く角括弧の中で実行されます。リクエストされた比較が真の場合、結果は 1 です。

比較命令の例

ラング	命令
0	LD %I0.2 AND [%MW10>100] ST %Q0.3
1	LD %M0 AND [%MW20<%KW35] ST %Q0.4
2	LD %I0.2 OR [%MF30>=%MF40] ST %Q0.5

括弧内に比較命令を使用する例

ラング	命令
0	LD %M0 AND([%MF20>10.0] OR %I0.0) ST %Q0.1

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

3.2
数値処理

このセクションの目的

このセクションでは、数値処理の概要について説明します。

このセクションについて

このセクションには次の項目が含まれています。

数値演算の概要

概要

数値命令は一般に 16 ビットワードと 32 ビットダブルワードに適用されます。角括弧の間に書かれています。直前の論理演算の結果が真 (ブール型アキュムレーター = 1) の場合、数値命令が実行されます。直前の論理演算の結果が偽 (ブール型アキュムレーター = 0) の場合、数値命令が実行されず、オペランドも変更されません。

代入命令

概要

代入命令は、Op1 (オペランド 1) に Op2 (オペランド 2) を代入します。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

代入命令の構文

演算子	構文
:=	[Op1:= Op2] Op1 は Op2 の値をとります

代入命令を使用できるものは以下になります。
- ビット文字列
- ワード
- ダブルワード
- フローティングワード
- ワードテーブル
- ダブルワードテーブル
- フローティングワードテーブル
- パルストレイン出力オブジェクト

ビット文字列の代入

概要

次のビット文字列に対して実行できます。
- ビット文字列からビット文字列へ (例 1)
- ビット文字列からワード (例 2) またはダブルワード (インデックス付き)
- ワードまたはダブルワード (インデックス付き) からビット文字列 (例 3)
- 即値からビット文字列

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

ビット文字列割り当ての構文

演算子	構文
:=	[Op1:= Op2] Op1 は Op2 の値をとります

次の表にオペランドの詳細を示します。

タイプ	Op1	Op2
ワード、ダブルワード	%MWi,%QWi, %SWi %MWi[%MWi], %MDi, %MDi[%MWi] %Mi:L, %Qi:L, %Si:L, %Xi:L %TMi.P, %Ci.P, %Ri.I, %Ri.O, %FCi.P, %PLSi.P, %PWMi.P %Ci.PD, %FCi.PD	即値 , %MWi, %KWi, %IW, %QWi, %IWSi, %QWSi, %SWi,%BLK.x, %MWi[%MWi], %KWi[%MWi], %MDi[%MWi], %KDi[%MWi], %Mi:L,%Qi:L, %Si:L, %Xi:L, %Ii:L %TMi.P, %Ci.P, %Ri.I, %Ri.O, %FCi.P, %PLSi.P, %PWMi.P %Ci.PD, %FCi.PD

注記：略語 %BLK.x (例えば、%C0.P) は、任意のファンクションブロックワードを記述するために使用されます。

構造

ビット文字列の代入の例

ラング	命令
0	LD 1 [%Q0.0:8:=%M64:8]
1	LD %I0.2 [%MW100:=%M0:16]
2	LDR %I0.3 [%MW104:16:=%KW0]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

使用規則 :
- ビット文字列からワードへの代入の場合、文字列内のビットは右からワードに送信されます (文字列の第 1 ビットからワードのビット 0 へ)。送信にかかわらないワードビット (長さ ≦ 16) は 0 に設定されます。
- ワードからビット文字列への代入の場合、ワードビットは右側から送信されます (ワードビット 0 から文字列の第 1 ビットへ)。

ワードの代入

概要

次のワードとダブルワードで実行できます。
- ワード (インデックス付き) から ワード (例えば、2) (インデックス付きまたは無)
- ダブルワード (インデックス付き) からダブルワード (インデックス付きまたは無)
- 即値からワード (例 3) またはダブルワード (インデックス付きまたは無)
- ビット文字列からワードまたはダブルワード
- 浮動小数点 (インデックス付きまたは無) から浮動小数点 (インデックス付きまたは無)
- ワードまたはダブルワードからビット文字列
- 浮動小数点即値から浮動小数点へ (インデックス付きまたは無)

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

ワードの代入の構文

演算子	構文
:=	[Op1:= Op2] Op1 は Op2 の値をとります

次の表にオペランドの詳細を示します。

タイプ	Op1	Op2
ワード、ダブルワード、ビット文字列	%BLK.x, %MWi, %QWi, %SWi %MWi[MWi], %MDi, %MDi[%MWj], %Mi:L, %Qi:L, %Si:L, %Xi:L	即値 , %MWi, %KWi, %IW, %QWi, %IWSi, QWSi, %SWi, %MWi[MWi], %KWi[MWi], %MDi, %MDi[%MWj], %KDi, %KDi[MWj], %Mi:L, %Qi:L, %Si:L, %Xi:L, %Ii:L
浮動小数点	%MFi, %MFi[%MWj]	浮動小数点型即値 , %MFi, %MFi[%MWj], %KFi, %KFi[%MWj]

注記：略語 %BLK.x (例えば、R3.I) は、任意のファンクションブロックワードを記述するために使用されます。ビット文字列 %Mi:L、%Si:L、および %Xi:L の場合、ビット文字列の先頭のベースアドレスは 8 (0、8、16 ・・・96 ・・・) の倍数でなければなりません。

構造

ワードの代入の例

ラング	命令
0	LD 1 [%SW112:=%MW100]
1	LD %I0.2 [%MW0[%MW10]:=%KW0[%MW20]]
2	LD %I0.3 [%MW10:=100]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

整数の算術演算子

概要

算術演算子は、2 つの整数オペランドまたは 1 つの整数オペランドで算術演算をするために使用します。

算術命令の種類を以下の表に示します。

演算子	ファンクション
+	2 つのオペランドを加算
-	2 つのオペランドを減算
*	2 つのオペランドを乗算
/	2 つのオペランドを除算
REM	2 つのオペランドの除算の余り
SQRT	オペランドの平方根
INC	オペランドのインクリメント
DEC	オペランドのデクリメント
ABS	オペランドの絶対値

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

算術命令の構文

演算子	構文
+, -, *, /, REM	[Op1:= Op2 演算子 Op3]
INC, DEC	[演算子 Op1]
SQRT (1)	[Op1:= SQRT(Op2)]
ABS (1)	[Op1:= ABS(Op2)]

次の表にオペランドの詳細を示します。

タイプ	Op1	Op2 および Op3(1)
ワード	%MWi, %QWi, %SWi, %BLK.x(2)	即値 , %MWi, %KWi, %IWi, %QWi, %IWSi, %QWSi, %SWi, %BLK.x(2)
ダブルワード	%MDi, %BLK.x	即値 , %MDi, %KDi, %BLK.x(2)

(1) この演算子では Op2 として即値は使用できません。ABS ファンクションはダブルワード (%MD および %KD) と浮動小数点 (%MF および %KF) でのみ使用できます。そのため、OP1 と OP2 はダブルワードまたは浮動小数点でなければなりません。
(2) %BLK.x はすべてのブロックオブジェクトを表します。

構造

算術命令の例

ラング	命令
0	LD %M0 [%MW0:=%MW10+10]
1	LD %I0.2 [%MW0:=SQRT(%MW10)]
2	LDR %I0.3 [%MW10:=32767]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

特殊な場合

加算

- ワード演算中のオーバーフロー
  演算結果がワードの容量を超える場合、ビット %S18 (オーバーフ が) が 1 に設定され、結果は無効です (アプリケーション例 (*58* ページ参照) のラング 1 を参照)。ユーザープログラムはビット %S18 を管理します。

注記：ダブルワードの範囲は -2147483648 と 2147483647 です。

乗算

- 演算中のオーバーフロー
  演算結果がワードの容量を超える場合、ビット %S18 (オーバーフロー) は 1 に設定され、結果は無効です。

除算 / 余り

- ゼロ除算
  0 で割る除算は不可能です。システムビット %S18 は 1 に設定されます。正しくない結果になります。
- 演算中のオーバーフロー
  除算の商がワードの容量を超える場合、ビット %S18 は 1 に設定されます。

平方根の計算

- 演算中のオーバーフロー
  平方根の計算は正の値に対してのみ実行されます。そのため、結果は常に正です。平方根のオペランドが負の場合、システムビット %S18 は 1 に設定され、正しくない結果になります。

検出される数学的なエラーの中には、アプリケーションの実行に重大な影響を与える場合があります。お客様の責任においてこれらの潜在的なエラーを監視し、このようなエラーが検出された場合にアプリケーションの実行を適切に制御するための命令をプログラムしてください。これらの検出されるエラーの影響は、設定、使用機器、および潜在的なエラーが検出される前後に実行されるプログラム命令により異なります。

⚠ 警告

装置の意図しない動作

- 算術演算で使用するオペランドの有効性を確認するためのプログラミング命令を書いてください。
- 算術演算で異なるデータ型のオペランドを使用しないでください。
- 無効な算術結果を表示するために割り当てられたシステムビットを常に監視してください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

注記：ユーザープログラムは、システムビット %S17 および %S18 の管理をします。これらはコントローラーによって 1 に設定されるため、再利用できるようにプログラムでリセットしてください (前のページの例を参照してください)。

使用例

加算中のオーバーフロー

ラング	命令
0	LD %M0 [%MW0:=%MW1+%MW2]
1	LDN %S18 [%MW10:=%MW0]
2	LD %S18 [%MW10 :=32767]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

%MW1 = 23241 および %MW2 = 21853 の場合、結果は (45094) となりますが、これは符号付き 16 ビットワード 1 つでは表現できません。そのため、ビット %S18 は 1 に設定され %MW0 の値 (-20442) は正しくありません。この例では、結果が 32767 より大きい場合、その値は 32767 に固定されています。

論理命令

概要

論理演算子は、2 つのワードオペランドの論理演算を実行するために使用できます。ただし、論理演算子 NOT のオペランドはワード 1 つとなります。

論理命令の種類を以下の表に示します。

命令	ファンクション
AND	2 つのオペランドの論理積 (ビット単位)
OR	2 つのオペランドの論理論理和 (ビット単位)
XOR	2 つのオペランドの排他的論理排他的論理和 (ビット単位)
NOT	オペランドの論理補数 (ビット単位)

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

論理命令の構文

演算子	構文	Op1	Op2 および Op3
AND, OR, XOR	[Op1:= Op2 演算子 Op3]	%MWi, %QWi, %SWi, %BLK.x	即値 (1), %MWi, %KWi, %IWi, %QWi, %IWSi, %QWSi, %SWi, %BLK.x
NOT	[Op1:=NOT(Op2)]		
(1) NOT の場合、Op2 として即値は使用できません。			

構造

論理命令の例

ラング	命令
0	LD %M0 [%MW0:=%MW10 AND 16#00FF]
1	LD 1 [%MW0:=%KW5 OR %MW10]
2	LD %I0.3 [%MW102:=NOT(%MW100)]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

使用例

論理積命令

[%MW15:=%MW32 AND %MW12]

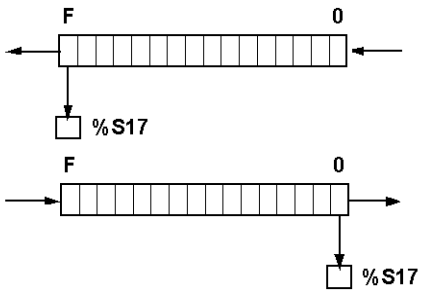
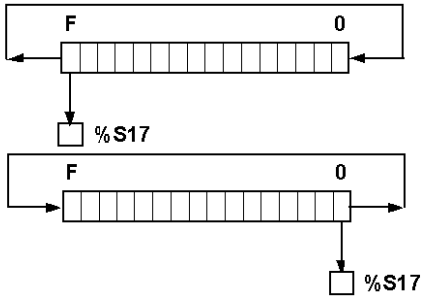
%MW32 = 0001 1011 (2 進数) (27 (10 進数)) および %MW12 = 0011 0110 (2 進数) (54 (10 進数)) の場合、結果は %MW15 = 0001 0010 (2 進数) (18 (10 進数)) となります。

シフト命令

概要

シフト命令は、オペランドのビットを指定された数だけ右または左に移動します。

シフト命令の種類を以下の表に示します。

命令	ファンクション	
論理シフト		
SHL (op2, n)	左へ n 位置論理シフト	
SHR (op2, n)	右へ n 位置論理シフト	
ローテートシフト		
ROL (op2, n)	左へ n 位置ローテートシフト	
ROR (op2, n)	右へ n 位置ローテートシフト	
n 整数型即値は : ❍ ワード :1...16 ❍ ダブルワード :1...32		

注記：システムビット %S17 は最後に排出されたビットの値を示します

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

シフト命令の構文

演算子	構文
SHL、SHR	[Op1:= 演算子 (Op2,n)]
ROL、ROR	
n 整数型即値は : ❍ ワード :1...16 ❍ ダブルワード :1...32	

次の表にオペランドの詳細を示します。

タイプ	Op1	Op2
ワード	%MWi, %QWi, %SWi %BLK.x	%MWi, %KWi, %IWi, %QWi, %IWSi, %QWSi, %SWi, %BLK.x
ダブルワード	%MDi %BLK.x	%MDi, %KDi %BLK.x

構造

シフト命令の例

ラング	命令
0	LDR %I0.1 [%MW0:=SHL(%MW10,5)]
1	LDR %I0.2 [%MW10:=ROR(%KW9,8)]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

BCD / 2 進数変換命令

概要

変換命令は、異なる数値の表記を変換します。

BCD / 2 進数変換命令の種類を以下の表に示します。

命令	ファンクション
BTI	BCD から 2 進数に変換
ITB	2 進数から BCD への変換

BCD コードの概要

BCD (2 進化 10 進法 / Binary Coded Decimal) は、2 進数の 4 桁を用いて、10 進数 (0 ~ 9) の 1 桁を表現します。16 ビットワードオブジェクトは 4 桁 (0000 ~ 9999) で表される数値を含むことができ、32 ビットのダブルワードオブジェクトは 8 桁の数字を含むことができます。

変換中、値が BCD でない場合はシステムビット %S18 は 1 に設定されます。このビットはテストされた後、プログラムによって 0 にリセットしてください。

10 進数の BCD 表現 :

10 進数	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

例 :

- ワード %MW5 は BCD の値 2450 を表します。2 進数では 0010 0100 0101 0000 です。
- ワード %MW12 は 10 進数の値 2450 を表します。2 進数では 0000 1001 1001 0010 です。

BTI 命令を使用して、ワード %MW5 はワード %MW12 に変換されます。

ITB 命令を使用して、ワード %MW12 はワード %MW5 に変換されます。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

BCD / 2 進数変換命令の構文

演算子	構文
BTI, ITB	[Op1:= 演算子 (Op2)]

次の表にオペランドの詳細を示します。

タイプ	Op1	Op2
ワード	%MWi, %QWi, %SWi %BLK.x	%MWi, %KWi, %IWi, %QWi, %IWSi, %QWSi, %SWi, %BLK.x
ダブルワード	%MDi %BLK.x	%MDi, %KDi %BLK.x

構造

BCD / 2 進数変換命令の例

ラング	命令
0	LD %M0 [%MW0:=BTI(%MW10)]
1	LD %I0.2 [%MW10:=ITB(%KW9)]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

使用例

BTI 命令は、BCD でエンコードされたサムホイールを介してコントローラー入力のセットポイント値を処理するために使用します。

ITB 命令は、BCD コード表示器で数値 (例えば、計算結果やファンクションブロックの現在値) を表示するために使用します。

シングル / ダブルワード変換命令

概要

シングルワードとダブルワードの間の変換を実行するために使用される命令を以下の表に示します。

命令	ファンクション
LW	ダブルワードの LSB をワードへ書出し。
HW	ダブルワードの MSB をワードへ書出し。
CONCATW	2 つのワードを連結してダブルワードにします。
DWORD	16 ビットワードを 32 ビットのダブルワードに変換します。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

シングル / ダブルワード変換命令の構文

演算子	構文	Op1	Op2	Op3
LW, HW	Op1 = 演算子 (Op2)	%MWi	%MDi, %KDi, %BLK.x	[-]
CONCATW	Op1 = 演算子 (Op2, Op3))	%MDi, %BLK.x	%MWi, %KWi, 即値	%MWi, %KWi, 即値
DWORD	Op1 = 演算子 (Op2)	%MDi, %BLK.x	%MWi, %KWi	[-]

構造

シングル / ダブルワード変換命令の例

ラング	命令
0	LD %M0 [%MW0:=HW(%MD10)]
1	LD %I0.2 [%MD10:=DWORD(%KW9)]
2	LD %I0.3 [%MD11:=CONCATW(%MW10,%MW5)]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

3.3
プログラム

このセクションの目的

このセクションでは、命令のプログラムの概要について説明します。

このセクションについて

このセクションには次の項目が含まれています。

END 命令

概要

END 命令は、プログラムスキャンの実行終了を定義します。

END、ENDC、および ENDCN

4 つの異なる END 命令があります。
- END : プログラムの無条件終了
- ENDC: 直前のテスト命令のブール結果が 1 の場合にプログラムを終了
- ENDCN: 直前のテスト命令のブール結果が 0 の場合にプログラムを終了
- ENDT : グラフセ (SFC) プログラム内の移行ラングを終了 (移行ラングでのみ有効)。

初期設定 (ノーマルモード) では、プログラムの終端が有効になるときに出力は更新され、次のスキャンを開始します。

スキャンが周期的の場合、スキャン周期の終端で出力は更新され、次のスキャンを開始します。

END 命令は次の場合に現在のレベルを終了します。
- サブルーチン、ユーザー定義ファンクション、またはユーザー定義ファンクションブロックで END 命令が実行されると、それぞれを終了して呼び出し側のプログラムに戻ります。
- マスタータスク、定期タスク、またはイベントタスクで END 命令が実行されると、現在のタスクを終了します。

例

無条件の END 命令の例

ラング	命令
0	LD %M1 ST %Q0.1
1	LD %M2 ST %Q0.2
2	END

条件付き END 命令の例

ラング	命令
0	LD %I0.0 ST %Q0.0
1	LD %I0.1 ST %Q0.1
2	LD %I0.2 ENDC
3	LD %I0.3 ST %Q0.2
4	END

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

NOP 命令

概要

NOP 命令は何も実行しません。NOP 命令を使用して、プログラム内の行を 「予約」しておくことで、後に行番号を変更することなく命令を挿入することができます。

ジャンプ命令

概要

ジャンプ命令はプログラムの実行を中断し、ラベル %Li (i = 最大モジュールの数) を含んだ行以降を続けて実行します。

JMP、JMPC、および JMPCN

3 種類のジャンプ命令があります。

- JMP : 無条件プログラムジャンプ
- JMPC: 直前のロジックのブール結果が 1 の場合にプログラムジャンプ
- JMPCN: 直前のロジックのブール結果が 0 の場合にプログラムジャンプ

例

ジャンプ命令の例

ラング	命令
0	LD %M15 JMPC %L8
1	LD [%MW24<%MW12] ST %Q0.3 JMPC %L12
2	%L8: LD %M12 AND %M13 ST %M12 JMPC %L12
3	LD %M11 S %Q0.0
4	%L12: LD %I0.0 ST %Q0.4

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

ガイドライン

- ジャンプ命令は括弧内では使用できません。AND 命令、OR 命令、および閉じ括弧命令 ")" の間にも配置しないでください。
- ラベルは LD、LDN、LDR、LDF、または BLK 命令の前に置いてください。
- ラベル %Li のラベル番号はプログラム内で一度だけ定義してください。
- プログラムジャンプはアップストリームまたはダウンストリームのプログラミングの行に対して実行されます。ジャンプがアップストリームの場合、プログラムのスキャン時間に注意してください。拡張スキャン時間はウォッチドッグタイマーをトリガーする可能性があります。

条件要素

説明

条件要素を使用すると、プログラムの条件をコーディングできます。この要素はオフラインモードでのみプログラムできます。

この機能には次に示す 3 つの要素が含まれています。

- IF
- ELSE
- ENDIF

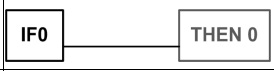
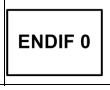
各ラングごとに 1 つの要素を挿入できます。

注記：アプリケーションのファンクションレベルは**レベル 6.0** 以上に設定してください。

要素の最大数

要素の最大数は、アプリケーションで宣言された %L から 128 を引いた数です。

要素の説明

要素	IL 演算	ラダー図	説明
IF	IF0...THEN0		ラングの先頭に置いてください。 THEN 要素は自動的に追加されます。この要素は変更できません。
ELSE	ELSE0		IF 要素が前のラングで定義されている場合のみ。 ラングの先頭に置いてください。
ENDIF	ENDIF0		ラングの末尾に置いてください。別の SoMachine Basic の要素を同じラングに挿入することはできません。

条件命令の設定

手順	アクション
1	空白のラングで、  →  → IF を順にクリックして IF 要素を挿入します。
2	オプションでインデックスを変更できます。 1 つの要素のインデックスを変更する場合、同じインデックスを他の要素にも割り当ててください。 同じインデックスを持つ要素は、同じ POU 内になければなりません。
3	プログラムを設定します。
4	必要に応じて、  →  → ELSE を順にクリックして ELSE 要素を挿入します。
5	最後のラングで、  →  → ENDIF を順にクリックして ENDIF 要素を挿入します。

ループ要素

説明

ループ要素を使用すると、プログラム内で一連の命令をコーディングできます。この要素はオフラインモードでのみプログラムできます。

この機能には次に示す 2 つの要素が含まれています。

- FOR
- ENDFOR

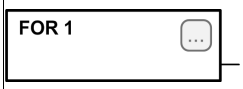
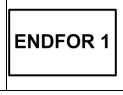
各ラングごとに 1 つの要素を挿入できます。

注記：アプリケーションのファンクションレベル *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* は**レベル 6.0** 以上に設定してください。

要素の最大数

要素の最大数は、アプリケーションで宣言された %L から 128 を引いた数です。

要素の説明

要素	IL 演算	ラダー図	説明
FOR	FORO		ラングの先頭に置いてください。 … をクリックして設定します。
ENDFOR	ENDFORO		ラングの末尾に置いてください。別の SoMachine Basic の要素を同じラングに挿入することはできません。

ループの設定

手順	アクション
1	空白のラングで、 → LOOP → FOR を順にクリックして FOR 要素を挿入します。
2	オプションでインデックスを変更できます。 1 つの要素のインデックスを変更する場合、同じインデックスを他の要素にも割り当ててください。 同じインデックスを持つ要素は、同じ POU 内になければなりません。
3	… をクリックして FOR 要素を設定します。 **結果** :**FOR アシスタント**ウィンドウが表示されます。
4	最後のラングで、 → LOOP → ENDFOR を順にクリックして ENDFOR 要素を挿入します。

FOR 要素の設定

ラベル	説明
ループカウンター	%MW x 変数を入力します。
初期値	%MWx 変数または -32768 ～ 32767 の値を入力します。
符号	● **<** ● **<=** ● **=** ● **>=** ● **>** ● **<>**
終値	%MWx 変数または -32768 ～ 32767 の値を入力します。
反復ステップ	%MWx 変数または -32768 ～ 32767 の値を入力します。

サブルーチン命令

概要

サブルーチン命令は、プログラムにサブルーチンを実行させ、その後、呼び出されたメインプログラムの位置に戻る命令です。

手順

Free POU にサブルーチンが作成されます。Free POU とサブルーチンの作成およびサブルーチン番号を定義する方法については、Free POUs *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を参照してください。また、タスクとラングを使用した POU 管理の詳細については、POU の管理 *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を参照してください。

サブルーチンを呼び出すための手順は以下です。

1. 直前のブール命令の結果が 1 の場合、SRn 命令は Free POU SRn によって参照されるサブルーチンを呼び出します。
2. サブルーチンは Free POU SRn によって参照されます (n はサブルーチンの番号です)。
3. サブルーチン命令は、メインプログラムとは独立して Free POU で記述してください。

サブルーチンの詳細については、定期タスクの作成 *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を参照してください。

例

サブルーチンを含む命令の例

ラング	命令
0	LD %M15 AND %M5 ST %Q0.0
1	LD [%MW24>%MW12] SR1
2	LD %I0.4 AND %M13 ST %Q0.1 END

サブルーチン命令の例 (SR1):

ラング	命令
0 (SR1)	LD %I0.0 ST %Q0.0

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

ガイドライン

- サブルーチンは別のサブルーチンを呼び出すことはできません。Free POU 内でサブルーチンを呼び出そうとすると、コンパイルエラーが発生します。
- サブルーチン命令は括弧内では使用できません。AND 命令、OR 命令、および閉じ括弧命令 ")" の間にも配置しないでください。
- 代入命令が IL のサブルーチン呼び出しの直後にある場合は注意が必要です。これはサブルーチンがブール型アキュムレーターの内容を変更する可能性があるためです。そのため、復帰時には、呼び出し前とは異なる値を持つ可能性があります。

3.4
浮動小数点

このセクションの目的

このセクションでは、浮動小数点の高度な命令について説明します。

このセクションについて

このセクションには次の項目が含まれています。

浮動小数点オブジェクトの算術命令

概要

これらの命令は、2 つの浮動小数点オペランド間または 1 つの浮動小数点オペランドに対し算術命令を実行するために使用します。

命令	目的
+	2 つのオペランドの加算
-	2 つのオペランドの減算
*	2 つのオペランドの乗算
/	2 つのオペランドの除算
LOG	基数 10 の対数
LN	自然対数
SQRT	オペランドの平方根
ABS	オペランドの絶対値
TRUNC	全体の浮動小数点の値
EXP	自然指数
EXPT	実数による整数のべき乗

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

浮動小数点算術命令の演算子と構文

演算子	構文
+, - *, /	Op1:=Op2 演算子 Op3
SQRT, ABS, TRUNC, LOG, EXP, LN	Op1:= 演算子 (Op2)
EXPT	Op1:= 演算子 (Op2,Op3)

浮動小数点算術命令のオペランド

演算子	Op1	Op2	Op3
+, - *, /	%MFi	%MFi, %KFi, 即値	%MFi, %KFi, 即値
SQRT, ABS, LOG, EXP, LN	%MFi	%MFi, %KFi	[-]
TRUNC	%MFi, %MDi	%MFi, %KFi	[-]
EXPT	%MFi	%MFi, %KFi	%MWi, %KWi, 即値
注記 : SoMachine Basic はファンクションの Op1 に、%MWi を使用できません。			

構造

算術命令の例

ラング	命令
0	LD %M0 [%MF0:=%MF10+129.7]
1	LD %I0.2 [%MF1:=SQRT(%MF10)]
2	LDR %I0.3 [%MF2:=ABS(%MF20)]
3	LDR %I0.4 [%MF3:=TRUNC(%MF2)]
4	LD %M1 [%MF4:=LOG(%MF10)]
5	LD %I0.5 [%MF5:=LN(%MF20)]

ラング	命令
6	LD %I0.0 [%MF6:=EXP(%MF30)]
7	LD %I0.1 [%MF7:=EXPT(%MF40,%MW52)]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

使用規則

- 浮動小数点と整数の値を混合した演算はできません。変換命令 (*76* ページ参照) は、これらの形式のいずれかに変換します。
- システムビット %S18 は 整数演算 (*76* ページ参照) と同じ方法で扱われ、ワード %SW17 は検出されたエラーの原因を示します。
- ファンクションのオペランドが無効な値 (負の数の対数など) である場合、不確定または無限が結果として生成され、ビット %S18 が 1 に変更されます。ワード %SW17 は、検出されたエラーの原因を示します。

注記：TRUNC 命令の場合、システムビット %S17 は影響を受けません。

%MDi を使用した TRUNC 命令のアプリケーション例

結果の格納に %MDi を使用する場合の TRUNC 命令の例を次に示します。

例	結果
TRUNC(3.5)	3
TRUNC(324.18765)	324
TRUNC(927.8904)	927
TRUNC(-7.7)	-7
TRUNC(45.678E+20)	2 147 483 647 (最大符号付きダブルワード) (1) %S18 を 1 に設定
TRUNC(-94.56E+13)	- 2 147 483 648 (最小符号付きダブルワード) (1) %S18 を 1 に設定
(1) この例は %MDi を使用した場合の TRUNC 命令です (%MFi を使用した場合の TRUNC 命令ではオーバーフローがないため、最大および最小の制限はありません)。	

三角関数命令

概要

三角関数演算ができる命令は以下です。

SIN	ラジアン表記の角度のサイン	ASIN	アークサイン (結果は、$-\frac{\pi}{2}$ から $\frac{\pi}{2}$ の範囲)
COS	ラジアン表記の角度のコサイン	ACOS	アークコサイン (結果は、0 から π の範囲)
TAN	ラジアン表記の角度のタンジェント	ATAN	アークタンジェント (結果は、$-\frac{\pi}{2}$ から $\frac{\pi}{2}$ の範囲)

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

演算子、オペランド、および三角関数演算命令の構文

演算子	構文	Op1	Op2
SIN, COS, TAN, ASIN, ACOS, ATAN	Op1:= 演算子 (Op2)	%MFi	%MFi, %KFi

構造

三角関数命令の例

ラング	命令
0	LD %M0 [%MF0:=SIN(%MF10)]
1	LD %I0.0 [%MF1:=TAN(%MF20)]
2	LD %I0.3 [%MF2:=ATAN(%MF30)]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

使用規則

- ファンクションのオペランドが無効な値 (1 より大きい数のアークコサインなど) である場合、不確定または無限が結果として生成され、ビット %S18 が 1 に変更されます。ワード %SW17 は、検出されたエラーの原因を示します。
- ファンクション SIN/COS/TAN は、-4096π から 4096π の間の角度をパラメーターとして使用できますが、-2π から $+2\pi$ の範囲外の角度においては正確性が徐々に減少します。これは、演算の前にパラメーターで実行された乗除演算 (modulo) 2π が不正確なためです。

角度変換命令

概要

これらの命令は変換演算に使用します。

DEG_TO_RAD	度からラジアンへの変換。結果は 0 から 2π の間です
RAD_TO_DEG	ラジアン表記の角度の変換。結果は 0 から 360 度の間です

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

演算子、オペランド、および変換命令の構文

演算子	構文	Op1	Op2
DEG_TO_RAD RAD_TO_DEG	Op1:= 演算子 (Op2)	%MFi	%MFi, %KFi

構造

変換命令の例

ラング	命令
0	LD %M0 [%MF0:=DEG_TO_RAD(%MF10)]
1	LD %M2 [%MF2:=RAD_TO_DEG(%MF20)]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

使用規則

変換する角度は -737280.0 ∽ +737280.0 (DEG_TO_RAD 変換の場合) または -4096π ∽ 4096π (RAD_TO_DEG 変換の場合) の間にしてください。

この範囲外の値の場合、表示される結果は +1 になります。# QNAN、%S18、および %SW17:X0 ビットは 1 に設定されます。

整数 / フロート変換命令

概要

4 種類の変換命令があります。

INT_TO_REAL	整数ワードからフロートへの変換
DINT_TO_REAL	ダブルワード (整数) からフロートへの変換
REAL_TO_INT	フロートを整数ワードに変換 (結果は最も近い代数値です)
REAL_TO_DINT	フロートを整数ダブルワードに変換 (結果は最も近い代数値です)

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

演算子と構文 (整数ワードからフロートへの変換)

演算子	構文
INT_TO_REAL	Op1=INT_TO_REAL(Op2)

演算子 (整数ワードからフロートへの変換)

Op1	Op2
%MFi	%MWi,%KWi

例 : 整数ワードからフロートへの変換 : 147...1.47e+02

演算子と構文 (ダブルワード (整数) からフロートへの変換)

演算子	構文
DINT_TO_REAL	Op1=DINT_TO_REAL(Op2)

演算子 (ダブルワード (整数) からフロートへの変換)

Op1	Op2
%MFi	%MDi,%KDi

例 : ダブルワード (整数) からフロートへの変換 : 68905000 ∽ 6.8905e+07

演算子と構文 (フロートから整数ワードまたは整数ダブルワードへの変換)

演算子	構文
REAL_TO_INT	Op1= 演算子 (Op2)
REAL_TO_DINT	

演算子 (フロートから整数ワードまたは整数ダブルワードへの変換)

タイプ	Op1	Op2
ワード	%MWi	%MFi, %KFi
ダブルワード	%MDi	%MFi, %KFi

例 :

- フロートから整数ワードへ変換 : 5978.6 ～ 5979
- フロートから整数ダブルワードへ変換 : -1235978.6 ～ -1235979

注記 : 実数から整数 (または実数から整数ダブルワード) 変換の間、フロートの値がワード (またはダブルワード) の範囲を超えている場合、ビット %S18 は 1 に設定されます。

構造

整数 / フロート変換命令の例

ラング	命令
0	LD 1 [%MF0:=INT_TO_REAL(%MW10)]
1	LD %I0.8 [%MD2:=REAL_TO_DINT(%MF9)]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

端数処理 (丸め) の精度

IEEE 754 では、浮動小数点演算に対して 4 つの丸め方法が定義されています。

上記の命令で使用される方法は、最近接な値に丸める方法です。

論理上の結果から最も近い表現できる値がそれぞれ等しい距離にある場合、最下位ビットが 0 の値になるほうを採用します。

つまり、値を切り上げて丸めても切り捨てて丸めても結果は偶数となります。

例 :

- 10.5 を 10 にする端数処理。
- 11.5 を 12 にする端数処理。

3.5 ASCII

このセクションの目的

このセクションでは ASCII の高度な命令について説明します。

このセクションについて

このセクションには次の項目が含まれています。

ROUND 命令

概要

ROUND 命令は ASCII 文字列に格納に格納された浮動小数点表記の端数処理 (丸め) をします。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

ROUND 命令では次の構文を使用してください。Op1 := ROUND(Op2,Op3)

例 :

[%MW0:7:=ROUND(%MW8,4)]

パラメーター

ROUND ファンクションのパラメーターについて次の表に示します。

パラメーター	説明
Op1	結果が格納される %MW
Op2	%MW は丸められる浮動小数点を含みます
Op3	丸めに必要な有効桁数 1 から 8 までの整数

使用規則

ROUND 命令の規則は次のとおりです。

- オペランドは常に切り捨てられます。
- オペランドの文字列の終了文字は、結果の文字列の終了文字として使用されます。
- [0 - 9] ([16#30 - 16#39]) の範囲にない ASCII 文字は終了文字とみなされます。ただし、次の場合を除きます。
  - ドット '.'(16#2E)
  - マイナス '-' (16#2D)
  - プラス '+' (16#2B)
  - Exp 'e' または 'E' (16#65 または 16#45)
- 13 バイトを超える結果とオペランドは使用できません。ASCII 文字列の最大サイズは 13 バイトです。
- 科学的記法は無効です。

特殊な場合

ソフトウェアは構文をチェックします。構文エラーが発生する例を次に示します。

不正な構文	正しい構文
%MW10 := ROUND(%MW1,4) ":7" が結果から抜けています。	%MW10:7 := ROUND(%MW1,4)
%MW10:13 := ROUND(%MW1,4) %MW10:n (例えば、n ≠ 7 は不正です)	%MW10:7 := ROUND(%MW1,4)

使用例

ROUND 命令の例を次に示します。

例	結果
ROUND("987654321", 5)	"987650000"
ROUND("-11.1", 8)	"-11.1"
ROUND("NAN")	"NAN"

ASCII から整数への変換命令

概要

ASCII から整数への変換命令は、ASCII 文字列を整数値に変換します。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

ASCII から整数への変換命令では次の構文を使用してください。Op1 := ASCII_TO_INT(Op2)

例 :

[%MW0:=ASCII_TO_INT(%MW8)]

命令は Op2 から最大 4 つのワードオブジェクトを読み込み、整数形式に変換し、その結果を Op1 に格納します。

パラメーター

ASCII から整数への変換ファンクションのパラメーターについて次の表に示します。

パラメーター	説明
Op1	結果が格納される %MW
Op2	%MW または %KW

使用規則

ASCII から整数への変換規則を次に示します。

- Op2 は -32768...32767 の範囲にしてください。
- ファクションは常に最上位バイトを最初に読み込みます。
- 先頭のスペースは無視されます。
- [0 - 9] ([16#30 - 16#39]) の範囲にない ASCII 文字は終了文字とみなされます。ただしマイナス記号 '-' (16#2D) が先頭の文字の場合を除きます。
- オーバーフロー (> 32767 または < -32768) の場合、システムビット %S18 (算術オーバーフローまたは検出されたエラー) が 1 に設定され、32767 または -32768 が返されます。
- オペランドの最初の文字が区切り文字の場合、値 0 が返され、ビット %S18 が 1 に設定されます。
  注記：区切り文字は '+'、'-'、文字 'e'、'E'、または '.' (小数点記号) です。
- 科学的記法は無効です。

使用例

次の ASCII データは %MW10 から %MW13 に格納されます。

パラメーター	16 進値	ASCII 表現
%MW10	16#3932	9, 2
%MW11	16#3133	1, 3
%MW12	16#2038	' ', 8
%MW13	16#387A	8, 'z'

ASCII から整数への変換の例を次の表に示します。

例	結果
%MW20 := ASCII_TO_INT(%MW10)	%MW20 = 29318
%MW20 := ASCII_TO_INT(%MW12)	%MW20 = 8
%MW20 := ASCII_TO_INT(%MW13)	%MW20 = 0 および %S18 は、1 に設定されます

整数から ASCII への変換命令

概要

整数から ASCII への変換命令は、整数を ASCII 文字列の値に変換します。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

整数から ASCII への変換命令では次の構文を使用してください。Op1 := INT_TO_ASCII(Op2)

例 :

[%MW0:4:=INT_TO_ASCII(%MW8)]

パラメーター

整数から ASCII への変換ファンクションのパラメーターについて次の表に示します。

パラメーター	説明
Op1	結果が格納される %MW
Op2	%MW, %KW, %SW, %IW, %QW または任意の WORD (即値は使用できません)

使用規則

整数から ASCII への変換規則を次に示します。

- Op2 は -32768...32767 の範囲にしてください。
- ファクションは常に最上位バイトを最初に書き込みます。
- 終了文字は「enter」(ASCII 13) です。
- ファンクションは、ASCII 値 (1 ∽ 4) を格納する %MW 変数の個数を自動的に決定します。

構文エラー

ソフトウェアは構文をチェックします。構文エラーが発生する例を次に示します。

不正な構文	正しい構文
%MW10 := INT_TO_ASCII(%MW1) ":4" が結果から抜けています。	%MW10:4 := INT_TO_ASCII(%MW1)
%MW10:n := INT_TO_ASCII(%MW1) %MW10:n (ここで n ≠ 4 は不正です)	%MW10:4 := INT_TO_ASCII(%MW1)

使用例

MW10:4 := INT_TO_ASCII(%MW1) の場合

条件	結果	
整数値	**16 進値**	**ASCII 表現**
%MW1 = 123	%MW10 = 16#3231	2, 1
	%MW11 = 16#0D33	‘enter’, 3
%MW1 = 45	%MW10 = 16#3534	5, 4
	%MW11 = 16#000D	‘enter’
%MW1 = 7	%MW10 = 16#0D37	‘enter’, 7
%MW1 = -12369	%MW10 = 16#3145	1, ‘-’
	%MW11 = 16#3332	3, 2
	%MW10 = 16#3936	9, 6
	%MW11 = 16#000D	‘enter’

ASCII からフロートへの変換命令

概要

ASCII からフロートへの変換命令は、ASCII 文字を浮動小数点の値に変換します。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

ASCII からフロートへの変換命令では次の構文を使用してください。Op1 := ASCII_TO_FLOAT(Op2)

例 :

[%MF0:=ASCII_TO_FLOAT(%MW8)]

パラメーター

ASCII からフロートへの変換ファンクションのパラメーターについて次の表に示します。

パラメーター	説明
Op1	%MF
Op2	%MW or %KW

使用規則

ASCII からフロートへの変換規則を次に示します。

- ファクションは常に最上位バイトを最初に読み込みます。
- [0 - 9] ([16#30 - 16#39]) の範囲にない ASCII 文字は終了文字とみなされます。ただし、次の場合は除きます。
  - ドット '.'(16#2E)
  - マイナス '-' (16#2D)
  - プラス '+' (16#2B)
  - Exp 'e' または 'E' (16#65 または 16#45)
- ASCII 文字列形式は指数表記 ("-2.34567e+13") または 10 進表記 (9826.3457) にすることができます。
- オーバーフロー (計算結果が > 3.402824E+38 または < -3.402824E+38) の場合
  - システムビット %S18 (算術オーバーフローまたは検出されたエラー) は 1 に設定されます。
  - %SW17: X3 は 1 に設定されます。
  - 値 +/- 1.#INF (+ または - 無限値) が返されます。
- 計算結果が -1.175494E-38 ～ 1.175494E-38 の間にある場合、結果は 0.0 に丸められます。
- オペランドが数値でない場合
  - 値 1.#QNAN が返されます。
  - ビット %SW17 : X0 は 1 に設定されます。

使用例

次の ASCII データは %MW10 から %MW14 に格納されます。

パラメーター	16 進値	ASCII 表現
%MW10	16#382D	8, '-'
%MW11	16#322E	2, '.'
%MW12	16#3536	5, 6
%MW13	16#2B65	'+', 'e'
%MW14	16#2032	' ', 2

ASCII からフロートへの変換の例を次の表に示します。

例	結果
%MF20 := ASCII_TO_FLOAT(%MW10)	%MF20 = -826.5
%MF20 := ASCII_TO_FLOAT(%MW11)	%MF20 = 1.#QNAN
%MF20 := ASCII_TO_FLOAT(%MW12)	%MF20 = 6500.0

例	結果
%MF20 := ASCII_TO_FLOAT(%MW13)	%MF20 = 1.#QNAN
%MF20 := ASCII_TO_FLOAT(%MW14)	%MF20 = 2.0

フロートから ASCII への変換命令

概要

フロートから ASCII への変換命令は浮動小数点の値を ASCII 文字列の値に変換します。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

フロートから ASCII への変換命令では次の構文を使用してください。Op1 := FLOAT_TO_ASCII(Op2)

例 :

[%MW0:7:=FLOAT_TO_ASCII(%MF8)]

パラメーター

フロートから ASCII への変換ファンクションのパラメーターについて次の表に示します。

パラメーター	説明
Op1	%MW
Op2	%MF or %KF

使用規則

フロートから ASCII への変換規則を次に示します。

- ファクションは常に最上位バイトを最初に書き込みます。
- 表現には従来の指数表記を用います。
- 結果が無限または数値以外である場合は文字列「NAN」を返します。
- 終了文字は「enter」(ASCII 13) です。
- ファンクションは ASCII 値 を格納する %MW 変数の個数を自動的に決定します。
- 変換精度は 6 桁です

構文エラー

ソフトウェアは構文をチェックします。構文エラーが発生する例を次に示します。

不正な構文	正しい構文
%MW10 := FLOAT_TO_ASCII(%MF1) ":7" が結果から抜けています。	%MW10:7 := FLOAT_TO_ASCII(%MF1)
%MW10:n := FLOAT_TO_ASCII(%MF1) %MW10:n:n (ここで n ≠ 7 は不正です)	%MW10:7 := FLOAT_TO_ASCII(%MF1)

使用例

%MW10:7 := FLOAT_TO_ASCII(%MF1) の場合

変換する数値	結果
1234567800	1.23456e+09
0.000000921	9.21e-07
9.87654321	9.87654
1234	1.234e+03

ASCII からダブルワードへの変換命令

概要

ASCII からダブルワードへの変換命令は ASCII 文字列をダブルワード値に変換します。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

ASCII からダブルワードへの変換命令では次の構文を使用してください。Op1 := ASCII_TO_DINT(Op2)

例 :

[%MD4 := ASCII_TO_DINT(%MW5)]

パラメーター

ASCII からダブルワードへの変換命令のパラメーターについて次の表に示します。

パラメーター	説明
Op1	%MDx
Op2	%MWy または %KWy

注記：Op1 および Op2 をアニメーションテーブルに定義する必要はありません。

使用規則

ASCII から整数への変換規則を次に示します。

- Op2 は -2147483648...2147483647 の範囲にしてください。
- ファクションは常に最上位バイトを最初に読み込みます。
- 先頭のスペースは無視されます。
- [0 - 9] ([16#30 - 16#39]) の範囲にない ASCII 文字は終了文字とみなされます。ただしマイナス記号 '-' (16#2D) が先頭の文字の場合を除きます。
- オーバーフロー (> 2147483647 または < -2147483648) の場合、システムビット %S18 (算術オーバーフローまたは検出されたエラー) が 1 に設定され、2147483647 または -2147483648 が返されます。
- オペランドの最初の文字が区切り文字の場合、値 0 が返され、ビット %S18 が 1 に設定されます。
  注記：区切り文字は '+'、'-'、文字 'e'、'E'、または '.' (小数点記号) です。
- 科学的記法は無効です。

使用例

次の ASCII データは %MW11 から %MW13 に格納されます。

パラメーター	16 進値	ASCII 表現
%MW8	16#3431	4, 1
%MW9	16#3532	5, 2
%MW10	16#3239	2, 9
%MW11	16#3133	1, 3
%MW12	16#2038	' ', 8
%MW13	16#387A	8, 'z'

ASCII からダブルワードへの変換の例を次の表に示します。

例	結果
%MD10 := ASCII_TO_DINT(%MW8)	%MD10 = 142592318
%MD10 := ASCII_TO_DINT(%MW12)	%MD10 = 8
%MD10 := ASCII_TO_DINT(%MW13)	%MD10 = 0 および %S18 は 1 に設定されます

ダブルワードから ASCII への変換命令

概要

ダブルワードから ASCII への変換命令はダブルワードの値を ASCII 文字列に変換します。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

ダブルワードから ASCII への変換命令では次の構文を使用してください。

```
Op1 := DINT_TO_ASCII( Op2 )
```

例 :

```
[%MW4:6 := DINT_TO_ASCII(%MD5)]
```

パラメーター

ダブルワードから ASCII への変換ファンクションのパラメーターについて次の表に示します。

パラメーター	説明
Op1	%MWx:6
Op2	%MD または %KD

注記：Op1 および Op2 をアニメーションテーブルに定義する必要はありません。

使用規則

整数から ASCII への変換規則を次に示します。

- Op2 は -2147483648...2147483647 の範囲にしてください。
- ファクションは常に最上位バイトを最初に書き込みます。
- 終了文字は「enter」(ASCII 13) です。
- ファンクションは、ASCII 値 (1 ～ 6) を格納する %MW 変数の個数を自動的に決定します。

構文エラー

ソフトウェアは構文をチェックします。構文エラーが発生する例を次に示します。

不正な構文	正しい構文
%MW2 := DINT_TO_ASCII (%MD1) ":6" が結果から抜けています。	%MW2:6 := DINT_TO_ASCII (%MW1)
%MW2:n := DINT_TO_ASCII (%KD7) %MW2 : n (ここで n ≠ 6 は不正です)	%MW2:6 := DINT_TO_ASCII (%KD7)

使用例

%MW0:6 := DINT_TO_ASCII(%MD10) の場合

条件	結果	
整数値	16 進値	ASCII 表現
%MD10 = 1236589	%MW0 = 16#3231	2, 1
	%MW1 = 16#3633	6, 3
	%MW2 = 16#3835	8, 5
	%MW3 = 16#0D37	‘enter’, 9
%MD10 = 45	%MW0 = 16#3534	5, 4
	%MW1 = 16#000D	‘enter’

条件	結果	
整数値	**16 進値**	**ASCII 表現**
%MD10 = -1236945	%MW0 = 16#3145	1, '-'
	%MW1 = 16#3332	3, 2
	%MW2 = 16#3936	9, 6
	%MW3 = 16#3534	5, 4
	%MW4 = 16#000D	'enter'

3.6
スタック演算子

スタック命令 (MPS、MRD、MPP)

概要

スタック命令はコイルへのルーティングを処理します。MPS、MRD、および MPP 命令は、最大 32 のブール式を格納できるスタックと呼ばれる一時記憶領域を使用します。

注記：これらの命令は括弧内の式で使用できません。

構文

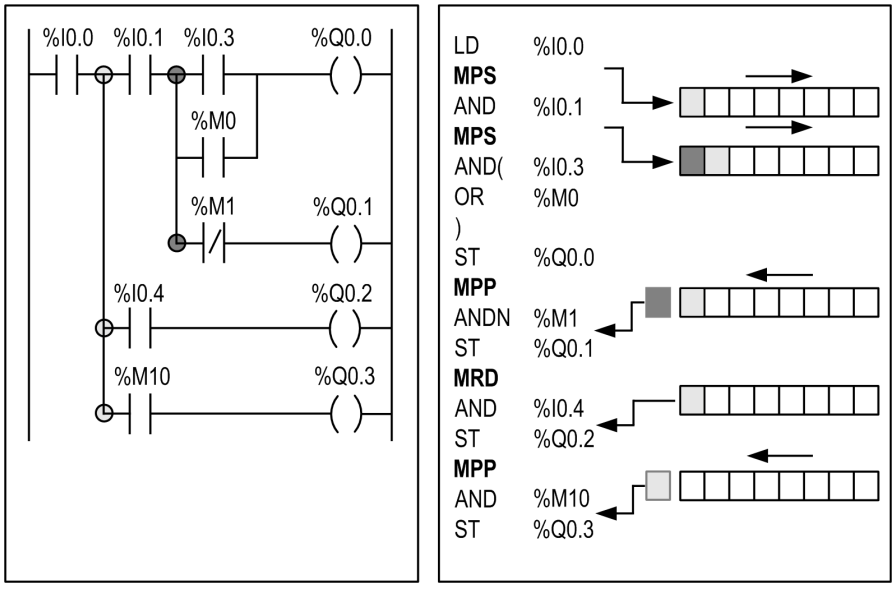
3 つのスタック命令について次の表に示します。

命令	説明	ファンクション
MPS	スタックへのメモリープッシュ	最後の論理命令 (アキュムレーターの内容) の結果をスタックの先頭 (プッシュ) に格納し、他の値をスタックの最下位にシフトします。
MRD	スタックからのメモリー読み込み	スタックの上部をアキュムレーターに読み込みます。
MPP	スタックからのメモリーポップ	スタックの一番上の値をアキュムレーター (ポップ) にコピーし、他の値をスタックの一番上にシフトします。

注記：各 MPS (プッシュ) 命令に対応する MPP (ポップ) 命令は同じラング内で実行してください。

演算

この図はスタック命令の動作を示しています。



使用例

スタック命令の使用例

ラング	命令
0	LD %I0.0 AND %M1 MPS AND %I0.1 ST %Q0.0 MRD AND %I0.2 ST %Q0.1 MRD AND %I0.3 ST %Q0.2 MPP AND %I0.4 ST %Q0.3

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

3.7
オブジェクトテーブルの命令

このセクションの目的

このセクションでは次のオブジェクトテーブルを管理する手順について説明します。

- ダブルワード
- 浮動小数点オブジェクト

このセクションについて

このセクションには次の項目が含まれています。

ワード、ダブルワードテーブル、および浮動小数点テーブルの代入

概要

代入演算は次のオブジェクトテーブルで実行できます。

- 即値からワードテーブル (構造体の例 (*92* ページ参照) のラング 0 を参照してください) またはダブルワードテーブル
- ワードからワードテーブル (構造体の例 (*92* ページ参照) のラング 1 を参照してください)
- ワードテーブルからワードテーブル (構造体の例 (*92* ページ参照) のラング 2 を参照してください)
  テーブルの長さ (L) は、両方のテーブルで同じにしてください。
- ダブルワードからダブルワードテーブル
- ダブルワードテーブルからダブルワードテーブル
  テーブルの長さ (L) は、両方のテーブルで同じにしてください。
- 浮動小数点即値から浮動小数点テーブル
- 浮動小数点から浮動小数点テーブル
- 浮動小数点テーブルから浮動小数点テーブル
  テーブルの長さ (L) は、両方のテーブルで同じにしてください。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

ワードテーブル、ダブルワードテーブル、および浮動小数点テーブルの代入の構文

演算子	構文
:=	[Op1:= Op2] Op1 は Op2 の値をとります

次の表にオペランドの詳細を示します。

タイプ	Op1	Op2
ワードテーブル	%MWi:L, %SWi:L	%MWi:L, %SWi:L, 即値 , %MWi, %KWi, %IW, %QW, %SWi, %BLK.x
ダブルワードテーブル	%MDi:L	即値 , %MDi, %KDi, %MDi:L, %KDi:L
フローティングワードテーブル	%MFi:L	浮動小数点型即値 , %MFi, %KFi, %MFi:L, %KFi:L
L テーブルの長さ (最大 255)。		

注記：略語 %BLK.x (例えば、R3.I) は、任意のファンクションブロックワードを記述するために使用されます。

構造

ワードテーブルの代入の例

ラング	命令
0	LD 1 [%MW0:10:=100]
1	LD %I0.0 [%MW0:10:=%MW11]
2	LDR %I0.3 [%MW10:20:=%KW20:20]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

テーブル集計ファンクション

概要

SUM_ARR ファンクションはオブジェクトテーブルのすべての要素を合計します。

- テーブルがダブルワードで構成されている場合、結果はダブルワードの形式で与えられます。
- テーブルがフローティングワードで構成されている場合、結果はフローティングワードの形式で与えられます。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

テーブル集計命令の構文

```
Res:=SUM_ARR(Tab)
```

テーブル集計命令のパラメーター

タイプ	結果 (Res)	テーブル (Tab)
ダブルワードテーブル	%MDi	%MDi:L,%KDi:L
フローティングワードテーブル	%MFi	%MFi:L,%KFi:L
L　テーブルの長さ (最大 255)。		

注記：結果がテーブルオペランドの有効なダブルワード形式の範囲内にない場合、システムビット %S18 は 1 に設定されます。

構造

集計ファンクションの例

ラング	命令
0	LD %I0.2 [%MD5:=SUM_ARR(%MD3:1)]
1	LD 1 [%MD5:=SUM_ARR(%KD5:2)]
2	LD 1 [%MF2:=SUM_ARR(%MF8:5)]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

使用例

```
%MD4:=SUM_ARR(%MD30:4)
```

例えば、%MD30=10、%MD32=20、%MD34=30、%MD36=40

結果、%MD4:=10+20+30+40

テーブル比較ファンクション

概要

EQUAL_ARR ファンクションは 2 つのテーブルの要素ごとの比較を行います。

相違がある場合、最初の相違要素の順位がワードで返されます。それ以外の場合の戻り値は -1 です。

比較はテーブル全体で実行されます。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

テーブル比較命令の構文

```
Res:=EQUAL_ARR(Tab1,Tab2)
```

テーブル比較命令のパラメーター

タイプ	結果 (Res)	テーブル (Tab1 および Tab2)
ダブルワードテーブル	%MWi	%MDi:L,%KDi:L
フローティングワードテーブル	%MWi	%MFi:L,%KFi:L
L　テーブルの長さ (最大 255)。		

注記：テーブルの長さとタイプは同じものを使用してください。

構造

テーブル比較ファンクションの例

ラング	命令
0	LD %I0.2 [%MW5:=EQUAL_ARR(%MD20:7,%KD0:7)]
1	LD 1 [%MW0:=EQUAL_ARR(%MD20:7,%KD0:7)]
2	LD 1 [%MF2:=SUM_ARR(%MF8:5)]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

使用例

```
%MW5:=EQUAL_ARR(%MD30:4,%KD0:4)
```

2 つのテーブルの比較

順位	ワードテーブル	ワード型定数テーブル	相違
0	%MD30=10	%KD0=10	=
1	%MD32=20	%KD2=20	=
2	%MD34=30	%KD4=60	同じでない
3	%MD36=40	%KD6=40	=

ワード %MW5 の値は 2 です (最初の相違の順位)

テーブル検索ファンクション

概要

3 つの検索ファンクションがあります。

- FIND_EQR : 最初の要素が指定された値と等しいダブルまたはフローティングワードテーブルの位置の検索をします。
- FIND_GTR : 最初の要素が指定された値より大きいダブルまたはフローティングワードテーブルの位置の検索をします。
- FIND_LTR : 最初の要素が指定された値より小さいダブルまたはフローティングワードテーブルの位置の検索をします。

これらの命令の結果は最初に検索された要素の順位と等しくなります。また、検索に失敗した場合は -1 になります。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

テーブル検索命令の構文

ファンクション	構文
FIND_EQR	Res:= ファンクション (Tab,Val)
FIND_GTR	
FIND_LTR	

フローティングワードおよびダブルワードのテーブル検索命令のパラメーター

タイプ	結果 (Res)	テーブル (Tab)	値 (Val)
フローティングワードテーブル	%MWi	%MFi:L,%KFi:L	%MFi,%KFi
ダブルワードテーブル	%MWi	%MDi:L,%KDi:L	%MDi,%KDi
L テーブルの長さ (最大 255)。			

構造

テーブル検索ファンクションの例

ラング	命令
0	LD %I0.2 [%MW5:=FIND_EQR(%MD20:7,%KD0)]
1	LD %I0.3 [%MW6:=FIND_GTR(%MD20:7,%KD0)]
2	LD 1 [%MW7:=FIND_LTR(%MF40:5,%KF4)]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

使用例

%MW5:=FIND_EQR(%MD30:4,%KD0)

最初のダブルワードの位置 = %KD0=30 をテーブル内で検索

順位	ワードテーブル	結果
0	%MD30=10	-
1	%MD32=20	-
2	%MD34=30	値 (Val)、順位
3	%MD36=40	-

最大値と最小値のテーブル検索ファンクション

概要

2 つの検索ファンクションがあります。
- MAX_ARR : ダブルワードテーブルおよびフローティングワードテーブルの最大値の検索
- MIN_ARR : ダブルワードテーブルおよびフローティングワードテーブルの最小値の検索

これらの命令の結果は、テーブル内の最大値 (または最小値) に等しくなります。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

最大値および最小値のテーブル検索命令の構文

ファンクション	構文
MAX_ARR	結果 := ファンクション (Tab)
MIN_ARR	

最大値および最小値のテーブル検索命令のパラメーター

タイプ	結果 (Res)	テーブル (Tab)
ダブルワードテーブル	%MDi	%MDn:L, %KDn:L
フローティングワードテーブル	%MFi	%MFn:L, %KFn:L
i 変数メモリーのオブジェクトインスタンス識別子。 **n** 検索のベースアドレスを示すテーブルのメモリーインデックス。 **L** ベースアドレスインデックス (L の最大値は 255 です) を含む、検索時に考慮されるべき位置の数。		

注記：L は検索の際に重複していないアドレスのみの数です。詳細については、オブジェクト間のオーバーラップの可能性 (*30* ページ参照) を参照してください。

構造

テーブル検索ファンクションの例

ラング	命令
0	LD %I0.2 [%MD0:=MIN_ARR(%MD20:7)]
1	LD 1 [%MF8:=MIN_ARR(%MF40:5)]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

テーブル内の値の出現回数

概要

OCCUR_ARR ファンクションは、与えられた値と等しい要素の数をダブルワードテーブルまたはフローティングワードテーブル内で検索します。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

最大値および最小値のテーブル検索命令の構文

ファンクション	構文
OCCUR_ARR	Res:= ファンクション (Tab,Val)

最大値および最小値のテーブル検索命令のパラメーター

タイプ	結果 (Res)	テーブル (Tab)	値 (Val)
ダブルワードテーブル	%MWi	%MDi:L,%KDi:L	%MDi,%KDi
フローティングワードテーブル	%MFi	%MFi:L,%KFi:L	%MFi,%KFi
L テーブルの長さ (最大 255)。			

構造

出現回数の例

ラング	命令
0	LD %I0.3 [%MW5:=OCCUR_ARR(%MF20:7,%KF0)]
1	LD %I0.2 [%MW5:=OCCUR_ARR(%MD20:7,%MD1)]

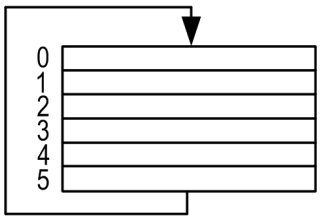
注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

テーブルローテートシフトファンクション

概要

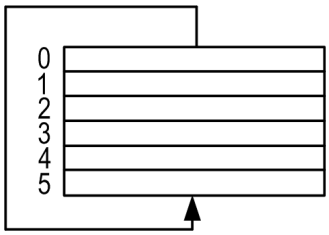
2 つのシフトファンクションがあります。
- ROL_ARR : フローティングワードテーブルの要素を n だけ上から下にローテートシフトします

ROL_ARR ファンクションの図



- ROR_ARR : フローティングワードテーブルの要素を n だけ下から上にローテートシフトします

ROR_ARR ファンクションの図



構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

フローティングワードテーブルまたはダブルワードテーブルのローテートシフト命令の構文 ROL_ARR および ROR_ARR

ファンクション	構文
ROL_ARR	ファンクション (n,Tab)
ROR_ARR	

フローティングワードテーブルのローテートシフト命令のパラメーター **ROL_ARR** および **ROR_ARR**

タイプ	位置の数 (n)	テーブル (Tab)
フローティングワードテーブル	%MWi、即値	%MFi:L
ダブルワードテーブル	%MWi、即値	%MDi:L
L テーブルの長さ (最大 255)。		

注記：n の値が負または null の場合、シフトは実行されません。

構造

テーブルローテートシフトファンクションの例

ラング	命令
0	LD %I0.2 [ROL_ARR(%KW0,%MD20:7)]
1	LD %I0.3 [ROR_ARR(2,%MD20:7)]
2	LD %I0.4 [ROR_ARR(2,%MF40:5)]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

テーブルソートファンクション

概要

SORT_ARR はダブルワードまたはフローティングワードテーブルの要素を昇順または降順でソートして、同じテーブルに結果を保存します。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

テーブルソートファンクションの構文

ファンクション	構文
SORT_ARR	ファンクション (direction,Tab)

"direction" パラメーターはソート順を示します
- Direction > 0 : ソートは昇順で行われます。
- Direction < 0 : ソートは昇順で行われます。
- Direction = 0: ソートは実行されません。

結果 (ソートされたテーブル) は、Tab パラメーター (ソートするテーブル) に返されます。

テーブルソートファンクションのパラメーター

タイプ	ソート順	テーブル (Tab)
ダブルワードテーブル	%MWi、即値	%MDi:L
フローティングワードテーブル	%MWi、即値	%MFi:L
L テーブルの長さ (最大 255)。		

構造

テーブルソートファンクションの例

ラング	命令
0	LD %I0.1 [SORT_ARR(%MW20,%MF0:6)]
1	LD %I0.2 [SORT_ARR(%MW20,%MF0:6)]
2	LD %I0.3 [SORT_ARR(0,%MF40:8)]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

浮動小数点テーブル補間 (LKUP) ファンクション

概要

LKUP ファンクションは与えられた X 値に対して、X 対 Y の浮動小数点データを補間します。

線形補間の概要

LKUP 関数は、次の式で定義される線形補間の方式を使用します。

$$Y = Y_i + \left[\frac{(Y_{i+1} - Y_i)}{(X_{i+1} - X_i)} \cdot (X - X_i)\right] \quad (式\ 1)$$

このとき $X_i \le X \le X_{i+1}$ 、$i = 1 \ldots (m-1)$;

X_i の値は昇順に並べられたものだと仮定します。$X_1 \le X_2 \le \ldots X \ldots \le X_{m-1} \le X_m$.

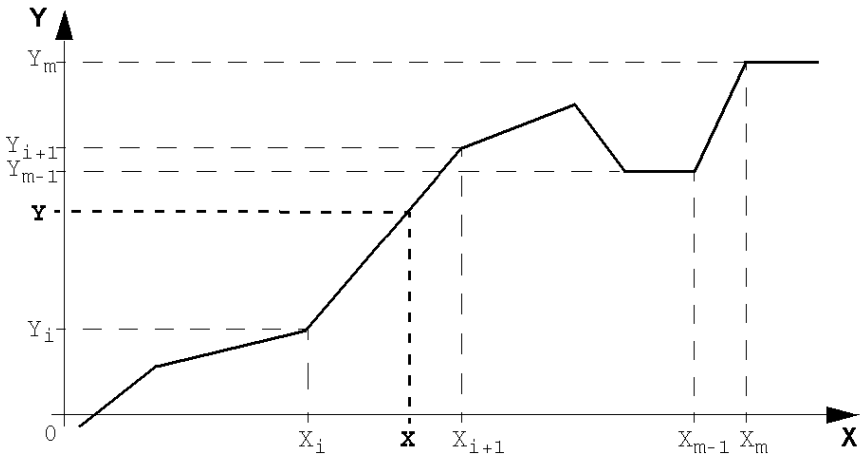
注記：2 つの連続する Xi のいずれか (X_i=X_{i+1}=X) に等しい場合、式 (1) は無効な例外を生じます。この場合、この例外に対処するために式 (1) の代わりに以下のアルゴリズムが使用されます。

$$Y = \left[\frac{(Y_{i+1} - Y_i)}{2}\right] \quad (式\ 2)$$

このとき $X_i = X_{i+1} = X$ 、$i = 1 \ldots (m-1)$

図

このグラフは上記の線形補間の方式を示しています。



構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

LKUP ファンクションは 3 つのオペランドを使用します。この表に示すように、このうちの 2 つはファンクション属性です。

構文	Op1 出力変数	Op2 ユーザー定義 (X) 値	Op3 ユーザー定義 (X_i,Y_i) 配列変数
[Op1:= LKUP(Op2,Op3)]	%MWi	%MF0	整数値、%MWi、または　%KWi

Op1 の定義

Op1 は補間ファンクションの出力変数を含むワードメモリーです。

次の表では補間が成功したかどうかを示す Op1 の値の説明をします。

Op1 (%MWi)	説明
0	補間に成功しました
1	補間エラーを検出 : 不正な配列、$X_m < X_{m-1}$
2	補間エラーを検出 :Op2 が範囲外、$X < X_1$
4	補間エラーを検出 :Op2 が範囲外、$X > X_m$
8	データ配列のサイズが無効 : ● Op3 に奇数を設定している、または ● Op3 < 6.

注記：Op1 には計算された補間値 (Y) は含まれません。与えられた (X) 値に対して、補間 (Y) の結果は Op3 配列 (*101* ページ参照) の %MF2 に含まれます。

Op2 の定義

Op2 は、補間された (Y) 値を計算するためのユーザー定義 (X) 値を含む浮動小数点変数 (Op3 浮動小数点配列の %MF0) です。

Op2 の有効範囲 : $X_1 \le Op2 \le X_m$.

Op3 の定義

Op3 はデータ対 (X_i,Y_i) が格納される浮動小数点配列のサイズ (Op3 / 2) を設定します。

X_i データと Y_i データは、%MF4 から始まる偶数インデックスの浮動小数点オブジェクトに格納されます。(浮動小数点オブジェクトの %MF0 と %MF2 は、それぞれユーザー設定ポイント X と補間値 Y に対して予約されています)。

配列に (m) 個のデータ対 (X_i、Y_i) を与えると、浮動小数点配列 (%MFu) の上位インデックス (u) は以下の関係で決まります。

- $Op3 = 2 \cdot m$ *(式 3)*
- $u = 2 \cdot (Op3 - 1)$ *(式 4)*

浮動小数点配列 Op3 (%MFi) は、この例の構造体と同様の構造を持ちます (例えば、Op3 = 8)。

(X)		(X_1)		(X_2)		(X_3)	
%MF0		%MF4		%MF8		%MF12	
	%MF2		%MF6		%MF10		%MF14
	Y:		(Y_1)		(Y_2)		(Y_3)
							(Op3=8)

注記：上記の浮動小数点構造の配列の結果として、Op3 は次の要件の両方を満たす必要があります。そうでない場合 LKUP ファンクションはエラーとなります。

- Op3 は偶数。
- Op3 ≧ 6 (線形補間をするには少なくとも 2 つのデータ点が必要です)。

構造

補間演算は以下のように実行されます。

ラング	命令
0	LD %I0.2 [%MW20:=LKUP(%MF0,%KW1)]
1	LD %I0.3 [%MW22:=LKUP(%MF0,10)]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

使用例

LKUP 補間ファンクションの使用 :

```
[%MW20:=LKUP(%MF0,10)]
```

この例では :

- %MW20 は Op1 (出力変数) です。
- %MF0 はユーザー定義 (X) 値で、対応する (Y) 値が線形補間によって計算してください。
- %MF2 は線形補間によって計算された値 (Y) を格納します。
- 10 は Op3 です (上記の*式 3* によって与えられます)。浮動小数点配列のサイズを設定します。最高順位のアイテムは %MFu です。例えば、u=18 は上記の*式 4* により与えられます。

Op3 配列 [%MF4,...%MF18] には 4 組のデータポイントが格納されています。

- %MF4 は X_1 を含み、%MF6 は Y_1 を含みます。
- %MF8 は X_2 を含み、%MF10 は Y_2 を含みます。
- %MF12 は X_3 を含み、%MF14 は Y_3 を含みます。
- %MF16 は X_4 を含み、%MF18 は Y_4 を含みます。

浮動小数点テーブルの値の MEAN ファンクション

概要

MEAN ファンクションは浮動小数点テーブルの値から平均を計算するのに使われます。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

浮動小数点テーブル平均計算ファンクションの構文

ファンクション	構文
MEAN	結果 = ファンクション (Op1)

浮動小数点テーブルから与えられた数 L (最大 255) の値に対する計算ファンクションのパラメーター

Op1	結果 (Res)
%MFi:L, %KFi:L	%MFi

構造

平均ファンクションの例

ラング	命令
0	LD %I3.2 [%MF0:=MEAN(%MF10:5)]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

3.8
I / O オブジェクトの命令

このセクションの目的

このセクションでは、I / O オブジェクトの命令について説明します。

このセクションについて

このセクションには次の項目が含まれています。

標準デジタル入力の読み込み (READ_IMM_IN)

概要

READ_IMM_IN 命令は、タスクの実行中に標準デジタル入力 (ロジックコントローラー付属の入力) を読み取り、入力イメージを直ちに更新します。そのため、入力イメージを更新するために次のタスクサイクルを待つ必要はありません。

注記：この命令は標準デジタル入力に対してのみ有効です。

注記：この命令を使用するときは、通常の入力と高速入力の相対的なパフォーマンス (オンとオフディレー) を評価します。デジタル入力 を参照してください。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

READ_IMM_IN 命令は次の構文を使用します。

Op1 := READ_IMM_IN(Op2)

ここで

オペランド	タイプ	説明
Op1	%MWi	ファンクションリターンコードを格納します (下の表を参照してください)。
Op2	即値 (整数) %MWi %KWi	入力インデックス (%I0.x) を定義します。
i　変数メモリーのオブジェクトインスタンス識別子。		

ファンクションリターンコード

次の表にファンクションリターンコードを示します。

コード	説明
0	エラー未検出
1	宣言された入力が最大入力よりも大きい。
2	宣言された入力は強制されます。

例

%MW0 := READ_IMM_IN(2)

この演算ブロックが実行されると、入力 %I0.2 の現在値が読み込まれ、入力イメージが直ちに更新されます。ファンクションリターンコードは、%MW0 ワードメモリーに格納されます。

構造

READ_IMM_IN 命令の例 :

ラング	命令
0	LD %M0 [%MW0:=READ_IMM_IN(%MW5)]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

標準デジタル出力の書き込み (WRITE_IMM_OUT)

概要

WRITE_IMM_OUT 命令が標準デジタル出力 (ロジックコントローラー付属の出力) に物理的に直ちに書き込みその値が出力イメージから読み込まれます。そのため、標準出力に書き込むために次のタスクサイクルを待つ必要はありません。

注記：この命令は標準デジタル出力に対してのみ有効です。

注記：この命令を使用するときは、通常の入力と高速入力の相対的なパフォーマンス (オンとオフディレー) を評価します。デジタル入力 を参照してください。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

WRITE_IMM_OUT 命令は次の構文を使用します。

Op1 := WRITE_IMM_OUT(Op2)

ここで

オペランド	タイプ	説明
Op1	%MWi	ファンクションリターンコードを格納します (下の表を参照してください)。
Op2	即値 (整数) %MWi %KWi	出力インデックス (%Q0.x) を定義します。
i　変数メモリーのオブジェクトインスタンス識別子。		

ファンクションリターンコード

次の表にファンクションリターンコードを示します。

コード	説明
0	エラー未検出
3	宣言された出力が最大出力よりも大きい。
4	宣言された出力は強制されます。
5	宣言された出力は専用のハードウェア出力として使用されます。
6	宣言された出力はアラーム出力として使用されます。

例

%MW0 := WRITE_IMM_OUT(%MW5) (%MW5 = 2 とする)

この演算ブロックの実行時に出力イメージ %Q0.2 は標準デジタル出力に物理的に書き込まれます。ファンクションリターンコードは、%MW0 ワードメモリーに格納されます。

構造

WRITE_IMM_OUT 命令の例 :

ラング	命令
0	LD %M0 [%MW0:= WRITE_IMM_OUT(%MW4)]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

ファンクションブロックパラメーターの読み込み (READ_IMM)

概要

READ_IMM 命令は、タスクの実行中にファンクションブロックのパラメーターを読み込み、同じサイクル中に入力イメージを更新します。

このファンクションは特定のファンクションブロックのパラメーターでのみ使用できます。READ_IMM 命令は高速カウンター (%HSC) ファンクションブロックの HSC.V と HSC.P レジスターを直接読み込みます。詳細については、高速カウンター (%HSC) を参照してください。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

READ_IMM 命令は次の構文を使用します。

READ_IMM(Op1)

ここで

オペランド	タイプ	説明
Op1	%HSCx.P、%HSCx.PD、%HSCx.V、%HSCx.VD	この命令は Op1 で与えられたファンクションブロックパラメーターを読み込み、I/O イメージおよび対応するレジスターの値を更新します。
x ファンクションブロックのオブジェクトインスタンス識別子。		

例

次のコードは READ_IMM 命令を使用した例です。

ラング	命令
0	LD %M0 [READ_IMM(%HSC0.P)]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

注記：READ_IMM 命令を使用するには、アプリケーションの ファンクションレベル *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を **レベル 3.3** 以上に設定してください。

ファンクションブロックパラメーターの書き込み (WRITE_IMM)

概要

WRITE_IMM 命令はタスクの実行中にファンクションブロックのパラメーターを書き込み、同じサイクル中に出力イメージを更新します。

このファンクションは特定のファンクションブロックのパラメーターでのみ使用できます。WRITE_IMM 命令は高速カウンター (%HSC) ファンクションブロックの HSC.V と HSC.P レジスターに直接書き込みます。詳細については、高速カウンター (%HSC) を参照してください。

構文

IL 構文は次のとおりです。**演算ブロック**グラフィカル要素を使用して、ラダー図のラングに IL 演算と代入命令を挿入 (*16* ページ参照) できます。

WRITE_IMM 命令は次の構文を使用します。

WRITE_IMM(Op1)

ここで

オペランド	タイプ	説明
Op1	%HSCx.P、%HSCx.PD、%HSCx.V、%HSCx.VD	この命令は Op1 で与えられたファンクションブロックパラメーターを書き込み、I/O イメージの値を更新します。
x ファンクションブロックのオブジェクトインスタンス識別子。		

例

次のコードは WRITE_IMM 命令を使用した例です。

ラング	命令
0	LD %M1 [WRITE_IMM(%HSC0.V)]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

注記：WRITE_IMM 命令を使用するには、アプリケーションの ファンクションレベル を**レベル 3.3** 以上に設定してください。

第 4 章
I/O オブジェクト

この章について

この章には次のセクションが含まれています。

4.1
高速カウンター (%FC)

高速カウンター (FC)

概要

コントローラーのアドバンスト *ファンクションライブラリーガイド* を参照してください。

4.2
高速カウンター (%HSC)

高速カウンター (HSC)

概要

コントローラーのアドバンスト *ファンクションライブラリーガイド* を参照してください。

4.3
パルス (%PLS)

パルス

概要

コントローラーのアドバンスト *ファンクションライブラリーガイド* を参照してください。

4.4
パルス幅変調 (%PWM)

パルス幅変調

概要

コントローラーのアドバンスト *ファンクションライブラリーガイド* を参照してください。

第 5 章
ネットワークオブジェクト

ネットワークオブジェクト

情報

ネットワークオブジェクトは、EtherNet/IP、Modbus TCP、または Modbus Serial IOScanner を介した通信に使われます。

EtherNet/IP 通信には、2 種類のネットワークオブジェクトがあります。
- %QWE: 入力アセンブリ
- %IWE: 出力アセンブリ

Modbus TCP 通信には、2 種類のネットワークオブジェクトがあります。
- %QWM: 入力レジスター
- %IWM: 出力レジスター

Modbus Serial IOScanner には以下のネットワークオブジェクトが使われます。
- %IN: デジタル入力 (IOScanner)
- %QN: デジタル出力 (IOScanner)
- %IWN: 入力レジスター (IOScanner)
- %QWN: 出力レジスター (IOScanner)
- %IWNS: IOScanner ネットワーク診断コード

注記：入力と出力の参照は、EtherNet/IP マスター、または Modbus TCP クライアントの視点から見たものです。

ネットワークオブジェクトの設定の詳細は、ロジックコントローラーのプログラミングガイドを参照してください。

第 6 章
ソフトウェアオブジェクト

この章について

この章には次のセクションが含まれています。

6.1
ファンクションブロックの使用

このセクションについて

このセクションには次の項目が含まれています。

ファンクションブロックプログラミングの原則

概要

ファンクションブロックは再利用可能なオブジェクトです。1 つ以上の入力値を受けつけ、1 つ以上の出力値を返します。

各ラダー図のラングには最大 5 つのファンクションブロックを連続して挿入できます。ファンクションブロックは並列に挿入できません。

次の場合、ファンクションブロックパラメーターは使用できません。
- 使用するコントローラーがファンクションブロックをサポートしていない
- ファンクションブロックが未設定

ラダー図プログラム

ラダー図のラングでファンクションブロックを使用するには

手順	アクション
1	ファンクションブロックをラングに挿入 (*121* ページ参照) します。
2	必要に応じて入力と出力を配線します。
3	そのパラメーターの値を指定してファンクションブロックを設定 (*123* ページ参照) します。

命令リスト (IL) プログラム

命令リストプログラムにファンクションブロックを追加するには、次のいずれかの方法を使用します。
- ファンクションブロック命令 (例えば、BLK %TM2): この可逆的なプログラミング方法によりプログラム内の 1 箇所のブロックで演算を実行することができます。
- 特定の命令 (例えば、CU %Ci)。この不可逆的なプログラミング方法によりプログラム内の複数箇所でファンクションブロック入力で実行される演算が有効になる。例 :

ライン	命令
1000	CU %C1
1074	CD %C1
1209	R %C1

ファンクションブロックの可逆プログラミングには、命令 BLK、OUT_BLK、および END_BLK を使用してください。
- BLK: ブロックの開始を示します。
- OUT_BLK: ブロック出力を直接配線するために使用されます。
- END_BLK: ブロックの終了を示します。

注記 : 関連ブロックのテストおよび入力命令は、BLK と OUT_BLK 命令の間 (または OUT_BLK がプログラムされていない場合は BLK と END_BLK の間) にのみ配置できます。

出力に接続済みの例

この例は、出力に接続済みのカウンターファンクションブロックを示しています。

ラング	命令
0	BLK %C8 LDF %I0.1 R LD %I0.1 AND %M0 CU OUT_BLK LD D AND %M1 ST %Q0.0 END_BLK

注記 : ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

出力に未接続の例

この例は、出力に未接続のカウンターファンクションブロックの可逆プログラミングを示しています。

ラング	命令
0	BLK %C8 LDF %I0.1 R LD %I0.2 AND %M0 CU END_BLK
1	LD %C8.D AND %M1 ST %Q0.4

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

ファンクションブロックの追加

ファンクションブロックをラダー図のプログラムに挿入する

次の手順で行います。

手順	アクション
1	SoMachine Basic のプログラミングワークスペースに新規ラダー図ラングを作成します。詳細については、SoMachine Basic オペレーティングガイド を参照してください。
2	プログラミングワークスペースの上部にあるグラフィカルツールバーの**ファンクション**ボタンをクリックします。 **結果** : 使用可能なファンクションブロックオブジェクトのリストが表示されます (以下の表を参照)。
3	ファンクションブロックを選択します。
4	ラング上の必要な位置にファンクションブロックを移動し、クリックして挿入します。 各ラングには最大 5 つのファンクションブロックを連続して挿入できます。

使用可能なファンクションブロックオブジェクト

使用可能なファンクションブロックオブジェクトを以下の表に示します。

ファンクションブロックオブジェクト	説明
	タイマー
	LIFO/FIFO レジスター
	ビットレジスターをシフト
	ステップカウンター
123	カウンター
1123	高速カウンター (FC)
11123	高速カウンター (HSC)
	ドラム
	RTC (リアルタイムクロック)
	パルス
	パルス幅変調
	メッセージ
	データロギング

ファンクションブロックオブジェクト	説明
	パルストレイン出力 **注記**：PTO オブジェクトのリストについては、アドバンストファンクションライブラリーガイド、PTO ファンクションブロック を参照してください。
DRV	ドライブオブジェクト **注記**：ドライブオブジェクトのリストについては、アドバンストファンクションライブラリーガイド、ドライブファンクションブロック を参照してください。
	通信ファンクションブロック **注記**：通信ファンクションブロックのリストについては、通信オブジェクト (*197* ページ参照) を参照してください。
UDFB	ユーザー定義ファンクションブロック

ファンクションブロックの設定

ラダー図プログラムのファンクションブロックの設定

次の手順で行います。

手順	アクション
1	必要に応じて、ファンクションブロック内の**アドレス**をクリックします。 テキストボックスには初期設定のアドレスが表示されます。例えば、タイマーファンクションブロックの場合は「%TM0」。 初期設定のアドレスを変更するには、アドレスの最後の桁 (インスタンス識別子) を削除します。 使用可能なアドレスのリストが表示されます。 ファンクションブロックのこのインスタンスを識別するために使用するアドレスを選択します。 ファンクションブロックのプロパティは、ファンクションブロックオブジェクトの中央と、プログラミングワークスペースの下半分の**プロパティ**テーブルに表示されます。 プロパティはファンクションブロック内の任意の場所をダブルクリックすることでも表示できます。
2	必要に応じて、ファンクションブロック内の**コメント**をクリックし、ファンクションブロックの短い説明を入力します。例えば、**パルスタイマー**など。
3	必要に応じて、ファンクションブロック内の**シンボル**をクリックし、このファンクションブロックに関連付けるシンボル名を入力します。 入力した文字で始まる名前を持つ既存のすべてのシンボルのリストが表示されます。使用するシンボルをクリックします。 このファンクションブロック用に新しいシンボルを作成するには、作成するシンボルの名前を入力し、シンボルに関連付けるオブジェクトを選択します。 シンボルの使用の詳細については、SoMachine Basic オペレーティングガイド *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を参照してください。
4	ファンクションブロックをクリックします。 **結果 : 設定** ツールチップが表示されます。
5	個々のファンクションブロックの「パラメーター」のトピックで説明されているように、各ファンクションブロックで使用可能なパラメーターを設定します。 オンラインモードでオブジェクトの値を変更することができます。オンライン編集 *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を参照してください。

注記：ラング内のファンクションブロックをダブルクリックして**プロパティ** テーブルを表示することもできます。

6.2
タイマー (%TM)

タイマーファンクションブロックの使用

このセクションではタイマーファンクションブロックのプログラミングガイドラインと説明をします。

このセクションについて

このセクションには次の項目が含まれています。

説明

概要

タイマーファンクションブロック はイベントのトリガーなどの動作までの時間を指定します。

図

次の図はタイマーファンクションブロックを示しています。

入力

タイマーファンクションブロックには次の入力があります。

ラベル	説明	値
IN	入力アドレス (または命令)	立上がり (TON または TP タイプ) または立下がり (TOF タイプ) が検出されたときにタイマーを開始します。

出力

タイマーファンクションブロックには次の出力があります。

ラベル	説明	値
Q	出力アドレス (%TMi.Q)	関連するビット %TMi.Q は、タイマーの指定時間経過時に 1 に設定されます (タイマーのタイプによって異なります)。

設定

パラメーター

パラメーターを設定するには、ファンクションブロックの設定手順 (*123* ページ参照) に従い、SoMachine Basic オペレーティングガイドのメモリー割り当てモードを読んでください。

タイマーファンクションブロックには次のパラメーターがあります。

パラメーター	説明	値	オンラインモードでの編集
使用	使用済みアドレス	選択されている場合、このアドレスはプログラムで使用されています。	不可
アドレス	タイマーオブジェクトアドレス (%TMi)	プログラムで使用できるタイマーオブジェクト数には制限があります。タイマーの最大数については関連するプラットフォームのプログラミングガイドを参照してください。	不可
シンボル	シンボル	オブジェクトに対応したシンボル。詳細については、SoMachine Basic オペレーティングガイド シンボルの定義と使用 *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を参照してください。	不可
タイプ	タイマータイプ	以下のいずれか。 ● TON (*128* ページ参照): タイマーオンディレー (初期設定) ● TOF (*130* ページ参照): タイマーオフディレー ● TP (*131* ページ参照): パルスタイマー (単安定)	可 [1]
保持	True/False	**保持**チェックボックスが選択されていない場合 (初期値)、**IN** パラメーターの立下がりが検出されると値がリセットされます。カウントは 0 から再開します。 **保持**チェックボックスが選択されている場合、**プリセット**値に達する前に **IN** パラメーターの立下がりが検出されると、タイマーはその値を保持します。この値からカウントを再開します。 **注記：保持**パラメーターを使用するには、アプリケーションの ファンクションレベル *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を **レベル 3.3** 以上に設定してください。	可 [1]
Time base	時間ベース	タイマーの基準時間単位。タイマーの基準単位が小さいほど、タイマーの正確性が上がります。 ● **1 ms** (%TM0...%TM5 でサポートされます) ● **10 ms** ● **100 ms** ● **1 s** ● **1 min** (初期値)	可 [1]
プリセット	プリセット値	0...9999。初期値は 9999 です。 タイマーの周期 = **プリセット** × 時間**ベース** タイマーディレー = **プリセット** × 時間**ベース** 設定されたプリセット値は関連オブジェクト %TMi.P を使用して、読み込み、テスト、および変更ができます	可 [1]
コメント	コメント	オブジェクトに対応するコメント。	不可
[1] オンラインモードではパラメーターの値が変更された後、タイマーは直ぐに 0 にリセットされます。			

オブジェクト

タイマーファンクションブロックには次のオブジェクトがあります。

オブジェクト	説明	値
%TMi.P	プリセット値	上のパラメーター表の説明を参照してください。
%TMi.V	現在値	タイマーが動作しているときに 0 からプリセット値 %TMi.P までインクリメントするワード。この値はプログラムによって読み込みおよびテストができますが、書き込みはできません。 その値はアニメーションテーブルで変更できます。
%TMi.Q	タイマー出力	上の出力表の説明を参照してください。

TON: On-Delay Timer

概要

TON (On-Delay Timer) タイプのタイマーを使用してオンディレー動作を制御します。この遅延はソフトウェアを使用してプログラムできます。

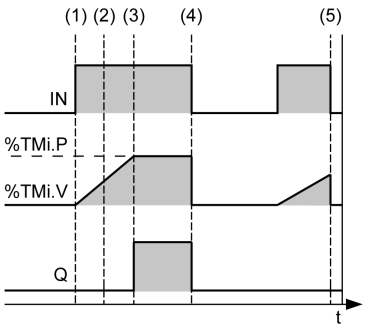
特殊な場合

タイマーファンクションブロックをプログラミングする際の特殊な場合のリストを以下の表に示します。

特殊な場合	説明
コールドリスタート (%S0=1)	値を強制的に 0 にします。%TMi.Q 出力を 0 に設定します。プリセット値は、設定で定義された値にリセットされます。
ウォームリスタート (%S1=1)	タイマー値とタイマーのプリセット値には影響しません。タイマー値は停電時に変化しません。
コントローラー停止	コントローラーを停止しても値は停止されません。
プログラムジャンプ	タイマーブロックを飛び越してもタイマーは停止しません。タイマーは、プリセット値に達するまでインクリメントを続けます (%TMi.P)。その時点で、タイマーブロックの出力 Q に割り当てられた終了ビット (%TMi.Q) は状態遷移します。ただし、ブロック出力に直接配線された関連出力は有効にならず、コントローラーによるスキャンもされません。
ビット %TMi.Q (終了ビット) によるテスト	ビット %TMi.Q はプログラム内で 1 度だけテストしてください。
プリセット %TMi.P 変更	命令を使用したプリセット値の変更または SoMachine Basic を使用した値の調整タイマーが次回有効になる時にのみ効果があります。

タイミングチャート

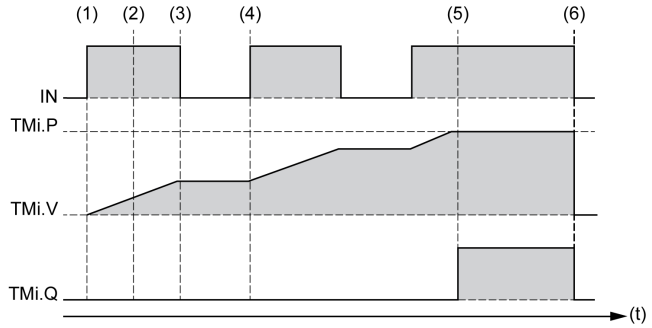
この図は TON タイプのタイマーの動作を示しています。



(1) タイマーは IN 入力の立上がりで開始します
(2) %TMi.V の値は、Time Base パラメーター TB の各パルスごとに 0 から %TMi.P まで 1 ずつ増加します
(3) 値がプリセット値 %TMi.P に達すると、%TMi.Q 出力ビットが 1 に設定されます
(4) IN 入力が 1 の間、%TMi.Q 出力ビットは 1 のままです
(5) タイマーが %TMi.P に達していなくても IN 入力の立下がりが検出されるとタイマーは停止します。%TMi.V を 0 に設定

保持チェックボックスを選択した場合のタイミングチャート

この図は**保持**チェックボックスが選択されている場合の TON タイプのタイマーの動作を示しています。



(1) タイマーは IN 入力の立上がりで開始します
(2) %TMi.V の値は、Time Base パラメーター TB の各パルスごとに 0 から %TMi.P まで 1 ずつ増加します
(3) IN 入力の立下がりでタイマーは停止し、IN 入力の次の立上がりまで変化しません
(4) IN 入力の立上がりで、タイマーは停止した値から再びスタートします
(5) 値がプリセット値 %TMi.P に達すると、%TMi.Q 出力ビットが 1 に設定されます
(6) IN 入力の立下がりが検出されたとき、タイマーがプリセット値 %TMi.P に達していると、%TMi.V 値は 0 に設定されます

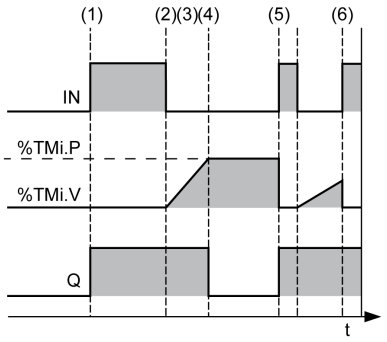
TOF: Off-Delay Timer

概要

TOF (Off-Delay Timer) タイプのタイマーを使用してオフディレー動作を制御します。この遅延はソフトウェアを使用してプログラムできます。

タイミングチャート

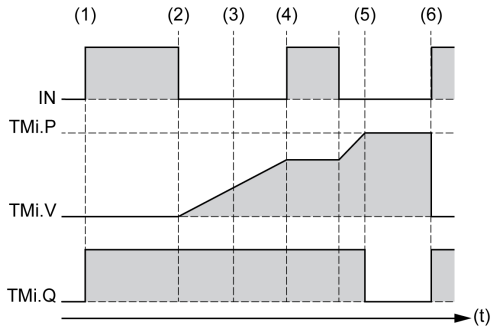
この図は TOF タイプのタイマーの動作を示しています。



(1) IN 入力の立上がりで %TMi.Q は 1 に設定されます
(2) タイマーは IN 入力の立下りで開始します
(3) %TMi.V の値は、Time Base パラメーター TB の各パルスごとにプリセット値 %TMi.P まで 1 ずつ増加します
(4) 値がプリセット値 %TMi.P に達すると、%TMi.Q 出力ビットが 0 にリセットされます
(5) 入力 IN の立上がりで %TMi.V は 0 に設定されます
(6) プリセット値に達していない場合でも、入力 IN の立上がりで %TMi.V は 0 に設定されます

保持チェックボックスを選択した場合のタイミングチャート

この図は**保持**チェックボックスが選択されている場合の TOF タイプのタイマーの動作を示しています。



(1) IN 入力の立上がりで %TMi.Q は 1 に設定されます
(2) タイマーは IN 入力の立下りで開始します
(3) %TMi.V の値は、Time Base パラメーター TB の各パルスごとにプリセット値 %TMi.P まで 1 ずつ増加します
(4) IN 入力の立上りでタイマーは停止し、IN 入力の次の立下りまで変化しません
(5) 値がプリセット値 %TMi.P に達すると、%TMi.Q 出力ビットが 0 にリセットされます
(6) 入力 IN の立上がりで %TMi.V は 0 に、%TMi.Q は 1 に設定されます

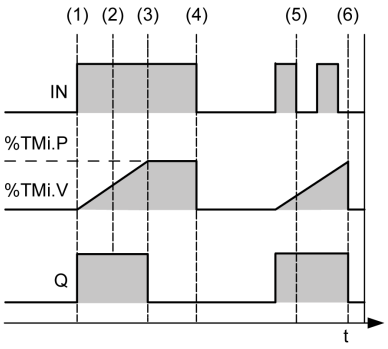
TP: パルスタイマー

概要

TP (パルスタイマー) タイプのタイマーは正確な幅のパルスを生成するために使用されます。この遅延はソフトウェアを使用してプログラムできます。

タイミングチャート

この図は TP タイプのタイマーの動作を示しています。



(1) タイマーは IN 入力の立上がりで開始します。タイマー開始前は現在値 %TMi.V は 0 に設定され、タイマー開始後は %TMi.Q は 1 に設定されます
(2) タイマーの現在値 %TMi.V は、Time Base パラメーター TB の各パルスごとに 0 からプリセット値 %TMi.P まで 1 ずつ増加します
(3) 現在値がプリセット値 %TMi.P に達すると、%TMi.Q 出力ビットが 0 に設定されます
(4) %TMi.V が %TMi.P に等しく、入力 IN が 0 に戻ったとき、現在値 %TMi.V は 0 に設定されます。
(5) このタイマーはリセットできません
(6) %TMi.V が %TMi.P に等しく、入力 IN が 0 であるとき、%TMi.Q は 0 に設定されます

プログラミング例

概要

タイマーファンクションブロックの動作モードを以下に示します。

- TON (Timer On-Delay) (*128* ページ参照) : 特定の入力が有効になってから出力センサーがオンになるまでの時間を指定するために使用します。
- TOF (Timer Off-Delay) (*130* ページ参照) : センサー関連の出力が検出されなくなってから対応する出力がオフになる間の時間を指定するために使用します。
- TP (Timer - Pulse) (*131* ページ参照) : 正確な幅のパルスの生成に使用します。

タイマーの遅延またはパルス周期はプログラムおよびソフトウェアで設定が可能です。

プログラミング

この例は、タイマーファンクションブロックで可逆が可能な命令です。

ラング	可逆命令
0	BLK %TM0 LD %M0 IN OUT_BLK LD Q ST %Q0.0 END_BLK
1	LD [%TM0.V<400] ST %Q0.1
2	LD [%TM0.V>=400] ST %Q0.2

この例は、同じタイマーファンクションブロックで可逆が不可能な命令です。

ラング	不可逆命令
0	LD %M0 IN %TM0
1	LD %TM0.Q ST %Q0.0
2	LD [%TM0.V<400] ST %Q0.1
3	LD [%TM0.V>=400] ST %Q0.2

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

6.3
カウンター (%C)

カウンターファンクションブロックの使用

このセクションではカウンターファンクションブロックのプログラミングガイドラインと説明をします。

このセクションについて

このセクションには次の項目が含まれています。

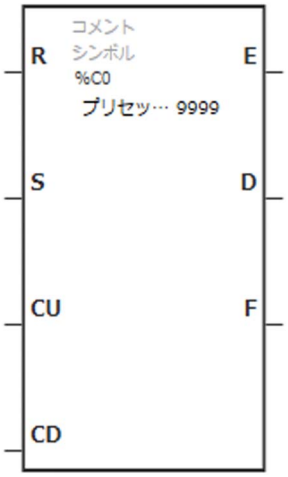
説明

概要

カウンターファンクションブロック 123 はイベントの加算および減算カウントをします。これらの 2 つの動作は同時に行うことができます。

図

次の図は、カウンターファンクションブロックを示しています。



入力

カウンターファンクションブロックには次の入力があります。

ラベル	説明	値
R	リセット入力 (または命令)	リセット入力 (**R**) が 1 のときカウンター (%Ci.V) を 0 に設定します。
S	セット入力 (または命令)	セット入力 (**S**) が 1 のときカウンター (%Ci.V) をプリセット値 (%Ci.P) に設定します。
CU	カウントアップ	カウントアップ入力 (**CU**) の立上がりでカウンター値 (%Ci.V) を 1 増加させます。
CD	カウントダウン	カウントダウン入力 (**CD**) の立上がりでカウンター値 (%Ci.V) を 1 減らします。

出力

カウンターファンクションブロックには次の出力があります。

ラベル	説明	値
E	ダウンカウントオーバーフロー	関連するビット %Ci.E (カウンター 0) は、カウンター が 0 になったときに 1 に設定されます。デクリメント後の場合、カウンターの値は 9999 になります。
D	プリセット出力に達しました	関連するビット %Ci.D (カウント完了) は、%Ci.V = %Ci.P のとき 1 に設定されます。
F	アップカウントオーバーフロー	%Ci.V が 9999 から 0 に変化するとき関連ビット %Ci.F=1 (カウンターフル) となります。(%Ci.V が 0 に達すると 1 にセットされ、カウンターがカウントアップし続けると 0 にリセットされます)

設定

パラメーター

パラメーターを設定するには、ファンクションブロックの設定手順 (*123* ページ参照) に従い、SoMachine Basic オペレーティングガイドのメモリー割り当てモードを読んでください。

カウンターファンクションブロックには次のパラメーターがあります。

パラメーター	説明	値	オンラインモードでの編集
使用	使用済みアドレス	選択されている場合、このアドレスは現在プログラムで使用されています。	不可
アドレス	カウンターオブジェクトアドレス	プログラムで使用できるカウンターオブジェクト数には制限があります。カウンターの最大数については、コントローラーの *プログラミングガイド* を参照してください。	不可
シンボル	シンボル	オブジェクトに対応したシンボル。詳細については、SoMachine Basic オペレーティングガイド シンボルの定義と使用 *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を参照してください。	不可
プリセット	プリセット値	プリセット値 [0 – 9999] で許容される値。初期値は 9999 です。設定されたプリセット値は関連オブジェクト %Ci.P を使用して、読み込み、テスト、および変更ができます	可
コメント	コメント	オブジェクトに対応するコメント。	不可

オブジェクト

カウンターファンクションブロックには次のオブジェクトがあります。

オブジェクト	説明	値
%Ci.V	カウンターの現在値	このワードは入力 (または命令) **CU** および **CD** に従ってインクリメントまたはデクリメントされます (入力テーブル (*134* ページ参照) を参照してください)。読み取り専用。 アニメーションテーブルで編集できます。
%Ci.P	プリセット値	パラメーター表 (*135* ページ参照) を参照してください。 アニメーションテーブルで編集できます。
%Ci.E	空	出力表 (*134* ページ参照) を参照してください。 アニメーションテーブルで編集できます。
%Ci.D	完了	出力表 (*134* ページ参照) を参照してください。 アニメーションテーブルで編集できます。
%Ci.F	フル	出力表 (*134* ページ参照) を参照してください。 アニメーションテーブルで編集できます。

動作

カウンターファンクションブロックの主な動作の説明を次の表に示します。

動作	アクション	結果
リセット	入力 R が状態 1 に設定されます (または R 命令がアクティブ)。	現在値 %Ci.V は 0 に強制されます。出力 %Ci.E、%Ci.D、および %Ci.F は 0 です。リセット入力が優先されます。
セット	入力 S が 1 に設定され (または S 命令がアクティブ)、リセット入力が 0 (または R 命令は非アクティブ) に設定されている場合。	現在値 %Ci.V は %Ci.P の値をとり %Ci.D の出力は 1 に設定されます。

動作	アクション	結果
カウント	カウントアップ入力 CU で立上がりが発生する (または 命令 CU がアクティブ)。	現在値 %Ci.V は 1 ずつ増加します。
	現在値 %Ci.V が、プリセット値 %Ci.P に等しい。	「プリセット到達」出力ビット %Ci.D が 1 に切り替わります。
	現在値 %Ci.V が 9999 から 0 に変わります。	出力ビット %Ci.F (アップカウントオーバーフロー) が 1 に切り替わります。
	カウンターがカウントアップし続けた場合。	出力ビット %Ci.F (アップカウントオーバーフロー) が 0 にリセットされます。
カウントダウン	カウントダウン入力 CD で立上がりが発生する (または 命令 CD CD がアクティブ)。	現在値 %Ci.V は 1 ずつ減少します。
	現在値 %Ci.V が 0 から 9999 に変更されます。	出力ビット %Ci.E (ダウンカウントオーバーフロー) が 1 に切り替わります。
	カウンターがカウントダウンし続けた場合。	出力ビット %Ci.F (ダウンカウントオーバーフロー) が 0 にリセットされます。

特殊な場合

カウンターファンクションブロックの特殊な動作 / 設定のリストを次の表に示します。

特殊な場合	説明
コールドリスタート (%S0=1) または **INIT**	● 現在値 %Ci.V は 0 に設定されます。 ● 出力ビット %Ci.E、%Ci.D、および %Ci.F は 0 に設定されます。 ● プリセット値は設定で定義した値に初期化されます。
コントローラー停止のウォームリスタート (%S1=1)	カウンター (%Ci.V) の現在値には影響しません。
プリセット %Ci.P 変更	プリセット値を命令で修正または調整することによって、ブロックがアプリケーションによって処理されたときに有効になります (入力のうち 1 つの有効化)。

注記：**INIT** は %S0=1 と同じです。

プログラミング例

概要

例として最大 5000 項目をカウントするカウンターを以下に示します。入力 %I0.2 の各パルス (ビットメモリー %M0 が 1 に設定されている場合) は、カウンターファンクションブロック %C8 を最終プリセット値 (ビット %C8.D=1) までインクリメントします。カウンターは入力 %I0.1 によりリセットされます。

プログラミング

この例は、カウンターファンクションブロックで可逆が可能な命令です。

ラング	可逆命令
0	BLK %C8 LD %I0.1 R LD %I0.2 AND %M0 CU END_BLK
1	LD %C8.D ST %Q0.0

この例は、同じカウンターファンクションブロックで可逆が不可能な命令です。

ラング	不可逆命令
0	LD %I0.1 R %C8
1	LD %I0.2 AND %M0 CU %C8
2	LD %C8.D ST %Q0.0

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

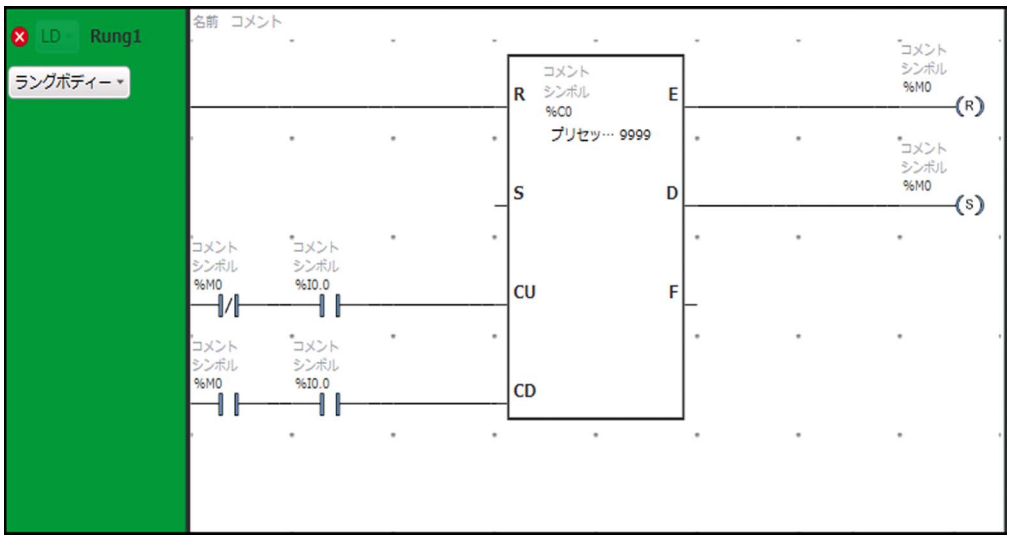
設定

設定中にパラメーターを入力してください。

プリセット値 (%Ci.P) : この例では 5000 に設定します。

加算および減算カウンターの例

次に、カウンターファンクションブロックを示します。



この例では、%M0 はインクリメント (%M0 = False) とデクリメント (%M0 = True) の命令です。カウンターは %I0.0 の立上がりをカウントします。%M0 が False の場合、%I0.0 の各立上がりで、%C1.V はプリセット値 %C1.P に達するまでインクリメントされ、完了インジケータ %C1.D は TRUE に切り替わります。%C1.D 出力は %M0 をセットし、命令をデクリメント命令にします。その後、%I0.0 の各立上がりで %C1.V が 0 になるまでデクリメントされます。空インジケーター (%C1.E) がオンになり、%M0 (インクリメント命令) がリセットされます。

6.4
メッセージ (%MSG) と Exchange (EXCH)

Message ファンクションブロックの使用

このセクションでは Message ファンクションブロックのプログラミングガイドラインと説明をします。

このセクションについて

このセクションには次の項目が含まれています。

概略

概要

ロジックコントローラーは、Modbus プロトコルで通信するように設定できます、またキャラクターモード (ASCII) でメッセージを送受信することもできます。

SoMachine Basic は通信に以下のファンクションを提供します。
- メッセージ送受信用の **Exchange** (EXCH) 命令
- データ交換制御用の Message ファンクションブロック (%MSG)

ロジックコントローラーは **Exchange** 命令を処理するときに、指定されたポート用に設定されたプロトコルを使用します。各通信ポートには異なるプロトコルを割り当てることができます。通信ポートは **Exchange** 命令 (EXCH1、EXCH2) または Message ファンクションブロック (%MSG1、%MSG2) にポート番号を付加することによってアクセスできます。

ロジックコントローラーは、EXCH3 命令と %MSG3 ファンクションブロックを利用して Ethernet ネットワーク上で Modbus TCP メッセージングを実行します。

次の表は、コントローラーの通信ポートへのアクセスに使用される **Exchange** 命令と Message ファンクションブロックを示しています。

通信ポート	Exchange 命令	Message ファンクションブロック
シリアルライン 2 つ	EXCH1	%MSG1
	EXCH2	%MSG2
シリアルライン 1 つおよび Ethernet 1 つ	EXCH1	%MSG1
	EXCH3	%MSG3

Exchange 命令

Exchange 命令は ASCII または Modbus デバイスとロジックコントローラー間の送受信を可能にします。送受信用に制御情報とデータを含んだワードテーブル (%MWi:L) を定義します。送信テーブルの設定 (*146* ページ参照) を参照してください。メッセージ交換は **Exchange** 命令を使用して実行されます。

構文

Exchange 命令の形式は次のとおりです。

[EXCHx %MWi:L]

x = ポート番号、L = ワードテーブルのワードの総数。

ロジックコントローラーは、2 番目の **Exchange** 命令を開始する前に、1 番目の **Exchange** 命令を終了してください。複数のメッセージを送信するときには、Message ファンクションブロックを使用してください。

ASCII プロトコル

ASCII プロトコルは単純なキャラクタモードプロトコルをロジックコントローラーに提供し、単純なデバイスでデータを送受信します。このプロトコルは **Exchange** 命令に対応し、Message ファンクションブロックを使用して制御されます。

ASCII プロトコルでは次の 3 種類の通信が可能です。
- 送信のみ
- 送信 / 受信
- 受信のみ

Modbus プロトコル

シリアルリンクの場合、Modbus プロトコルはマスター／スレーブのプロトコルで、1 つのマスターがスレーブからの応答をリクエスト、またはリクエストに基づき動作します。Ethernet は複数のマスター (クライアント) と 1 つのスレーブ (サーバー) の通信をサポートします。各スレーブには固有のアドレスが必要です。マスターは個々のスレーブにアドレス指定することも、すべてのスレーブに一斉送信メッセージを送信することもできます。スレーブは個別に宛てられたクエリにメッセージ (応答) を返します。マスターからの一斉送信クエリに応答は返されません。

Modbus マスターモードでは、コントローラーは Modbus クエリをスレーブに送信し、応答を待つことができます。Modbus マスターモードは、**Exchange** 命令でのみサポートされています。Modbus のマスターモードでは、Modbus ASCII と RTU の両方がサポートされています。

Modbus スレーブモードでは、コントローラーは Modbus マスターからの標準 Modbus クエリに応答することができます。

Modbus プロトコルの詳細については、*http://www.modbus.org* の *Modbus アプリケーションプロトコル* のドキュメントを参照してください。

Modbus スレーブ

Modbus プロトコルは OSI モデル形式の 2 つのデータリンク層をサポートしています。ASCII および RTU。それぞれ物理レイヤーの実装で定義され、ASCII では 7 データビット、RTU では 8 データビットを使用します。

Modbus ASCII モードを使用する場合、メッセージの各バイトは 2 つの ASCII 文字として送信されます。Modbus ASCII フレームは開始文字 (':') で始まり、2 つの終了文字 (CR と LF) で終わります。フレーム文字の終わりは初期設定で 0x0A (LF) になります。Modbus ASCII フレームのチェック値は、開始文字と終了文字を除いて、フレームの 2 の補数です。

Modbus RTU モードでは、送信前にメッセージを再フォーマットしません。ただし、CRC として指定された異なるチェックサム計算モードを使用します。

Modbus データリンク層には次の制限があります。

- アドレス 1-247
- ビット : リクエストに応じて 128 ビット
- ワード : リクエストに応じて 16 ビットの 125 ワード

説明

概要

Message ファンクションブロック  はデータ交換を管理し、3 つのファンクションを備えています。

- 通信エラーチェック
  エラーチェックでは、各 **Exchange** テーブルのサイズが検証され、設定に関連する Exchange の有効性が検証されます。
- 複数のメッセージの調整
  複数のメッセージを送信するときの調整を確実にするため、Message ファンクションブロックは、直前のメッセージの完了時期を判断するのに必要な情報を提供します。
- 優先メッセージの送信
  Message ファンクションブロックは、緊急メッセージの即時送信を可能にするために、進行中のメッセージ送信を停止します。

Message ファンクションブロックのプログラミングはオプションです。

エラーが検出されると、Exchange ブロック EXCH1、EXCH2、および EXCH3 のシステムワード %SW63、%SW64、および %SW65 にコードが書き込まれます。詳細については、コントローラーの *プログラミングガイド* を参照してください。

図

次の図は Message ファンクションブロックを示しています。

入力

Message ファンクションブロックには次の入力があります。

ラベル	説明	値
R	リセット入力 (%MSGx.R)	通信を再初期化するには 1 に設定します。 ● 通信完了 (%MSGx.D) 出力は 1 に設定されます。 ● 通信エラー検出 (%MSG.E) 出力は 0 に設定されます。 ● アクティブな通信ファンクションブロック (%READ_VAR、%WRITE_VAR、など) にエラーが設定されています ● 他の Modicon M221 ロジックコントローラーへのアクティブな TCP 接続は切断されます。 **注記：** マスタータスクサイクル中に通信ポート上で 1 度に 1 つの Message ファンクションブロック、EXCH 命令、または通信ファンクションブロックのみをアクティブにすることができます。複数の通信ファンクションブロック、MSG、または EXCH 命令を同じ通信ポートで同時に使用しようとすると、ファンクションブロックはエラーコードを返します。従って通信ファンクションブロック、Message ファンクションブロック、または EXCH 命令を開始する前に、通信ポートでアクティブな通信 (%MSGx.D が TRUE) が進行中でないことを確認してください。さらに IOScanner が通信ポートでアクティブでないことを確認します。 **注記：** IOScanner は %MSG ファンクションブロックの出力を更新しません。従って %MSG.D ビットは IOScanner のファンクションとは無関係です。

出力

Message ファンクションブロックには次の出力があります。

ラベル	説明	値
D	通信完了 (%MSGx.D)	状態 1: ● 送信終了 (送信の場合) ● 受信終了 (終了文字受信) ● エラー ● ブロックをリセットします 状態 0 : リクエストが進行中です。
E	通信エラーが検出されました (%MSGx.E)	状態 1: ● 未定義コマンド ● テーブルが正しく設定されていません ● 不適切な文字を受信しました (速度、パリティなど) ● 受信テーブルが一杯です (更新されません) 状態 0 : メッセージ長は正しいです。リンクが確立されました。 通信エラーが検出されたときにシステムワードに書き込まれるエラーコードについては、下の表を参照してください。

通信エラーコード

通信エラーが検出されたときにシステムワードに書き込まれるエラーコードについては、下の表に示します。

システムワード	ファンクション	説明
%SW63	EXCH1 ブロックエラーコード	EXCH1 エラーコード : 0 - オペレーション正常終了 1 - 送信バイト数が制限を越えています (> 255)。 2 - 不十分な送信テーブル 3 - 不十分なワードテーブル 4 - 受信テーブルがオーバーフロー 5 - タイムアウト 6 - 送信 7 - テーブルに不正なコマンドがあります。 8 - ポートが未設定、または無効 9 - 受信エラー : 不正、または破損した受信フレームのエラーです。物理パラメーター (パリティ、データビット、ボーレート等) が正しく設定されていないか、不安定な接続による信号劣化が原因の可能性があります。 10 - 受信に %KW は使えません。 11 - 送信オフセットが送信テーブルより大きいです。 12 - 受信オフセットが受信テーブルより大きいです。 13 - コントローラーによって EXCH 処理が停止
%SW64	EXCH2 ブロックエラーコード	EXCH2 エラーコード :%SW63 を参照してください。

システムワード	ファンクション	説明
%SW65	EXCH3 ブロックエラーコード	1-4、6-13:%SW63 を参照してください。(エラーコード 5 は無効のため、以下に示されている Ethernet 固有エラーコード 109、および 122 に置き換えられます)。 Ethernet 固有エラーコードを以下に示します。 101 - 無効な IP アドレス 102 - TCP 接続がありません。 103 - ソケットがありません (すべての接続チャンネルが使われています)。 104 - ネットワークがダウンしています。 105 - ネットワークに到達できません。 106 - リセット時のネットワーク接続切断 107 - ピアデバイスによる接続の中断 108 - ピアデバイスによる接続のリセット 109 - 接続タイムアウト 110 - 接続拒否 111 - ホストがダウンしています。 120 - 不正なインデックス (リモートデバイスは設定テーブルでインデックスされていません)。 121 - システムエラー (MAC、チップ、重複 IP) 122 - データ送信後に、処理タイムアウトを受信 123 - Ethernet 初期化中

設定

検出されたエラー

Exchange 命令を使用するときにエラーが検出された場合、ビット %MSGx.D および %MSGx.E は 1 に設定され、システムワード %SW63 はポート 1 のエラーコードを含んでいます。また %SW64 にはポート 2 のエラーコードが含まれています。ロジックコントローラーのプログラミングガイドのシステムワードの章を参照してください。

動作

Message ファンクションブロックの主な動作の説明を次の表に示します。

動作	アクション	結果
リセット	入力 R は状態 1 に設定されます (または R 命令がアクティブ)。	● 送信中のメッセージはすべて送信停止されます。 ● 通信エラー出力は 0 にリセットされます。 ● 完了ビットは 1 に設定されます。 新しいメッセージを送信できます。
通信完了	出力 D は状態 1 に設定されます。	ロジックコントローラーは次のメッセージを送信する準備ができています。%MSGx.D ビットを使用すると、複数のメッセージを送信した場合にメッセージが失われるのを防ぐことができます。
通信エラーを検出	通信エラー出力は 1 にセットされます。 ● 通信プログラミングエラーまたはメッセージ送信エラーが原因です。 ● **Exchange** 命令 (ワード 1、最下位バイト) に関連付けられたデータブロックに定義されたバイト数が 128 より大きい場合 (FA では 16 進数で +80)。 ● Modbus メッセージを Modbus デバイスに送信する際に問題がある場合。この場合、配線の確認および送信先のデバイスが Modbus 通信に対応していることの確認をしてください。	

特殊な場合

この表は Message の動作の特殊な場合のリストを示しています。

特殊な場合	説明
コールドリスタート (%S0=1) または **INIT**	通信の再初期化を強制します。
ウォームリスタート (%S1=1)	影響はありません。
コントローラー停止	メッセージ送信中の場合、コントローラーは送信を停止し、出力 %MSGx.D および %MSGx.E を再初期化します

注記：**INIT** は %S0=1 と同じです。

制限事項

次の制限事項に注意してください。

- 電源投入時またはリセット時にのみ、ポート 2 (ASCII プロトコル用) のタイプ (%SW7) と状態が確認されます
- 電源投入時またはリセット時にポート 2 (Modbus プロトコル用) の存在と設定 (RS-485) が確認されます
- SoMachine Basic が接続されている場合は**ポート 1 のすべてのメッセージ処理が中止されます**
- **Exchange** 命令は、アクティブな Modbus スレーブ処理を中止します
- エラーが検出されても **Exchange** 命令の処理は再試行されません
- リセット入力 (R) を使用して **Exchange** 命令の受信処理を中止できます
- Modbus プロトコルの場合、**Exchange** 命令にはタイムアウトが設定されています
- 複数のメッセージは %MSGx.D により制御されます

送信 / 受信テーブルの設定

送信フレームまたは受信フレームの最大サイズは次のとおりです。

- Modbus プロトコルの場合は 250 バイト。
- ASCII プロトコルの場合は 256 バイト。

Exchange 命令に関連するワードテーブルは、制御テーブル、送信テーブル、および受信テーブルで構成されています。

	最上位バイト		**最下位バイト**	
	Modbus	**ASCII**	**Modbus**	**ASCII**
制御テーブル	コマンド		長さ (送信 / 受信)	
	Rx オフセット	予約済み (0)	Tx オフセット	予約済み (0)
送信テーブル	送信バイト 1		送信バイト 2	
	...		...	
			送信バイト n	
	送信バイト n+1			
受信テーブル	受信バイト 1		受信バイト 2	
	...		...	
			受信バイト p	
	受信バイト p+1			

注記：個々のスレーブへのクエリに加えて、Modbus マスタコントローラーはすべてのスレーブに対して一斉送信クエリを開始できます。一斉送信クエリの場合の **コマンド** バイトは 00 に、またスレーブアドレスは 0 に設定してください。

ASCII プロトコルの制御テーブル

長さバイトには、送信テーブルの長さ (最大バイト数 250) が含まれます。受信がリクエストされた場合、受信終了時に受信した文字数で上書きされます。

コマンドバイトには、次のいずれかを含めてください。

- 0: 送信のみ
- 1: 送信 / 受信
- 2: 受信のみ

Modbus プロトコルの制御テーブル

長さバイトには、送信テーブルの長さ (最大バイト数 250) が含まれます。受信がリクエストされた場合、受信終了時に受信した文字数で上書きされます。

このパラメーターは送信テーブルのバイト長です。**Tx オフセット**パラメーターが 0 に等しい場合、このパラメーターは送信フレームの長さに等しいです。**Tx オフセット**パラメーターが 0 でなければ、送信テーブルの 1 バイト (オフセット値によって示される) は送信されず、このパラメーターはフレーム長に 1 を加えたものに等しいです。

Modbus RTU リクエスト (一斉送信を除く) の場合の**コマンド**バイトは常に 1 (**Tx** と **Rx**) に等しくしてください。一斉送信の場合は 0 にしてください。

Tx オフセットバイトはバイト送信テーブルのランクを含みます。ランクはバイトを送信する際に無視するバイトを指定します (1 は第 1 バイト、2 は第 2 バイトなどの数値です)。これは、Modbus プロトコル内のバイト / ワード値に関連する問題を処理するために使用されます。例えば、このバイトに 3 が含まれる場合、第 3 バイトは無視され、テーブルの第 4 バイトが第 3 バイトとして送信されます。

Rx オフセットバイトは受信テーブルのランクを含みます。ランクはパケットを送信する際に追加するバイトを指定します (1 は第 1 バイト、2 は第 2 バイトなどの数値です)。これは、Modbus プロトコル内のバイト / ワード値に関連する問題を処理するために使用されます。例えば、このバイトに 3 が含まれる場合、テーブル内の第 3 バイトは 0 で埋められ、受信された第 3 バイトはテーブル内の 4 番目の位置に入力されます。

ASCII プロトコルの送信 / 受信テーブル

送信専用モードでは **Exchange** (EXCH) 命令を実行する前に %MW タイプの制御テーブルと送信テーブルは埋められます。送信専用モードにおいて、文字を受信するためのスペースは必要ありません。すべてのバイトが送信されると %MSGx.D は 1 にセットされ、新しい **Exchange** (EXCH) 命令が実行できます。

送信 / 受信モードの場合、制御テーブルと送信テーブルは **Exchange** (EXCH) 命令を実行する前に埋められます。タイプは %MW となります。送信テーブルの終端には、最大 256 受信バイトのスペースが必要です。すべてのバイトが送信されると、ロジックコントローラーは受信モードに切り替わり、受信するバイトを待機します。

受信専用モードの場合、制御テーブルは **Exchange** (EXCH) 命令を実行する前に埋められます。タイプは %MW となります。制御テーブルの終端には、最大 256 受信バイトのスペースが必要です。ロジックコントローラーは直ちに受信モードに入り、任意のバイトの受信を待機します。

使用されたフレームバイトの末尾が受信された場合や受信テーブルがいっぱいになった場合に受信は終了します。この場合、検出されたエラーコード (受信テーブルのオーバーフロー) がシステムワード %SW63 および %SW64 に含まれます。ゼロ以外のタイムアウトが設定されている場合、タイムアウト完了で受信終了となります。タイムアウト値がゼロに設定されている場合、受信タイムアウトはありません。従って、受信を停止するには %MSGx.R 入力をアクティブにしてください。

Modbus プロトコルの送信 / 受信テーブル

いずれかのモード (Modbus ASCII または Modbus RTU) を使用する場合、送信テーブルは **Exchange** (EXCH) 命令を実行する前にリクエストで一杯になります。実行時にロジックコントローラーは、データリンク層が何であるかを判断し、送信および応答を処理するために必要なすべての変換を実行します。開始、終了、および文字チェックは送受信テーブルに格納されません。

すべてのバイトが送信されると、ロジックコントローラーは受信モードに切り替わり、受信するバイトを待機します。

受信は下記のいずれかで完了します。
- 文字またはフレームのタイムアウトが検出された場合
- ASCII モードでフレーム終了文字を受信した場合
- 受信テーブルがいっぱいになった場合

送信バイト x エントリは、送信される Modbus プロトコル (RTU エンコーディング) データを含みます。通信ポートが Modbus ASCII に設定されている場合、正しいフレーミング文字が送信に追加されます。第 1 バイトにはデバイスアドレス (固有または一斉送信)、第 2 バイトにはファンクションコードが含まれます。残りにはそのファンクションコードに関連付けられた情報が含まれます。

注記：これは一般的な使用であり、すべての可能性を表しているものではありません。送信されるデータの検証は実行されません。

受信バイト x エントリは、受信される Modbus プロトコル (RTU エンコーディング) データを含みます。通信ポートが Modbus ASCII に設定されている場合、正しいフレーミング文字は応答から削除されます。第 1 バイトにはデバイスアドレス、第 2 バイトにはファンクションコード (または応答コード) が含まれます。残りにはそのファンクションコードに関連付けられた情報が含まれます。

注記：これは一般的な使用であり、すべての可能性を表しているものではありません。受信されているデータの検証は、チェックサム検証以外実行されません。

プログラミング例

概要

以下は Message ファンクションブロックのプログラミング例です。

複数の連続したメッセージの送信のプログラミング

Exchange 命令を実行すると、アプリケーションプログラムの Message ファンクションブロックが有効になります。Message ファンクションブロックがまだ有効でない場合 (%MSGx.D = 1)、メッセージが送信されます。複数のメッセージが同じサイクルで送信された場合、最初のメッセージのみが送信されます

ポート 1 で 2 つのメッセージを連続して送信する例

ラング	可逆命令	コメント
0	LD %M142 [%MW2:=16#0106] [%MW3:=0] [%MW4:=16#0106] [%MW5:=4] [%MW6:=7]	アドレス 1 のスレーブのレジスター 4 に値 7 を書き込みます。 [%MW2:=16#0106]: コマンドコード :16 進数 01、送信長 :16 進数 06 [%MW3:=0]: 受信または送信オフセットなし [%MW4:=16#0106]: スレーブアドレス 16 進数 01、ファンクションコード : 16 進数 06 (シングルレジスター書き込み) [%MW5:=4]: レジスターアドレス [%MW6:=7]: 書き込み値
1	LD %MSG1.D AND %M0 [EXCH1 %MW2:8] R %M0	%MSG2.D: ポートがビジーであるかどうかを検出し、それによって複数のメッセージの調整を管理します。
2	LDR %I0.0 AND %MSG1.D [EXCH1 %MW2:8] S %M0	–

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

再初期化 Exchange プログラミング

Exchange は入力 (または命令) R を有効にすることでキャンセルされます。この入力は通信の初期化、出力 %MSGx.E を 0 にリセット、および %MSGx.D を 1 に出力します。エラーが検出された場合、Exchange の再初期化が可能です。

Exchange の再初期化の例

ラング	可逆命令	コメント
0	BLK %MSG1 LD %M0 R END_BLK	–

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

ASCII の例

アプリケーションの書き込み

ASCII アプリケーションの例

ラング	命令	コメント
0	LD 1 [%MW10:=16#0104] [%MW11:=16#0000] [%MW12:=16#4F4B]	[%MW10:=16#0104]: コマンドコード :16 進数 01、送信長 :16 進数 04 [%MW11:=16#0000]:0000:Null [%MW12:=16#4F4B]:OK **注記**：テーブルには 8 つの要素があります。
1	LD 1 AND %MSG2.D [EXCH2 %MW10:8]	
2	LD %MSG2.E ST %Q0.0 END	

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

SoMachine Basic を使用して 3 つのラングを持つプログラムを作成します

- まず、**Exchange** 命令に使用するコントロールテーブルと送信テーブルを初期化します。この例では、データの送受信の両方にコマンドが設定されています。送信するデータの量はアプリケーションで定義されているように 4 バイトに設定され、設定で定義したフレーム文字の末尾が続きます。開始文字と終了文字はアニメーションテーブルには表示されず、データ文字のみが表示されます。いずれの場合も、これらの文字は自動的に転送される、または受信時に %SW63 および %SW64 によって確認されます。
  注記：設定で定義された終了文字は、フレームの最後に自動的に送信されます。例えば、最初の終了文字を 10 に設定し 2 番目の終了文字を 13 に設定した場合、16#0A0D (ASCII コード、0A = LF および 0D = CR) がフレームの最後に送信されます。
- 次に %MSG2 に関連するステータスビットをチェックし、EXCH2 命令はポートが準備完了である場合にのみ使用します。EXCH2 命令では 8 ワードの値が指定されます。2 つの制御ワード (%MW10 および %MW11)、送信情報 (%MW12 および %MW13) に使用する 2 つのワード、およびデータを受信する 4 つのワード (%MW14 から %MW17) があります。
- 最後に %MSG2 のエラーステータスの検出はローカルベースコントローラー I/O の 1 番目の出力ビットで検出および格納されます。%SW64 を追加することでより正確なエラー処理ができます。

アニメーションテーブルの初期化

オンラインモードでアニメーションテーブルを初期化する例

アドレス	値	形式
%MW10	0104	16 進数
%MW11	0000	16 進数
%MW12	4F4B	16 進数
%MW13	0A0D	16 進数
%MW14	AL	ASCII
%MW15	OH	ASCII
%MW16	A	ASCII

使用できる形式を表示するには、アニメーションテーブルの**値**ボックスを右クリックします。

最後にアプリケーションをコントローラーにダウンロードし実行します。ワード %MW10 から %MW16 をアニメーション化して表示するためにアニメーションテーブルを初期化します。この情報はロジックコントローラーと通信され、アニメーションテーブルに表示されます。

Modbus 標準リクエストおよび例

Modbus マスター : N ビットの読み込み

このテーブルはリクエスト 01 と 02 を表します (01 は出力またはビットメモリー、02 は入力ビット)。

	テーブル インデックス	最上位バイト	最下位バイト
コントロールテーブル	0	01 (送信 / 受信)	06 (送信長)(1)
	1	03 (受信オフセット)	00 (送信オフセット)
送信テーブル	2	スレーブ @(1...247)	01 または 02 (リクエストコード)
	3	スレーブで読み込む第 1 ビットのアドレス	
	4	N_1 = 読み込むビット数	
受信テーブル (応答後)	5	スレーブ @(1...247)	01 または 02 (応答コード)
	6	00 (Rx オフセット動作によって追加されたバイト)	N_2 = 読み込むデータバイト数 = [1+(N_1-1)/8]、結果は除算の整数部分です。
	7	1 バイトに拡張された第 1 ビットの値 (値 00 または 01)	1 バイトに拡張された第 2 ビットの値 (N_2>1 の場合)
	8	1 バイトに拡張された第 3 ビットの値 (N_1>1 の場合)	–
	...	...	...
	(N_2/2)+6 (N_2 が偶数の場合) (N_2/2+1)+6 (N_2 が奇数の場合)	1 バイトに拡張された N_2 th ビットの値 (N_1>1 の場合)	–
(1) このバイトは応答後に送信された文字列の長さも受け取ります。			

Modbus マスター :N ワードの読み込み

このテーブルはリクエスト 03 と 04 を表します (03 は出力またはワードメモリー、04 は入力ワード)。

	テーブル インデックス	最上位バイト	最下位バイト
コントロールテーブル	0	01 (送信 / 受信)	06 (送信長)(1)
	1	03 (受信オフセット)	00 (送信オフセット)
送信テーブル	2	スレーブ @(1...247)	03 または 04 (リクエストコード)
	3	読み込む第 1 ワードのアドレス	
	4	N = 読み込むワード数	
受信テーブル (応答後)	5	スレーブ @(1...247)	03 または 04 (応答コード)
	6	00 (Rx オフセット動作によって追加されたバイト)	2*N (読み込みバイト数)
	7	第 1 ワードの読み込み	
	8	第 2 ワードの読み込み (N> 1 の場合)	
	...	...	
	N+6	ワード N の読み込み (N> 2 の場合)	
(1) このバイトは応答後に送信された文字列の長さも受け取ります。			

注記 : 受信オフセットの 3 は、受信テーブルの 3 番目の位置に 1 バイト (値 = 0) を追加します。これにより、読み込まれたバイト数とこの表の読み込みワードの値の正しい位置が保証されます。

Modbus マスター : ビットの書き込み

このテーブルはリクエスト 05 (シングルビットの書き込み : 出力またはメモリー) を表します。

	テーブルインデックス	最上位バイト	最下位バイト
コントロールテーブル	0	01 (送信 / 受信)	06 (送信長)(1)
	1	00 (受信オフセット)	00 (送信オフセット)
送信テーブル	2	スレーブ @(1...247) または一斉送信の場合は 0	05 (リクエストコード)
	3	インデックスワード 4 の MSB に書き込む値。0xFF または 0x00(2) のいずれか。	
	4	スレーブに書き込むビット値 (16#0000 = False および 16#FF00 = True)	
受信テーブル (応答後)	5	スレーブ @(1...247)	05 (応答コード)
	6	書き込まれたビットのアドレス	
	7	書き込まれた値	
(1) このバイトは応答後に送信された文字列の長さも受け取ります。 **(2)** 1 を書き込むビットには、送信テーブルの関連するワードの値を FF00h にし、0 を書き込むビットには値を 0 にしてください。			

注記 :

- このリクエストではオフセットを使用する必要はありません。
- 通常の場合、応答フレームはこのリクエストフレームと同じです。

Modbus マスター : ワードの書き込み

このテーブルはリクエスト 06 (シングルワードの書き込み : 出力またはメモリー) を表します。

	テーブルインデックス	最上位バイト	最下位バイト
コントロールテーブル	0	01 (送信 / 受信)	06 (送信長)(1)
	1	00 (受信オフセット)	00 (送信オフセット)
送信テーブル	2	スレーブ @(1...247) または一斉送信の場合は 0	06 (リクエストコード)
	3	書き込むワードのアドレス	
	4	読み込むワードの値	
受信テーブル (応答後)	5	スレーブ @(1...247)	06 (応答コード)
	6	書き込まれたワードのアドレス	
	7	書き込まれた値	
(1) このバイトは応答後に送信された文字列の長さも受け取ります。			

注記 :

- このリクエストではオフセットを使用する必要はありません。
- 通常の場合、応答フレームはこのリクエストフレームと同じです。

Modbus マスター :N ビットの書き込み

このテーブルはリクエスト 15 (ビット N の書き込み : 出力またはメモリー) を表します。

	テーブルインデックス	最上位バイト	最下位バイト
コントロールテーブル	0	01 (送信 / 受信)	8 + バイト数 (送信)
	1	00 (受信オフセット)	07 (送信オフセット)

	テーブルインデックス	最上位バイト	最下位バイト
送信テーブル	2	スレーブ @(1...247) またはー斉送信の場合は 0	15 (リクエストコード)
	3	書き込む第 1 ビットのアドレス	
	4	N_1 = 書き込むビット数	
	5	00 (オフセットの影響でバイトは送信されません)	N_2 = 書き込むデータバイト数 = [1+(N_1-1)/8]、 結果は除算の整数部分です。
	6	第 1 バイトの値	第 2 バイトの値
	7	第 3 バイトの値	第 4 バイトの値
	...	...	...
	(N_2/2)+5 (N_2 が偶数の場合) (N_2/2+1)+5 (N_2 が奇数の場合)	N_2^{th} バイトの値	
受信テーブル (応答後)	–	スレーブ @(1...247)	15 (応答コード)
	–	書き込まれた第 1 ビットのアドレス	
	–	書き込まれたビット数 (= N_1)	

注記：送信オフセット = 7 で、送信フレーム内の第 7 バイトが取り除かれます。これにより送信テーブルのワードの値が正しく通信されます。

Modbus マスター :N ワードの書き込み

このテーブルはリクエスト 16 を表します。

	テーブルインデックス	最上位バイト	最下位バイト
コントロールテーブル	0	01 (送信 / 受信)	8 + (2*N) (送信長)
	1	00 (受信オフセット)	07 (送信オフセット)
送信テーブル	2	スレーブ @(1...247) またはー斉送信の場合は 0	16 (リクエストコード)
	3	書き込む第 1 ワードのアドレス	
	4	N = 書き込むワード数	
	5	00 (オフセットの影響でバイトは送信されません)	2*N = 書き込むバイト数
	6	書き込む第 1 ワードの値	
	7	書き込む 2 番目の値	
	...	...	
	N+5	書き込む N の値	
受信テーブル (応答後)	N+6	スレーブ @(1...247)	16 (応答コード)
	N+7	書き込まれた第 1 ワードのアドレス	
	N+8	書き込まれたワード数 (= N)	

注記：送信オフセット = 7 で、送信フレーム内の第 7 バイトが取り除かれます。これにより送信テーブルのワードの値が正しく通信されます。

Modbus リクエスト : デバイス識別情報の読み込み

このテーブルは、リクエスト 43 (デバイス識別の読み込み) を表します。

ラング	命令	コメント
0	LD 1 [%MW800:=16#0105] [%MW801:=16#0000] [%MW802:=16#032B] [%MW803:=16#0E01] [%MW804:=16#0000]	[%MW800:=16#0105]: 標準 Modbus ヘッダー [%MW801:=16#0000]: 送信および受信オフセットなし [%MW802:=16#032B]: スレーブアドレス、ファンクションコード [%MW803:=16#0E01]:MEI タイプ、デバイス ID コードの読み取り [%MW804:=16#0000]: オブジェクト ID、未使用

Modbus リクエスト : 診断

このテーブルはリクエスト 8 を表します (診断用)

ラング	命令	コメント
0	LD 1 [%MW1000:=16#0106] [%MW1001:=16#0000] [%MW1002:=16#0308] [%MW1003:=16#0000] [%MW1004:=16#1234]	[%MW1000:=16#0106]: 標準 Modbus ヘッダー [%MW1001:=16#0000]: 送信および受信オフセットなし [%MW1002:=16#0308]: スレーブアドレス、ファンクションコード [%MW1003:=16#0000]: サブファンクションコード [%MW1004:=16#1234]: すべてのデータ スレーブはリクエストのコピーを返します。このモードはエコーモードまたはミラーモードと呼ばれます。

例 1:Modbus アプリケーションの書き込み

マスタープログラム :

ラング	命令	コメント
0	LD 1 [%MW0:=16#0106] [%MW1:=16#0300] [%MW2:=16#0203] [%MW3:=16#0000] [%MW4:=16#0004]	[%MW0:=16#0106]: 送信長 = 6 [%MW1:=16#0300]: オフセット受信 = 3、オフセット送信 = 0 %MW2 から %MW4 : 送信 [%MW2:=16#0203]: スレーブ 2、ファンクション 3 (マルチワード読み込み) [%MW3:=16#0000]: スレーブで読み込む第 1 ワードアドレス : アドレス 0 番地 [%MW4:=16#0004]: 読み込むワード数 : 4 ワード (%MW0 から %MW3)
1	LD 1 AND %MSG2.D [EXCH2 %MW0:11]	–
2	LD %MSG2.E ST %Q0.0 END	–

スレーブプログラム :

ラング	命令	コメント
0	LD 1 [%MW0:=16#6566] [%MW1:=16#6768] [%MW2:=16#6970] [%MW3:=16#7172] END	–

注記 : ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

SoMachine Basic を使用して、マスターとスレーブの両方のアプリケーションプログラムを作成します。スレーブの場合、いくつかのワードメモリーを既知の値に書き込みます。マスターでは、%MW0 から始まる Modbus アドレス 2 のスレーブから 4 ワード読み込むために、**Exchange** 命令のワードテーブルが初期化されます。

注記：Modbus マスター %MW1 に設定された受信オフセットの使用に注意してください。オフセットの 3 は、受信テーブルの 3 番目の位置に 1 バイト (値 = 0) を追加します。ワード境界上に正しく収まるように、マスターのワードを整列させます。このオフセットがないとデータの各ワードは **Exchange** ブロックで 2 ワードに分割されます。このオフセットは便宜上使用されるものです。

EXCH2 命令を実行する前に、アプリケーションは %MSG2 に関連する通信ビットをチェックします。最後に %MSG2 のエラーステータスはローカルベースコントローラー I/O の 1 番目の出力ビットにより検出および格納されます。より正確にチェックするために %SW64 を使用したエラーチェックを追加することもできます。

受信テーブル部分に対応してオンラインモードで行うアニメーションテーブルの初期化

アドレス	値	形式
%MW5	0203	16 進数
%MW6	0008	16 進数
%MW7	6566	16 進数
%MW8	6768	16 進数
%MW9	6970	16 進数
%MW10	7172	16 進数

各ロジックコントローラーをダウンロードして実行するように設定したら、マスター上でアニメーションテーブルを開きます。表の応答セクションを調べて、応答コードが 3 になっており正しいバイト数が読み取られたことを確認します。また、この例では、スレーブから読み込まれたワード (%MW7 で始まる) は、マスタのワード境界に正しく整列されています。

例 2:Modbus アプリケーションの書き込み

マスタープログラム :

ラング	命令	コメント
0	LD 1 [%MW0:=16#010C] [%MW1:=16#0007] [%MW2:=16#0210] [%MW3:=16#0010] [%MW4:=16#0002] [%MW5:=16#0004] [%MW6:=16#6566] [%MW7:=16#6768]	[%MW0:=16#010C]: 送信テーブル長 : 16 進数 0C =10 進数 12、%MW2 から %MW7 [%MW1:=16#0007] [%MW2:=16#0210] : スレーブアドレス 2、10h ファンクションコード書き込みワード [%MW3:=16#0010]: スレーブのアドレス 16 から [%MW4:=16#0002]: 2 ワードの書き込み [%MW5:=16#0004]: 書き込むバイト数 [%MW6:=16#6566]: 第 1 ワードの値 [%MW7:=16#6768]: 第 2 ワードの値
1	LD 1 AND %MSG2.D [EXCH2 %MW0:12]	–
2	LD %MSG2.E ST %Q0.0 END	–

スレーブプログラム :

ラング	命令	コメント
0	LD 1 [%MW18:=16#FFFF] END	–

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

SoMachine Basic を使用して、マスターとスレーブの両方のアプリケーションプログラムを作成します。スレーブの場合は シングルワードメモリー %MW18 を書き込みます。これにより %MW0 から %MW18 までのメモリーアドレスに対してスレーブ上のスペースを割り当てます。スペースを割り当てないと Modbus リクエストはスレーブ上に存在しない場所に書き込もうとします。

マスターでは、EXCH2 命令のワードテーブルは、アドレス MW16 (10 進数 10) の Modbus アドレス 2 のスレーブに 4 バイトを読み込むように初期化されます。

注記：Modbus マスター %MW1 に設定された送信オフセットの使用に注意してください。オフセットの 7 は、6 番目のワードの上位バイトを取り除きます (%MW5 の値 00 (16 進数))。これによりワードテーブルの送信テーブル内のデータ値を、ワード境界上に正しく収まるように整列できます。

EXCH2 命令を実行する前に、アプリケーションは %MSG2 に関連する通信ビットをチェックします。最後に %MSG2 のエラーステータスはローカルベースコントローラー I/O の 1 番目の出力ビットにより検出および格納されます。より正確にチェックするために %SW64 を使用したエラーチェックを追加することもできます。

マスター上のアニメーションテーブルの初期化

アドレス	値	形式
%MW0	010C	16 進数
%MW1	0007	16 進数
%MW2	0210	16 進数
%MW3	0010	16 進数
%MW4	0002	16 進数
%MW5	0004	16 進数
%MW6	6566	16 進数
%MW7	6768	16 進数
%MW8	0210	16 進数
%MW9	0010	16 進数
%MW10	0004	16 進数

スレーブ上のアニメーションテーブルの初期化

アドレス	値	形式
%MW16	6566	16 進数
%MW17	6768	16 進数

各ロジックコントローラーをダウンロードして実行するように設定したら、スレーブ上でアニメーションテーブルを開きます。%MW16 と %MW17 の 2 つの値がスレーブに書き込まれます。

マスターではアニメーションテーブルを使用して通信データの受信テーブル部分を調べることができます。上記の例ではこのデータは、スレーブアドレス、応答コード、書き込まれた最初のワード、および %MW8 から書き込まれたワードの数を表示します。

6.5
LIFO/FIFO レジスター (%R)

LIFO/FIFO レジスターファンクションブロックの使用

このセクションではLIFO/FIFO レジスターファンクションブロックのプログラミングガイドラインと説明をします。

このセクションについて

このセクションには次の項目が含まれています。

説明

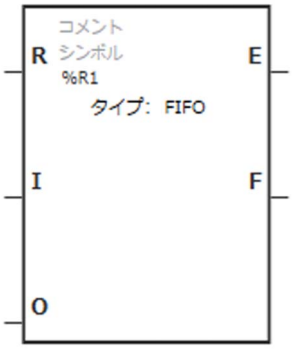
概要

LIFO/FIFO レジスターファンクションブロック は 2 種類の方法で、それぞれ 16 ビットの 16 ワードを格納することができるメモリーブロックです。

- FIFO と呼ばれるキュー (先入れ先出し)。
- LIFO と呼ばれるスタック (後入れ先出し)。

図

次の図は LIFO/FIFO レジスターファンクションブロックを示しています。



入力

LIFO/FIFO レジスターファンクションブロックには次の入力があります。

ラベル	説明	値
R	リセット入力 (または命令)	状態 1 で、LIFO/FIFO レジスターを初期化。
I	格納入力 (または命令)	立上がりで LIFO/FIFO レジスター内の関連ワード %Ri.I の内容を格納します。
O	取得入力 (または命令)	立上がりでは LIFO/FIFO レジスターのデータワードを関連ワード %Ri.O に読み込みます。

出力

LIFO/FIFO レジスターファンクションブロックには次の出力があります。

ラベル	説明	値
E	空の出力 (%Ri.E)	関連するビット %Ri.E は、LIFO/FIFO レジスターが空であることを示します。%Ri.E の値はアニメーションテーブルまたは命令などでテストできます。
F	フル出力 (%Ri.F)	関連するビット %Ri.F は、LIFO/FIFO レジスターがフルであることを示します。%Ri.F の値はアニメーションテーブルまたは命令などでテストできます。

設定

パラメーター

パラメーターを設定するには、ファンクションブロックの設定手順 (*123* ページ参照) に従い、SoMachine Basic オペレーティングガイドのメモリー割り当てモードを読んでください。

LIFO/FIFO レジスターファンクションブロックには次のパラメーターがあります。

パラメーター	説明	値	オンラインモードでの編集
使用	使用済みアドレス	選択されている場合、このアドレスは現在プログラムで使用されています。	不可
アドレス	LIFO/FIFO レジスターオブジェクトアドレス	プログラムで使用できる LIFO/FIFO レジスターオブジェクト数には制限があります。レジスターの最大数については、ハードウェアプラットフォームのプログラミングガイドを参照してください。	不可
シンボル	シンボル	オブジェクトに対応したシンボル。詳細については、SoMachine Basic オペレーティングガイド シンボルの定義と使用 *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を参照してください。	不可
タイプ	LIFO/FIFO レジスタータイプ	**FIFO** (キュー) または **LIFO** (スタック)。	可
コメント	コメント	オブジェクトに対応するコメント。	不可

オブジェクト

LIFO/FIFO レジスターファンクションブロックには次のオブジェクトがあります。

オブジェクト	説明	値
%Ri.I	LIFO/FIFO レジスター入力ワード	読み込み、テスト、書き込みができます。 アニメーションテーブルで編集できます。
%Ri.O	LIFO/FIFO レジスター出力ワード	読み込み、テスト、書き込みができます。 アニメーションテーブルで編集できます。
%Ri.E	空の出力	上の出力表を参照してください。
%Ri.F	フル出力	上の出力表を参照してください。

特殊な場合

この表は、LIFO/FIFO レジスターファンクションブロックをプログラミングする際の特殊な場合のリストを示しています。

特殊な場合	説明
コールドリスタート (%S0=1) または **INIT**	LIFO/FIFO レジスターの内容を初期化します。出力 E に関連する出力ビット %Ri.E は 1 にセットされます。
ウォームリスタート (%S1=1) またはコントローラー停止	LIFO/FIFO レジスターの現在値や出力ビットの状態には影響しません。
%Ri.O および %Ri.I の立上がりの検出。	同じ LIFO/FIFO レジスターのファンクションブロックの呼び出しで %Ri.O および %Ri.I の両方の立上がりが検出された場合、その値は格納も取得もされません。プログラムで値の格納または取得を管理してください。

注記：**INIT** は %S0=1 と同じです。

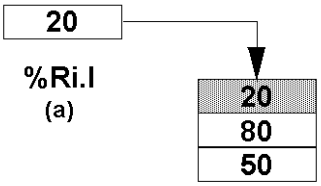
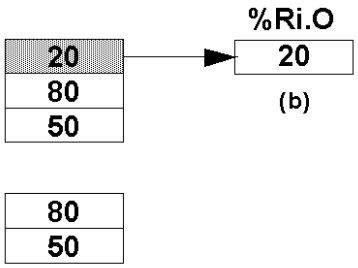
LIFO レジスター動作

概要

LIFO 動作 (後入れ先出し) では、最後に入力されたデータ項目が最初に取得されます。

操作

この表では LIFO の動作について説明します。

ステージ	説明	例
1	**格納** : 格納リクエストを受けると (入力 I の立上がりまたは命令 I が有効となるとき)、入力ワード %Ri.I の内容はスタックの最上位に格納されます (図 a)。スタックがフルの場合 (出力 F=1)、それ以上の格納はできません。	Storage of the contents of %Ri.I at the top of the stack. 
2	**取得** : 取得リクエストを受けると (入力 0 の立上がりまたは命令 0 が有効となるとき)、最上位のデータワード (最後に入力されたワード) がワード %Ri.O (図 b) に読み込まれます。LIFO/FIFO レジスターが空の場合 (出力 E = 1) それ以上の取得はできません。出力ワード %Ri.O は変更されず、その値を保持します。	Retrieval of the data word highest in the stack. 
3	**リセット** : スタックはいつでもリセットできます (入力 R が状態 1 または命令 R が有効)。リセット後のスタックは空です (%Ri.E = 1)。	–

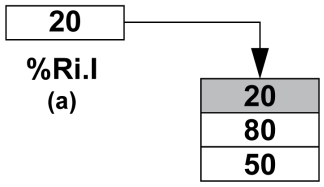
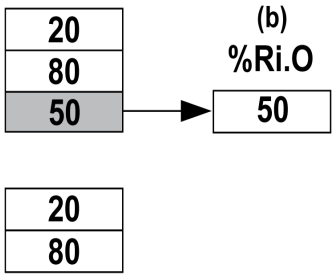
FIFO レジスター動作

概要

FIFO 動作 (先入れ先出し) では、最初に入力されたデータ項目が最初に取得されます。

操作

この表では FIFO の動作について説明します。

ステージ	説明	例
1	**格納** : 格納リクエストを受けると (入力 I の立上がりまたは命令 I が有効となるとき)、入力ワード %Ri.I の内容はキューの最上位に格納されます (図 a)。キューがフルの場合 (出力 F=1)、それ以上の格納はできません。	%Ri.I の内容がキューの最上位に格納されます。 
2	**取得** : 格納リクエストを受けると (入力 O の立上がりまたは命令 O が有効となるとき)、キューの最下位のデータワードは出力ワード %Ri.O に読み込まれ、LIFO/FIFO レジスターの内容はキュー内の 1 つ下に移動します (図 b)。 LIFO/FIFO レジスターが空の場合 (出力 E = 1) それ以上の取得はできません。出力ワード %Ri.O は変更されず、その値を保持します。	最初のデータ項目を取得し、%Ri.O に読み込みます。 
3	**リセット** : キューはいつでもリセットできます (入力 R が状態 1 または命令 R が有効)。リセット後のキューは空です (%Ri.E = 1)。	–

プログラミング例

概要

以下のプログラミング例は、LIFO/FIFO レジスター %R2 が埋まっていない (%R2.F = 0) 場合、格納リクエスト (%I0.2) の状態で ワードメモリー (%MW34) の内容が LIFO/FIFO レジスター (%R2.I) に読み込まれることを示しています。LIFO/FIFO レジスターの格納要求は %M1 によって行われます。取得要求は入力 %I0.3 で確認され、%R2.0 はレジスターが空でない場合 (%R2.E = 0)、%MW20 に読み込まれます。

プログラミング

この例は、LIFO/FIFO レジスターファンクションブロックで可逆が可能な命令です。

ラング	可逆命令
0	BLK %R2 LD %M1 I LD %I0.3 ANDN %R2.E O END_BLK
1	LD %I0.3 [%MW20:=%R2.O]
2	LD %I0.2 ANDN %R2.F [%R2.I:=%MW34] ST %M1

この例は、同じ LIFO/FIFO レジスターファンクションブロックで可逆が不可能な命令です。

ラング	不可逆命令
0	LD %M1 I %R2
1	LD %I0.3 ANDN %R2.E O %R2
2	LD %I0.3 [%MW20:=%R2.O]
3	LD %I0.2 ANDN %R2.F [%R2.I:=%MW34] ST %M1

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

6.6
ドラム (%DR)

ドラムファンクションブロックの使用

このセクションではドラムファンクションブロックのプログラミングガイドラインと説明をします。

このセクションについて

このセクションには次の項目が含まれています。

説明

概要

ドラムファンクションブロック  は外部イベントに応じてステップが変わる電気機械式ドラムシーケンサーに類似した原理で動作します。各ステップにおいて、カムの高点でコントローラーにより実行されるコマンドが与えられます。ドラムファンクションブロックの場合、これらの高点は、各ステップの状態 1 によってシンボル化され、出力ビット %Qi.j またはビットメモリー %Mi に割り当てられます。

図

次の図はオフラインモードのドラムファンクションブロックです。

ステップ ドラムアシスタントで設定されたステップの総数を表示します。
手順 ブロックが作成されたときにオフラインモードで表示されます。オンラインモードでは現在のステップ番号が表示されます。

入力

ドラムファンクションブロックには次の入力があります。

ラベル	説明	値
R	ステップ 0 (または命令) に戻る	状態 1 では**ドラム**をステップ 0 にセットします。
U	前進入力 (または命令)	立上がりで**ドラム** 1 ステップ進めて、制御ビットを更新します。

出力

ドラムファンクションブロックには次の出力があります。

ラベル	説明	値
F	出力 (%DRi.F)	現在のステップが最後に定義されたステップと等しいことを示します。関連するビット %DRi.F はテスト可能です。

設定

パラメーター

パラメーターを設定するには、ファンクションブロックの設定手順 (*123* ページ参照) に従い、SoMachine Basic オペレーティングガイドのメモリー割り当てモードを読んでください。

ドラムファンクションブロックには次のパラメーターがあります。

パラメーター	説明	値	オンラインモードでの編集
使用	使用済みアドレス	選択されている場合、このアドレスは現在プログラムで使用されています。	不可
アドレス	ドラムオブジェクトアドレス	プログラムで使用できるドラムオブジェクト数には制限があります。ドラムオブジェクトの最大数については、コントローラーの *プログラミングガイド* を参照してください。	不可
シンボル	シンボル	オブジェクトに対応したシンボル。詳細については、SoMachine Basic オペレーティングガイド シンボルの定義と使用 *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を参照してください。	不可
設定	ドラムアシスタント	クリックしてドラムアシスタント (*164* ページ参照) を表示します。	可 (ドラムアシスタントウィンドウのすべてのパラメーター)
コメント	コメント	オブジェクトに対応するコメント。	不可

オブジェクト

ドラムファンクションブロックには次のオブジェクトがあります。

オブジェクト	説明	値
%DRi.S	現在のステップ番号	0<=%DRi.S<=7 読み込みおよび書き込みが可能なワード。書き込まれる値は 10 進値の即値でなければなりません。書き込み後、次のファンクションブロックの実行時に効果が発生します。 アニメーションテーブルまたはオンラインモードで編集できます。
%DRi.F	フル	出力表 (*163* ページ参照) を参照してください。

動作

ドラムファンクションブロックは以下のもので構成されています。

- 0 ～ 15 の番号のついた列に配置された 8 つのステップ (0 ～ 7) と 16 ビット (ステップの状態) で構成された定数データ (カム) の行列。
- 制御ビットのリストは、設定された出力 (%Qi.j)、またはワードメモリー (%Mi) に関連付けられています。現在のステップでは、制御ビットはこのステップで定義されたバイナリ状態をとります。

ドラムアシスタント

ドラムアシスタントを使用してドラムファンクションブロックを設定します。

ステップ数の設定 1...8 および各ステップに関連する出力またはビットメモリー : **ビット 0 ... ビット 15**、次に **OK** をクリックします

注記：設定はビットメモリー (%Mi) を使用して設定することもできます。

特殊な場合

この表は、ドラムの動作の特殊な場合のリストを示しています。

特殊な場合	説明
コールドリスタート (%S0=1)	ドラムをステップ 0 にリセットします (制御ビットの更新)。
ウォームリスタート (%S1=1)	現在のステップの後に制御ビットを更新します。
プログラムジャンプ	ドラムはスキャンされなくなり、制御ビットは最後の状態を保持します。
制御ビットの更新	ステップが変更された場合またはリスタート (ウォームまたはコールド) の場合にのみ発生します。

プログラミング例

概要

以下はステップ 0 でコントロールを設定せず、ステップ 1 からステップ 6 でコントロールを出力 %Q0.0 から %Q0.5 にそれぞれ設定するようにドラムをプログラミングする例です (詳細は設定 (*168* ページ参照) を参照してください)。

最初の 6 点の出力 %Q0.0 ～ %Q0.5 は、入力 %I0.1 が 1 に設定されるたびに順次有効となります。次に示す値が 1 の場合、入力 %I0.0 はそれらを 0 にリセットします。

- ドラム出力 **F** (%DRi.F = 0)
- 現在のステップ番号 (%DRi.S = 0)

プログラミング

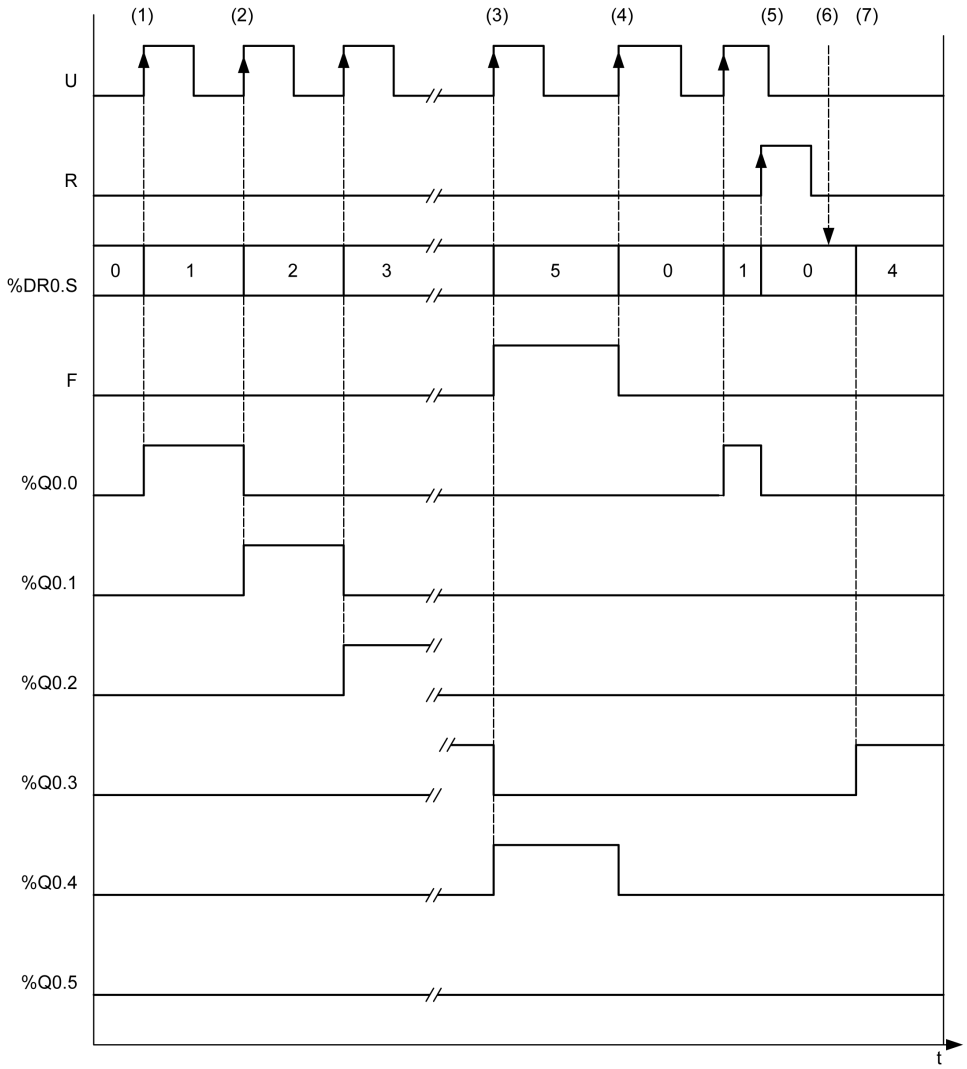
次の例はドラムファンクションブロックプログラムです。

ラング	命令
0	BLK %DR1 LD %I0.0 R LD %I0.1 U OUT_BLK LD F ST %Q0.7 END_BLK

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

タイミングチャート

この図はドラムの動作を示しています。



(1) 入力 U の立上がりで現在のステップはインクリメントされます。
(2) 現在のステップが更新されると出力も更新されます
(3) 最後のステップに達すると、出力 F は 1 に設定されます
(4) 最後のステップが有効なときに入力 U の立上がりがあると、現在のステップは 0 にリセットされます
(5) %DR0.R = 1 (立上がり) で現在値は 0 に設定されます
(6) ユーザーがステップ番号の値を書き込みます : %DR0.S = 4
(7) ユーザーによって書き込まれた値は、次の実行時に更新されます

設定

設定中に以下の情報が定義されます。

- ステップの数 :6
- 各ドラムステップの出力状態 (制御ビット)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ステップ0 :	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ステップ1 :	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ステップ2 :	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
ステップ3 :	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
ステップ4 :	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
ステップ5 :	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

- 制御ビットの割り当て

関連する制御ビットの出力を以下の表に示します。

Bit	関連出力
0	関連出力無し
1	%Q0.1
2	%Q0.2
3	%Q0.3
4	%Q0.4
5	%Q0.5

6.7
ビットレジスターをシフト (%SBR)

ビットレジスターをシフトファンクションブロックの使用

このセクションでは、ビットレジスターをシフトファンクションブロックのプログラミングガイドラインと説明をします。

このセクションについて

このセクションには次の項目が含まれています。

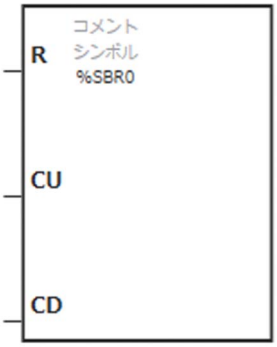
説明

概要

ビットレジスターをシフトファンクションブロック  のバイナリーデータビット (0、または 1) を右、または左にシフトします。

図

次の図はビットレジスターをシフトファンクションブロックを示しています。



ビットレジスターをシフトの現在値はファンクションブロックの中央に表示されます。

- 10 進数 (例 : 7)
- 2 進数 (例 : 111)
- 16 進数値 (例 : 16 # 7)

入力

ビットレジスターをシフトファンクションブロックには次の入力があります。

ラベル	説明	値
R	リセット入力 (または命令)	ファンクションパラメーター R が 1 の場合、レジスタービットは 0 は 15 になり、%SBRi. j は 0 になります。
CU	左にシフト入力 (または命令)	立上がりでレジスタービットを左にシフトします。
CD	右にシフト入力 (または命令)	立上がりでレジスタービットを右にシフトします。

設定

パラメーター

パラメーターを設定するには、ファンクションブロックの設定手順 (*123* ページ参照) に従い、SoMachine Basic オペレーティングガイドのメモリー割り当てモードを読んでください。

ビットレジスターをシフトファンクションブロックには次のパラメーターがあります。

パラメーター	説明	値
使用	使用済みアドレス	選択されている場合、このアドレスは現在プログラムで使用されています。
アドレス	**ビットレジスターをシフト** オブジェクトアドレス	プログラムで使用できるビットレジスターをシフトオブジェクト数には制限があります。レジスターの最大数については、ハードウェアプラットフォームのプログラミングガイドを参照してください。
シンボル	シンボル	オブジェクトに対応したシンボル。詳細については、SoMachine Basic オペレーティングガイド シンボルの定義と使用 *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を参照してください。
コメント	コメント	オブジェクトに対応するコメント。

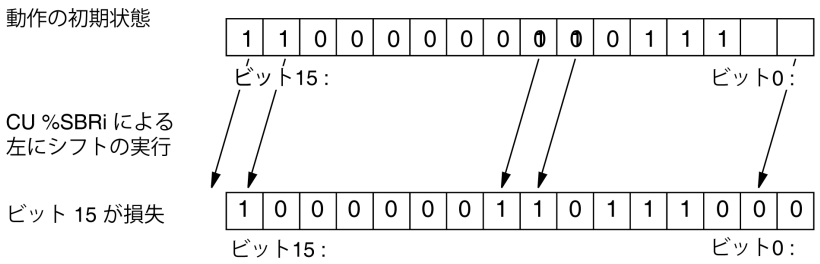
オブジェクト

ビットレジスターをシフトファンクションブロックには次のオブジェクトがあります。

オブジェクト	説明	値
%SBRi	レジスタ番号	0 ～ 7 アニメーションテーブルで編集できます。
%SBRi.j	レジスタビット	シフトレジスタのビット 0 ～ 15 (j = 0 ～ 15) はテスト命令でテストすることができ、また代入命令で書き込むことができます。

演算

シフトの前後のビットパターンを次の図に示しています。



これは CD 命令を使用してビットを右にシフトするリクエスト (ビット 15 ～ ビット 0) にも当てはまります。ビット 0 は失われます。

16 ビットレジスタが適切でない場合はプログラムを使用して複数のレジスタをカスケード接続することができます。

特殊な場合

この表は、ビットレジスターをシフトファンクションブロックをプログラミングする際の特殊な場合のリストを示しています。

特殊な場合	説明
コールドリスタート (%S0=1)	レジスタワードのすべてのビットを 0 に設定します。
ウォームリスタート (%S1=1)	レジスタワードのビットには影響しません。

プログラミング例

概要

ビットレジスターをシフトファンクションブロックはバイナリーデータビット (0、または 1) を右、または左にシフトします。

プログラミング

この例では、ビットは 1 秒おきに左にシフトされ、ビット 0 の状態はビット 15 を引き継ぎます。

可逆命令においては

ラング	可逆命令
0	BLK %SBR0 LD %S6 CU END_BLK
1	LD %SBR0.15 ST %SBR0.0

不可逆命令においては

ラング	不可逆命令
0	LD %S6 CU %SBR0
1	LD %SBR0.15 ST %SBR0.0

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

6.8
ステップカウンター (%SC)

ステップカウンターファンクションブロックの使用

このセクションではステップカウンターファンクションブロックのプログラミングガイドラインと説明をします。

このセクションについて

このセクションには次の項目が含まれています。

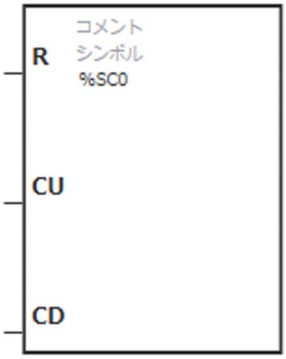
説明

概要

ステップカウンターファンクションブロック  は、処理を割り当てられる一連のステップを提供します。ステップから別のステップへの移動は、外部イベントまたは内部イベントに依存します。ステップがアクティブになるたびに関連ビット (ステップカウンタービット %SCi.j) が 1 にセットされます。一度にアクティブにできるのはステップカウンターの 1 ステップのみです。

図

次の図は、ステップカウンターファンクションブロックを示しています。



入力

ステップカウンターファンクションブロックには次の入力があります。

ラベル	説明	値
R	リセット入力 (または命令)	ファンクションパラメーター R が 1 のときステップカウンターはリセットされます。
CU	インクリメント入力 (または命令)	立上がりでステップカウンターは 1 ずつインクリメントされます。
CD	デクリメント入力 (または命令)	立上がりでステップカウンターは 1 ずつデクリメントされます。

設定

パラメーター

パラメーターを設定するには、ファンクションブロックの設定手順 (*123* ページ参照) に従い、SoMachine Basic オペレーティングガイドのメモリー割り当てモードを読んでください。

ステップカウンターファンクションブロックには次のパラメーターがあります。

パラメーター	説明	値
使用	使用済みアドレス	選択されている場合、このアドレスは現在プログラムで使用されています。
アドレス	ステップカウンターオブジェクトアドレス	プログラムで使用できる ステップカウンター オブジェクト数には制限があります。ステップカウンターの最大数については、ハードウェアプラットフォームのプログラミングガイドを参照してください。
シンボル	シンボル	オブジェクトに対応したシンボル。詳細については、SoMachine Basic オペレーティングガイド シンボルの定義と使用を参照してください。
コメント	コメント	オブジェクトに対応するコメント。

オブジェクト

ステップカウンターファンクションブロックには次のオブジェクトがあります。

オブジェクト	説明	値
%SCi.j	ステップカウンタービット	ステップカウンタービット 0 ～ 255 (j = 0 ～ 255) は、ロード論理演算によってテストすることができ、代入命令で書き込むことができます。 アニメーションテーブルで編集できます。

特殊な場合

この表は、ステップカウンターファンクションブロック使用上の特殊な場合のリストを示しています。

特殊な場合	説明
コールドリスタート (%S0=1)	ステップカウンターを初期化します。
ウォームリスタート (%S1=1)	ステップカウンターに影響はありません。

プログラミング例

概要

次の例は、ステップカウンターファンクションブロックです。

- ステップカウンター 0 は入力 %I0.1 によってデクリメントされます。
- ステップカウンター 0 は入力 %I0.2 によってインクリメントされます。
- ステップカウンター 0 は入力 %I0.3 またはステップ 3 到達時に 0 にリセットされます。
- ステップ 0 は出力 %Q0.1 を制御し、ステップ 1 は出力 %Q0.2 を制御し、ステップ 2 は出力 %Q0.3 を制御します。

プログラミング

この例は、ステップカウンターファンクションブロックで可逆が可能な命令です。

ラング	可逆命令
0	BLK %SC0 LD %SC0.3 OR %I0.3 R LD %I0.2 CU LD %I0.1 CD END_BLK
1	LD %SC0.0 ST %Q0.1
2	LD %SC0.1 ST %Q0.2
3	LD %SC0.2 ST %Q0.3

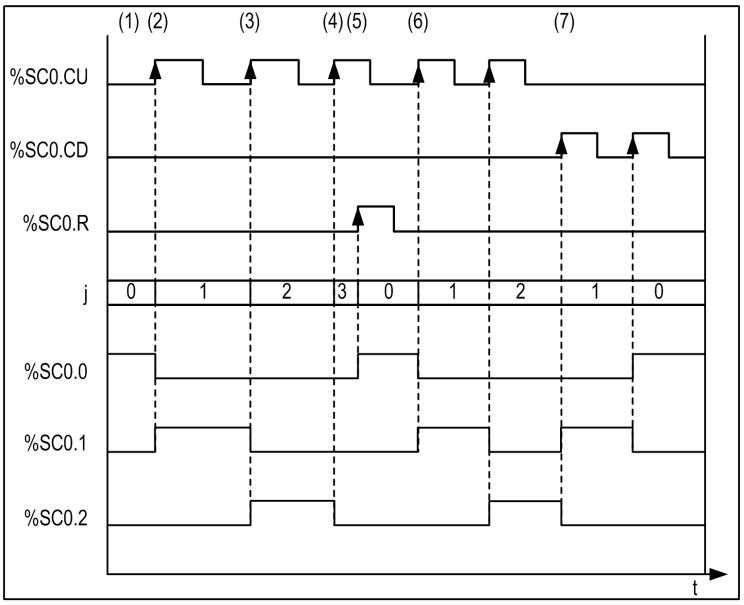
この例は、ステップカウンターファンクションブロックで可逆が不可能な命令です。

ラング	不可逆命令
0	LD %SC0.3 OR %I0.3 R %SC0
1	LD %I0.2 CU %SC0
2	LD %I0.1 CD %SC0
3	LD %SC0.0 ST %Q0.1
4	LD %SC0.1 ST %Q0.2
5	LD %SC0.2 ST %Q0.3

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

タイミングチャート

この図は、ステップカウンターファンクションブロックの動作を示しています。



(1) ステップ 0 はアクティブなので %SC0.0 は 1 に設定されます
(2) 立上がりで CU 入力があれば、ステップをインクリメントして出力を更新します
(3) ステップがインクリメントされ、出力が更新されます。
(4) ステップ 3 がアクティブなので、**Reset** 入力は CPU サイクル 1 回後にアクティブとなります
(5) **Reset** がアクティブであれば、現在のステップは 0 に設定され、CPU サイクル 1 回後にリセット入力が 0 に設定されます
(6) 現在のステップは CU 入力立上がりにおいてインクリメントされます
(7) 立上がりで CD 入力があれば、ステップをデクリメントして出力を更新します

6.9
スケジュールブロック (%SCH)

スケジュールブロックの使用

このセクションではスケジュールブロックのプログラミングガイドラインと説明をします。

このセクションについて

このセクションには次の項目が含まれています。

説明

概要

スケジュールブロックは定義済みの月、日、時間での動作を管理します。

スケジュールブロック は SoMachine Basic でのみ設定可能です。他のファンクションブロックと同様にプログラムラングに挿入することはできません。

注記: システムビット %S51 およびシステムワード %SW118 をチェックして、Real-Time Clock (RTC) オプションがインストールされていることを確認します。スケジュールブロックを使用するには RTC オプションが必要です。

スケジュールブロックは次のいずれかで設定できます。
- **スケジュールブロックプロパティ**ウィンドウ (**ツールタブ → スケジュールブロック**)。
- 専用オブジェクト %SCHi.xxx (*179* ページ参照) の使用 。

スケジュールブロックプロパティの設定

スケジュールブロックを設定するには、メモリー割り当てモード *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* の説明を参照してください。

スケジュールブロックには次のパラメータがあります。

パラメーター	説明	値
使用	使用済みアドレス	選択されている場合、このアドレスは現在プログラムで使用されています。
アドレス	オブジェクトアドレス	プログラムで使用できるスケジュールブロックオブジェクト数には制限があります。スケジュールブロックの最大数については、ハードウェアプラットフォームのプログラミングガイドを参照してください。
設定済み	使用するために構成されているかどうか。	チェックボックスが選択されている場合、使用設定されています。選択されていない場合は、使用されていません。
出力ビット	出力ビット	出力の割り当てはスケジュールブロック %Mi または %Qj.k により有効化されます。 現在の日付と時刻がアクティブ期間 (開始と終了の設定の間) にある場合、この出力は 1 に設定されます。
開始日	スケジュールブロックを開始する日付。	**1...31**
開始月	スケジュールブロックを開始する月。	スケジュールブロック
終了日	スケジュールブロックを終了する日付。	**1...31**
終了月	スケジュールブロックを終了する月。	**1 月 ～ 12 月**
開始時間	スケジュールブロックを開始する時刻、時間、分。	時 :**0...23** 分 :**0...59**
終了時間	スケジュールブロックを終了する時刻、時間、分。	時 :**0...23** 分 :**0...59**
月曜日	スケジュールブロックを有効にする曜日を指定するチェックボックス。	チェックボックスが選択されている場合、使用設定されています。選択されていない場合は、使用されていません。
火曜日		
水曜日		
木曜日		
金曜日		
土曜日		
日曜日		
コメント	コメント	オブジェクトに対応するコメント。

オブジェクト

これらのオブジェクトはブロックが上記のように設定されている場合にのみ使用できます。

アプリケーションのファンクションレベル *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* は**レベル 6.0** 以上に設定してください。

スケジュールブロックのオブジェクトを以下に示します。

オブジェクト	説明
%SCHi.STARTDAY	スケジュールブロックを開始する日付。
%SCHi.STARTMONTH	スケジュールブロックを開始する月。
%SCHi.STARTHOUR	スケジュールブロックを開始する時間。
%SCHi.STARTMIN	スケジュールブロックを開始する分。
%SCHi.ENDDAY	スケジュールブロックを終了する日付。
%SCHi.ENDMONTH	スケジュールブロックを終了する月。
%SCHi.ENDHOUR	スケジュールブロックを終了する時間。
%SCHi.ENDMIN	スケジュールブロックを終了する分。
%SCHi.DOW	スケジュールブロックを有効にする曜日。

プログラム内でこれらのオブジェクト値を変更することで、スケジュールブロックの設定を動的に変更できます。この変更は次回の MAST スキャンで考慮されます。

コールドリスタート (%S0 = 1) 後、オブジェクト値は設定中に定義された値にリセットされます。

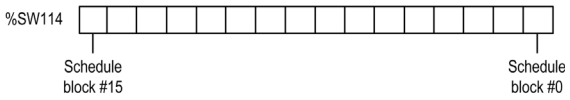
システムワードまたはワードメモリーを使用したスケジュールブロック設定

システムワードまたはワードメモリーを使用して**スケジュールブロック**設定することもできます。

スケジュールブロックの有効化

システムワード %SW114 のビットは、16 ビットの各スケジュールブロックを有効 (ビットを 1 に設定) または無効 (ビットを 0 に設定) にします。

%SW114 のスケジュールブロックの割り当て



初期設定では (またはコールドリスタート後)、このシステムワードのすべてのビットが 1 (有効) に設定されます。プログラム内でのこれらのビットの使用はオプションです。

スケジュールブロックの出力

同じ出力 (%Mi または %Qj.k) が複数のスケジュールブロックによって割り当てられている場合は、このオブジェクトに最終的に割り当てられた各ブロックの結果の論理和 (OR) です (同じ出力に対し複数のスケジュールブロックを持つことができます)。

例えば、スケジュールブロック %SCH0 および %SCH1 の両方が出力 %Q0.0 に割り当てられます。%SCH0 は、月曜日の 12 時から 13 時までの出力を設定し、%SCH1 は、火曜日の 12 時から 13 時まで出力を設定します。結果、出力は月曜日と火曜日の両方で 12 時から 13 時まで設定されます。

プログラミングと設定

概要

スケジュールブロックは定義済みの月、日、時間での動作を管理します。

プログラミング例

夏季の散布プログラムの例のパラメーターを次の表に示します。

パラメーター	値	説明	オンラインモードでの編集
アドレス	**リアルタイムクロック 6**	スケジュールブロック番号 6	不可
設定済み	ボックスをチェックする (有効)	スケジュールブロック番号 6 を設定するため、ボックスをチェックしました。	不可
出力ビット	**%Q0.2**	出力 %Q0.2 を有効にします	可
開始日	**21**	6 月 21 日にアクティビティを開始します	可
開始月	**6 月**	6 月にアクティビティを開始します	可
開始時間	**21**	21:00 に活動開始	可
終了日	**21**	9 月 21 日に活動を停止します	可
終了月	**9 月**	9 月にアクティビティを停止します	可
終了時間	**22**	22:00 にアクティビティを停止します	可
月曜日	ボックスをチェックする (有効)	月曜日にアクティビティを実行します	可
火曜日	ボックスをチェックしない (無効)	アクティビティーなし	可
水曜日	ボックスをチェックする (有効)	水曜日にアクティビティを実行します	可
木曜日	ボックスをチェックしない (無効)	アクティビティーなし	可
金曜日	ボックスをチェックする (有効)	金曜日にアクティビティを実行します	可
土曜日	ボックスをチェックしない (無効)	アクティビティーなし	可
日曜日	ボックスをチェックしない (無効)	アクティビティーなし	可

このプログラムを使用すると、スケジュールブロックを入力 %I0.1 に接続されたスイッチまたは湿度計によって無効にすることができます。

ラング	命令	コメント
0	LD %I0.1 ST %SW114:X6	この例では %SCH6 は有効です。

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

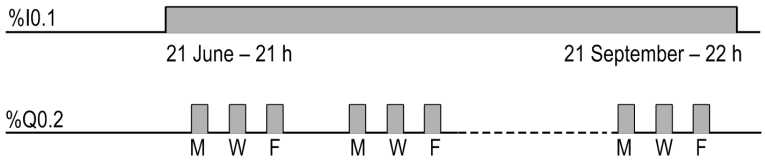
次の表に、プログラムの例を示します。

ラング	命令
0	LD %M4 [%MO:4 := 7]
1	LD %MO RISING1 OPER [%SCHO.STARTDAY := %MWO] OPER [%SCHO.STARTMONTH := %MW1] OPER [%SCHO.STARTHOUR := %MW2] OPER [%SCHO.STARTMIN := %MW3] R %MO

ラング	命令
2	LD %M1 RISING0 OPER [%SCH0.ENDDAY := %MW4 OPER [%SCH0.ENDMONTH := %MW5] OPER [%SCH0.ENDHOUR := %MW6] OPER [%SCH0.ENDMIN := %MW7] R %M1
3	LD %M2 RISING2 [%SCH0.DOW := %MW8]

タイミングチャート

出力 %Q0.2 が有効になるタイミングを次の図に示します。



6.10
リアルタイムクロック (%RTC)

RTC ファンクションブロックの使用

このセクションでは RTC ファンクションブロックのプログラミングガイドラインと説明をします。

注記：RTC ファンクションブロックを使用するには、アプリケーションの ファンクションレベル *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を**レベル 5.0** 以上に設定してください。

このセクションについて

このセクションには次の項目が含まれています。

説明

概要

RTC ファンクションブロック  では、M221 ロジックコントローラーのリアルタイムクロック (RTC) の読み書きが可能です。

図

次の図は RTC ファンクションブロックを示しています。

入力

RTC ファンクションブロックには次の入力があります。

ラベル	オブジェクト	値
Enable	-	この入力の立上がりが検出されるとファンクションブロックを有効にします。 状態 1 では **RD** および **WR** 入力値は連続的に読み取られ、実行するアクションが決まります。 状態 0 ではファンクションブロックは無効化され出力はリセットされます。
RD	-	2 つの入力の値の組み合わせによって、実行するアクションが決まります。 ● RD = 0 および WR = 0。アクションはありません。 出力ビット Done および Error は 0 に設定されます。 ● RD = 1 および WR = 0。RTC 値を読み取ります。 成功した場合 Done 出力は 1 に設定され、Error 出力は 0 に設定されます。 入力オブジェクトは RTC から読み取られた値で継続的に更新されます。オブジェクト値を表示するには、アニメーションテーブル *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を使用してください。 ● RD = 0 および WR = 1。WR の立上がりが検出されると、このファンクションブロックに関連する RTC パラメーターで指定されたオブジェクト値を使用して RTC を更新します (下記参照)。 更新に成功した場合 Done 出力は 1 に設定され、Error 出力は 0 に設定されます。RTC が更新されます。 失敗した場合 Done 出力は 0 に設定され、Error 出力は 1 に設定されます。 ● RD = 1 および WR = 1。対応していません。 Done 出力を 0 に設定すると、Error 出力は 1 に設定され、ErrorId 出力 (*185* ページ参照) は 256 (同時読み書き) に設定されます。
WR	-	

ラベル	オブジェクト	値
Day	%RTCi.DAY	日 初期値 :12 -32767...32768
Month	%RTCi.MONTH	月 初期値 :6 -32767...32768
Year	%RTCi.YEAR	年 初期値は 2017 -32767...32768
Hours	%RTCi.HOURS	時間 初期値 :0 -32767...32768
Minutes	%RTCi.MINUTES	分 初期値 :0 -32767...32768
Seconds	%RTCi.SECONDS	秒 初期値 :0 -32767...32768

出力

RTC ファンクションブロックには次の出力があります。

ラベル	オブジェクト	値
Done	%RTCi.Done	RTC が読み込みおよび書き込みに成功すると、1 に設定されます。 読み取りまたは書き込み操作が失敗した場合は 0 に設定します。
Error	%RTCi.Error	実行中にエラーが検出された場合 1 に設定。ファンクションブロックの実行は完了されます。ErrorId 出力オブジェクトはエラーの原因を示します。
DayOfWeek	%RTCi.DayOfWeek	現在の週の値から計算された曜日を返します。 Range:0...7 0: ファンクションブロック未実行 1...7 曜日
ErrorId	%RTCi.ErrorId	エラーコード識別子。 下記のエラーコードを (*185* ページ参照) を参照してください。

エラーコード

Error 出力が 1 に設定されているときに、次のコードは %RTCi.ErrorId オブジェクトに返すことができます。

エラーコード	説明
0	エラーなし
1	年がエラー
2	月がエラー
3	日がエラー
4	週がエラー
5	時がエラー
6	分がエラー
7	秒がエラー
8	組み合わせエラー
9	RTC 内部エラー
256	同時読み書き
257	RTC 更新が進行中 (システムビット %S50 = 1)

設定

概略

ロジックコントローラー内の RTC の更新に使用する値で、RTC ファンクションブロックのプロパティを設定します。

RTC のプロパティページを表示するには、次のいずれかを行います。

- RTC ファンクションブロックをダブルクリックします。
- **プログラミング**タブで、**ツール → ソフトウェアオブジェクト → RTC** を選択します。

RTC のプロパティ

パラメーターを設定するには、ファンクションブロックの設定手順 (*123* ページ参照) に従い、SoMachine Basic オペレーティングガイドのメモリー割り当てモードを読んでください。

RTC プロパティページには、次のプロパティが表示されます。

プロパティ	説明	値	オンラインモードでの編集
使用	オブジェクトアドレスが使用中です	選択されている場合、このアドレスは現在プログラムで使用されています。	不可
アドレス	RTC オブジェクトアドレス	%RTC i、ここで i はオブジェクト番号です。プログラムで使用できる RTC オブジェクト数には制限があります。RTC オブジェクトの最大数については、コントローラーの *プログラミングガイド* を参照してください。	不可
シンボル	シンボル	オブジェクトに対応したシンボル。詳細については、SoMachine Basic オペレーティングガイド シンボルの定義と使用 *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を参照してください。	可
日	日付の日の部分	1...31	可
月	日付の月の部分	1...12	可
年	年	2000 以上	可
時間	時刻の時の部分	0...23	可
分	時刻の分の部分	0...59	可
秒	時刻の秒の部分	0...59	可
コメント	コメント	オブジェクトに対応するコメント。	可

6.11
PID

PID ファンクション

概要

PID ファンクションは、動的プロセスを連続的に制御するために使用されます。PID 制御の目的は、プロセスを目標値に可能な限り近づけておくことです。

PID 動作、機能、および PID ファンクションの実装の詳細については、アドバンストファンクションライブラリーガイドを参照してください。

- PID 動作モード
- PID オートチューニングの設定
- PID 標準設定
- PID アシスタント
- PID プログラミング
- PID パラメーター
  - PID パラメーターの役割と影響
  - PID パラメーターの調整方法

6.12
データロギング

データロギング

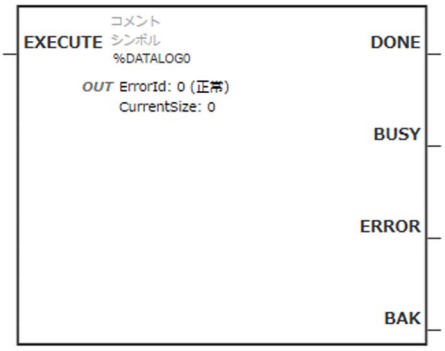
ファンクションブロックの説明

データロギング機能を使用すると、オブジェクトまたは文字列のデータを /user/DATALOGx.csv ファイルに格納することができます。ここで x はデータロギングインスタンス番号に対応する整数です。このファイルは SD カードに保存されています。

SD カード名は DATA にしてください。

注記: アプリケーションは少なくとも**レベル 6.0** のファンクションレベルとブートローダバージョン 50 以降で設定してください。コントローラーのブートバージョンについては、システムワード %SW13 を参照してください。

図



データロギング設定

手順	アクション
1	**プログラミング**ウィンドウ内で次の順にクリックします。**ツール → ソフトウェアオブジェクト → データロギング**
2	**設定**列で、 … をクリックします。 **結果 :DATALOG アシスタント**が表示されます。
3	**最大ファイルサイズ**を入力します。 **最大ファイルサイズ**は 1 ～ 500 です。初期値は 100 です。
4	**ヒストリカルデータ** または **イベントログ** を選択します **ヒストリカルデータ**機能を使用すると、オブジェクトからいくつかのデータを保存できます。SoMachine Basic オブジェクトのみ許可されます。32 個までオブジェクトを保存できます。 **イベントログ**機能を使用すると、1 つの文字列を保存できます。
5	クリアするか、**タイムスタンプを追加**を選択します。 この機能はイベントの時間を CSV ファイルに追加します。
6	**アドレス**列にオブジェクトを入力します。 **イベントログ**を選択した場合は文字列の最初の %MW を入力してください。 オブジェクトにシンボルを割り当てた場合は**シンボル**列にそれが表示されます。ここでシンボルを変更することはできません。
7	**適用**をクリックします。
8	ファンクションブロックをラダーエディタに挿入するには、次のいずれかの方法を実行します。 ●  を順にクリックします。 ● **データロギングのプロパティ**からドラッグアンドドロップします。

入力

ファンクションブロックの入力を次の表に示します。

ラベル	タイプ	説明
EXECUTE	BOOL	立上がりが検出されたときにファンクションブロックの実行を開始します。

出力

ファンクションブロックの出力を次の表に示します。

ラベル	タイプ	説明
DONE	BOOL	立上がりが検出されたときにファンクションブロックの実行を開始します。 ファンクションブロックの実行中に 2 度目の立上がりが検出されても、それは無視されます。また実行中のコマンドは影響を受けません。
BUSY	BOOL	TRUE の場合、ファンクションブロックの実行が進行中であることを示します。
ERROR	BOOL	TRUE の場合、エラーが検出されたことを示します。ファンクションブロックの実行は停止されます。ErrorId 出力はエラーコードを示します。
BAK	BOOL	TRUE の場合 BAK ファイルが作成されます。

エラーコード

エラーコード	説明
0	エラーなし
1	動作中
2	データを CSV ファイルに保存する際にエラーが発生しました。
3	BAK ファイル作成時のエラー。
4	すでに使用されている Datalog インスタンス。
10	タイムアウトを復元します。SD カードを取り外してください。
11	SD カードが検出されませんでした。
12	SD カードの書き込みが禁止です。
50	ファイルシステムエラーが検出されました。
51	CSV ファイル操作時にエラーが発生しました。

6.13
グラフセステップ

グラフセステップ

概要

グラフセステップオブジェクト (%X *i*) は、プログラム中の対応するグラフセステップ *i* の状態を識別します。

パラメーター

グラフセステッププロパティウィンドウには次のプロパティを表示します。

パラメーター	説明	値
使用	使用済みアドレス	選択されている場合、このアドレスはプログラムで使用されています。
アドレス	グラフセステップオブジェクトアドレス	プログラムには最大 96 個のグラフセステップオブジェクトを使用できます。
シンボル	シンボル	オブジェクトに対応したシンボル。詳細については、SoMachine Basic オペレーティングガイド シンボルの定義と使用 *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を参照してください。
コメント	コメント	オブジェクトに対応するコメント。

第 7 章
PTO オブジェクト

この章について

この章には次のセクションが含まれています。

7.1
モーションタスクテーブル (%MT)

モーションタスクテーブル

概要

コントローラーのアドバンスト *ファンクションライブラリーガイド* を参照してください。

7.2
パルストレイン出力 (%PTO)

パルストレイン出力

概要

コントローラーのアドバンスト *ファンクションライブラリーガイド* を参照してください。

第 8 章
ドライブオブジェクト

ドライブオブジェクト

概要

ドライブオブジェクトは、Modbus Serial IOScanner または Modbus TCP IOScanner 上に設定された ATV ドライブおよび他のデバイスを制御します。

アドバンストファンクションライブラリーガイドを参照してください。

第 9 章
通信オブジェクト

概要

通信ファンクションブロックは Modbus デバイスとの通信およびキャラクタモード (ASCII) でのメッセージの送受信に使用されます。

注記：マスタータスクサイクル中は通信ポート上で 1 度に 1 つの通信ファンクションブロックのみをアクティブにすることができます。同じ通信ポートで複数の通信ファンクションブロックまたは EXCH 命令を同時に使用しようとすると、ファンクションブロックはエラーコードを返します。従って通信ファンクションブロックまたは EXCH 命令を開始する前に通信ポート上の通信がアクティブでないこと (%MSGx.D が TRUE) を確認してください。

この章について

この章には次のセクションが含まれています。

9.1
リモートデバイスからのデータの読み取り (%READ_VAR)

%READ_VAR ファンクションブロックの使用

このセクションでは %READ_VAR ファンクションブロックのプログラミングガイドラインと説明をします。

このセクションについて

このセクションには次の項目が含まれています。

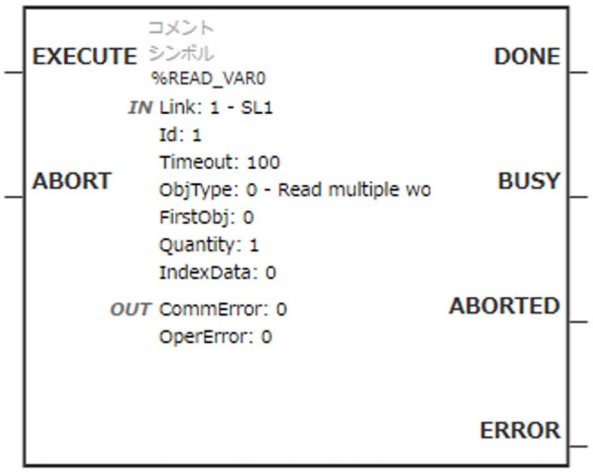
説明

概要

%READ_VAR ファンクションブロックは、Modbus SL または Modbus TCP 上のリモートデバイスからのデータ読み込みに使われます。

図

次の図は %READ_VAR ファンクションブロックを示しています。



入力

%READ_VAR ファンクションブロックには次の入力があります。

ラベル	タイプ	値
Execute	BOOL	立上がりが検出されたときにファンクションブロックの実行を開始します。ファンクションブロックの実行中に 2 度目の立上がりが検出されても、それは無視されます。また実行中のコマンドは影響を受けません。
Abort	BOOL	立上がりが検出されたときにファンクションブロックの実行を停止します。**Aborted** 出力は 1 に設定され、%READ_VARi.CommError オブジェクトはコード 16 進数 02 (ユーザーリクエストによる通信停止) が含まれます。

注記 : RUN の最初のタスクサイクルで **Execute** または **Abort** 入力を TRUE に設定しても、立上がりとして検出されません。ファンクションブロックが次の立上がりを検出するためには、最初に入力を FALSE にしてください。

出力

%READ_VAR ファンクションブロックには次の出力があります。

ラベル	タイプ	値
Done	BOOL	TRUE の場合、ファンクションブロックの実行がエラーなしに完了したことを示します。
Busy	BOOL	TRUE の場合、ファンクションブロックの実行が進行中であることを示します。
Aborted	BOOL	TRUE の場合、ファンクションブロックの実行が **Abort** 入力でキャンセルされたことを示します。
Error	BOOL	TRUE の場合、エラーが検出されたことを示します。ファンクションブロックの実行は停止されます。 CommError および OperError の詳細については、通信エラーコード (*200* ページ参照) および 動作エラーコード (*200* ページ参照) の表を参照してください。

ファンクションブロックの出力オブジェクトを次の表に示します。

出力オブジェクト	タイプ	説明
CommError	BYTE	%READ_VARi.CommError の詳細は、通信エラーコード (*200* ページ参照) を参照してください。
OperError	DWORD	%READ_VARi.OperError の詳細は、動作エラーコード (*200* ページ参照) を参照してください。

通信エラーコード

この表は %READ_VARi.CommError ワードオブジェクトのエラーコードを示します。

名前	検出されたエラーコード	説明
CommunicationOK	0 (16 進数 00)	通信は正常です。
TimedOut	1 (16 進数 01)	タイムアウトにより通信は停止しました。
Abort	2 (16 進数 02)	ユーザーのリクエスト (**Abort** 入力) により Exchange が停止しました 。
BadAddress	3 (16 進数 03)	アドレス形式がが不正です。
BadRemoteAddr	4 (16 進数 04)	リモートアドレスが不正です。
BadMgtTable	5 (16 進数 05)	管理テーブルの書式が不正です。
BadParameters	6 (16 進数 06)	特定のパラメーターが不正です。
ProblemSendingRq	7 (16 進数 07)	宛先への送信リクエストが失敗しました。
RecvBufferNotAlloc	9 (16 進数 09)	受信バッファサイズが不十分です。
SendBufferNotAlloc	10 (16 進数 0A)	送信バッファサイズが不十分です。
SystemResourceMissing	11 (16 進数 0B)	システムリソースがありません。
BadLength	14 (16 進数 0E)	長さが不正です。
ProtocolSpecificError	254 (16 進数 FE)	Modbus プロトコルエラーを示します。詳細は オペレーションエラーコード (*200* ページ参照) を参照してください。
Refused	255 (16 進数 FF)	メッセージは拒否されました。詳細は オペレーションエラーコード (*200* ページ参照) を参照してください。

動作エラーコード

通信エラーコード (CommError オブジェクト) が次の値をもつ場合、このリターンコードは重要です。

- 0 (16 進数 00) (正常)
- 254 (16 進数 FE) (Modbus 例外コード)
- 255 (16 進数 FF) (拒否)

この表は %READ_VARi.OperError ダブルワードオブジェクトのエラーコードを示します。

CommError	名前	検出されたエラーコード	説明
0 (16 進数 00) (正常)	OperationOK	0 (16 進数 00000000)	通信は正常です。
	NotProcessed	1 (16 進数 00000001)	リクエストは処理されていません。
	BadResponse	2 (16 進数 00000002)	受信した応答が不正です。

CommError	名前	検出されたエラーコード	説明
254 (16 進数 FE) (Modbus 例外コード)	IllegalFunction	1 (16 進数 00000001)	リクエストで受信したファンクションコードは、スレーブに対して許可されたアクションではありません。スレーブは特定のリクエストを処理する状態にない可能性があります。
	IllegalDataAddress	2 (16 進数 00000002)	スレーブが受信したデータアドレスがスレーブに対して許可されたアドレスではありません。
	IllegalDataValue	3 (16 進数 00000003)	リクエストデータフィールドの値は、スレーブが許可した値ではありません。
	SlaveDeviceFailure	4 (16 進数 00000004)	スレーブは重大なエラーのためにリクエストされたアクションを実行できません。
	Acknowledge	5 (16 進数 00000005)	スレーブはリクエストを確認しましたが、スレーブが実施する前に通信がタイムアウトしました。
	SlaveDeviceBusy	6 (16 進数 00000006)	スレーブは別のコマンドを処理中です。
	MemoryParityError	8 (16 進数 00000008)	スレーブが拡張メモリーを読み込む際、メモリー内のパリティエラーを検出しました。
	GatewayPathUnavailable	10 (16 進数 0000000A)	ゲートウェイがオーバーロードしているか、正しく設定されていません。
	GatewayTargetDeviceFailedToRespond	11 (16 進数 0000000B)	スレーブがネットワーク上に存在しません。
255 (16 進数 FF) (拒否)	TargetResourceMissing	1 (16 進数 00000001)	ターゲットシステムリソースがありません。
	BadLength	5 (16 進数 00000005)	長さが不正です。
	CommChannelErr	6 (16 進数 00000006)	通信チャンネルでエラーが検出されました。
	BadAddr	7 (16 進数 00000007)	アドレスが不正です。
	SystemResourceMissing	11 (16 進数 0000000B)	システムリソースがありません。
	TargetCommInactive	12 (16 進数 0000000C)	ターゲット通信機能が有効ではありません。
	TargetMissing	13 (16 進数 0000000D)	ターゲットが通信不能です。
	ChannelNotConfigured	15 (16 進数 0000000F)	チャンネルが設定されていません。

ファンクション設定

プロパティ

ファンクションブロックをダブルクリックして、ファンクションプロパティテーブル *(Compact Modbus SL ロジックコントローラー M221 Book, System User Guide 参照)* を開きます。

このファンクションブロックのプロパティは、オンラインモードでは変更できません。

%READ_VAR ファンクションブロックのプロパティを次に示します。

プロパティ	値	説明
使用	有効または無効のチェックボックス	アドレスが使用中かどうかを示します。
アドレス	%READ_VARi の i は 0 からロジックコントローラーで使用可能なオブジェクトの数までです。	i はインスタンス識別子です。インスタンスの最大数については オブジェクトの最大数 *(Modicon M221, ロジックコントローラープログラミングガイド)* を参照してください。
シンボル	ユーザー定義テキスト	シンボルはオブジェクト固有の識別子です。詳細については、SoMachine Basic オペレーティングガイド (シンボルの定義と使用) *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)*
Link	• **SL1**: シリアル 1 • **SL2**: シリアル 2 • **ETH1**:Ethernet	ポートの選択。 **注記：** 標準通信ポート **SL2** および **ETH1** は、特定のコントローラーでのみ使用できます。
Id	このパラメーターはリンク設定によって異なります。 • シリアルラインスレーブアドレスの場合は 1...247 • Ethernet インデックスの場合は 1...16	デバイス識別子。 Ethernet インデックスの詳細については、リモートサーバーの追加 *(Modicon M221, ロジックコントローラープログラミングガイド)* を参照してください。
タイムアウト	100 ms 単位で指定され、初期値は 100 (10 s) です。 値が 0 の場合はタイムアウトが強制されません。	タイムアウトは応答を受信するまで待機する最大時間の設定です。 タイムアウトになると、通信はエラーとなって終了しエラーコードを生成します (CommError = 16 進数 01)。タイムアウト後、システムが応答を受信してもこの応答は無視されます。 **注記：** ファンクションブロックに設定されたタイムアウトは、SoMachine Basic の設定画面に設定された値を上書きします (Modbus TCP 設定 *(Modicon M221, ロジックコントローラープログラミングガイド)* および シリアルライン設定 *(Modicon M221, ロジックコントローラープログラミングガイド)*)。
ObjType	読み込むオブジェクトの型 • 0 : ワード • 1 : デジタル入力 • 2 : デジタル出力 • 3 : 入力ワード	Modbus 読み込みファンクションコードの種類は次のとおりです。 • **Mbs 0x03 - 複数のワードの読み込み (保持レジスター)** • **Mbs 0x02 - 複数のビットの読み込み (デジタル入力)** • **Mbs 0x01 - 複数のビットの読み込み (デジタル出力)** • **Mbs 0x04 - 複数のワードの読み込み (入力レジスター)**
FirstObj	0...65535	値を読み取るリモートデバイス上の最初のオブジェクトのアドレス。
Quantity	• **%MW** : 1...125 • **%I** : 1...2000 • **%Q** または **%M** : 1...2000 • **%IW** : 1...125	リモートデバイスから読み込むオブジェクトの数。

プロパティ	値	説明
IndexData	0...7999	読み込んだ値を格納するローカルワードテーブル (%MW) のアドレス。 ビット (%I または %Q) を読み込むとき、取得されたビットは指定された先頭アドレスから始まるワードテーブルに書き込まれます。例えば、**IndexData** = 10 および **Quantity** = 16 の 16 ビットを読み取る場合、結果は %MW10:X0 ～ %MW10:X15 に格納されます
コメント	ユーザー定義テキスト	オブジェクトに対応するコメント。

オブジェクト

%READ_VAR ファンクションブロックには次のオブジェクトがあります。

オブジェクト	説明	値
%READ_VARi.LINK	ポートの選択	プロパティ (*202* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%READ_VARi.ID	リモートデバイスの識別子	プロパティ (*202* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%READ_VARi.TIMEOUT	ファンクションブロックタイムアウト	プロパティ (*202* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%READ_VARi.OBJTYPE	読み込むオブジェクトのタイプ	プロパティ (*202* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%READ_VARi.FIRSTOBJ	値を読み取るリモートデバイス上の最初のオブジェクトのアドレス。	プロパティ (*202* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%READ_VARi.QUANTITY	リモートデバイスから読み込むオブジェクトの数。	プロパティ (*202* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%READ_VARi.INDEXDATA	読み込んだ値を格納するローカルワードテーブル (%MW) のアドレス。	プロパティ (*202* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%READ_VARi.COMMERROR	通信エラーコード	通信エラーコード (*200* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。
%READ_VARi.OPERERROR	動作エラーコード	動作エラーコード (*200* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。
%READ_VARi.DONE	実行は正常に完了しました。	出力 (*199* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。
%READ_VARi.BUSY	実行中です	出力 (*199* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。

オブジェクト	説明	値
%READ_VARi.ABORTED	実行がキャンセルされました	出力 (*199* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。
%READ_VARi.ERROR	エラーが検出されました	出力 (*199* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。

プログラミング例

概要

%READ_VAR ファンクションブロックはこのプログラミング例に示すように設定することができます。

プログラミング

次の例は %READ_VAR ファンクションブロックです。

ラング	命令
0	BLK %READ_VAR0 LD %I0.0 EXECUTE LD %I0.1 ABORT OUT_BLK LD DONE ST %Q0.0 LD BUSY ST %Q0.1 LD ABORTED ST %M1 LD ERROR ST %Q0.2 END_BLK

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

タイミングチャート

通信ファンクションブロックのタイミングチャート (*236* ページ参照)

9.2
データを Modbus デバイス (%WRITE_VAR) に書き込む

%WRITE_VAR ファンクションブロックの使用

このセクションでは %WRITE_VAR ファンクションブロックのプログラミングガイドラインと説明をします。

このセクションについて

このセクションには次の項目が含まれています。

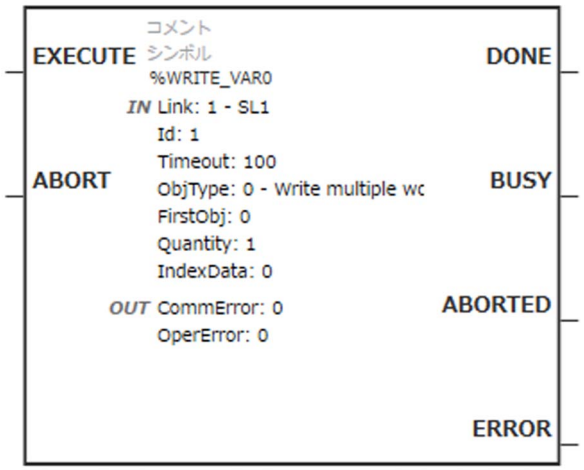
説明

概要

%WRITE_VAR ファンクションブロックは、Modbus SL、または Modbus TCP プロトコルを使用した外部機器へのデータ書き込みに使われます。

図

次の図は %WRITE_VAR ファンクションブロックを示しています。



入力

%WRITE_VAR ファンクションブロックには次の入力があります。

ラベル	タイプ	値
Execute	BOOL	立上がりが検出されたときにファンクションブロックの実行を開始します。 ファンクションブロックの実行中に 2 度目の立上がりが検出されても、それは無視されます。また実行中のコマンドは影響を受けません。
Abort	BOOL	立上がりが検出されたときにファンクションブロックの実行を停止します。 **Aborted** 出力は 1 に設定され、%WRITE_VARi.CommError オブジェクトはコード 16 進数 02 (ユーザーリクエストにより通信停止) を含みます。

注記: RUN の最初のタスクサイクルで **Execute** または **Abort** 入力を TRUE に設定しても、立上がりとして検出されません。ファンクションブロックが連続した立上がりを検出するためには、最初に入力を FALSE にしてください。

出力

%WRITE_VAR ファンクションブロックには次の出力があります。

ラベル	タイプ	値
Done	BOOL	TRUE の場合、ファンクションブロックの実行がエラーなしに完了したことを示します。
Busy	BOOL	TRUE の場合、ファンクションブロックの実行が進行中であることを示します。
Aborted	BOOL	TRUE の場合、ファンクションブロックの実行が **Abort** 入力でキャンセルされたことを示します。
Error	BOOL	TRUE の場合、エラーが検出されたことを示します。ファンクションブロックの実行は停止されます。 CommError および OperError の詳細については、通信エラーコード (*200* ページ参照) および 動作エラーコード (*200* ページ参照) の表を参照してください。

ファンクションブロックの出力オブジェクトを次の表に示します。

出力オブジェクト	タイプ	説明
CommError	BYTE	%READ_VARi.CommError の詳細は、通信エラーコード (*200* ページ参照) を参照してください。
OperError	DWORD	%READ_VARi.OperError の詳細は、動作エラーコード (*200* ページ参照) を参照してください。

通信エラーコード

通信エラーコード (*200* ページ参照) を参照してください。

動作エラーコード

動作エラーコード (*200* ページ参照) を参照してください。

ファンクション設定

プロパティ

ファンクションブロックをダブルクリックして、ファンクションプロパティテーブルを開きます。
このファンクションブロックのプロパティは、オンラインモードでは変更できません。
%WRITE_VAR ファンクションブロックのプロパティを次に示します。

プロパティ	値	説明
使用	有効または無効のチェックボックス	アドレスが使用中かどうかを示します。
アドレス	%WRITE_VARi の i は 0 からロジックコントローラーで使用可能なオブジェクトの数までです。	i はインスタンス識別子です。インスタンスの最大数については オブジェクトの最大数の表 *(Modicon M221, ロジックコントローラープログラミングガイド)* を参照してください。
シンボル	ユーザー定義テキスト	シンボルはオブジェクト固有の識別子です。詳細については、SoMachine Basic オペレーティングガイド (シンボルの定義と使用) *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)*
Link	● **SL1**: シリアル 1 ● **SL2**: シリアル 2 ● **ETH1**:Ethernet	ポートの選択。 **注記**：標準通信ポート **SL2** および **ETH1** は、特定のコントローラーでのみ使用できます。
Id	このパラメーターはリンク設定によって異なります。 ● 一斉送信の場合 0 ● シリアルラインスレーブアドレスの場合は 1...247 ● Ethernet インデックスの場合は 1...16	デバイス識別子。 値 0 の場合、Modbus マスターコントローラーは接続されているすべてのスレーブに一斉送信を開始します。一斉送信モードでは、スレーブはマスターに応答しません。 Ethernet インデックスの詳細については、リモートサーバーの追加 *(Modicon M221, ロジックコントローラープログラミングガイド)* を参照してください。
タイムアウト	100 ms 単位で指定され、初期値は 100 (10 s) です。 値が 0 の場合はタイムアウトが強制されません。	タイムアウトは応答を受信するまで待機する最大時間の設定です。 タイムアウトになると、通信はエラーとなって終了しエラーコードを生成します (CommError = 16 進数 01)。タイムアウト後、システムが応答を受信してもこの応答は無視されます。 **注記**：ファンクションブロックに設定されたタイムアウトは、SoMachine Basic の設定画面に設定された値を上書きします (Modbus TCP 設定 *(Modicon M221, ロジックコントローラープログラミングガイド)* および シリアルライン設定 *(Modicon M221, ロジックコントローラープログラミングガイド)*)。

プロパティ	値	説明
ObjType	次に示す Modbus ファンクションコードがサポートされています。 ● 0 ● 2 ● 4 ● 5	書き込むオブジェクトのタイプ ● **Mbs 0x10 - 複数のワードの書き込み (レジスター)** ● **Mbs 0x0F - 複数のビットの書き込み (デジタル出力)** ● **Mbs 0x05 - 1 つのビットの書き込み (デジタル出力)** ● **Mbs 0x06 - 1 つのワードの書き込み (レジスター)** **注記：** シングルコイル (Mbs 0x05) または シングルレジスター (Mbs 0x06) の Modbus ファンクションコードを使用するには ファンクションレベル を 5.0 以上に設定してください。
FirstObj	0...65535	値が書き込まれるリモートデバイス上の最初のオブジェクトのアドレス。
Quantity	● **%MW** の場合 1 ... 123 (内部レジスタ) ● **%M** または **%Q** の場合 1 ... 1968 (内部ビット)	リモートデバイスに書き込むオブジェクトの数。 シングルコイルおよびシングルレジスタオブジェクトタイプでは無視されます。
IndexData	0...7999	リモートデバイスに書き込む値を格納するローカルワードテーブルのアドレス (%MW)。 ビット (%M または %Q) への書き込みは、指定された先頭のアドレスから始まるワードテーブルから取得された値で書き込まれます。例えば、**IndexData** = 10 および **Quantity** = 16 の 16 ビットを書き込む場合、値は %MW10:X0 ～ %MW10:X15 から取得されます。
コメント	ユーザー定義テキスト	オブジェクトに対応するコメント。

オブジェクト

%WRITE_VAR ファンクションブロックには次のオブジェクトがあります。

オブジェクト	説明	値
%WRITE_VARi.LINK	ポートの選択	プロパティ (*209* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%WRITE_VARi.ID	リモートデバイスの識別子	プロパティ (*209* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%WRITE_VARi.TIMEOUT	ファンクションブロックタイムアウト	プロパティ (*209* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%WRITE_VARi.OBJTYPE	書き込むオブジェクトのタイプ	プロパティ (*209* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%WRITE_VARi.FIRSTOBJ	値が書き込まれるリモートデバイス上の最初のオブジェクトのアドレス。	プロパティ (*209* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。

オブジェクト	説明	値
%WRITE_VARi.QUANTITY	リモートデバイスに書き込むオブジェクトの数。	プロパティ (*209* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%WRITE_VARi.INDEXDATA	リモートデバイスに書き込む値を格納するローカルワードテーブルのアドレス (%MW)。	プロパティ (*209* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%WRITE_VARi.COMMERROR	通信エラーコード	通信エラーコード (*208* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。
%WRITE_VARi.OPERERROR	動作エラーコード	動作エラーコード (*208* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。
%WRITE_VARi.DONE	実行は正常に完了しました。	出力 (*207* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。
%WRITE_VARi.BUSY	実行中です	出力 (*207* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。
%WRITE_VARi.ABORTED	実行がキャンセルされました	出力 (*207* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。
%WRITE_VARi.ERROR	エラーが検出されました	出力 (*207* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。

プログラミング例

概要

%WRITE_VAR ファンクションブロックはこのプログラミング例に示すように設定することができます。

プログラミング

次の例は %WRITE_VAR ファンクションブロックです。

ラング	命令
0	BLK %WRITE_VAR0 LD %I0.0 EXECUTE LD %I0.1 ABORT OUT_BLK LD DONE ST %Q0.0 LD BUSY ST %Q0.1 LD ABORTED ST %M1 LD ERROR ST %Q0.2 END_BLK

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

タイミングチャート

通信ファンクションブロックのタイミングチャート (*236* ページ参照)

9.3
Modbus デバイスのデータの読み書き (%WRITE_READ_VAR)

%WRITE_READ_VAR ファンクションブロックの使用

このセクションでは %WRITE_READ_VAR ファンクションブロックのプログラミングガイドラインと説明をします。

このセクションについて

このセクションには次の項目が含まれています。

説明

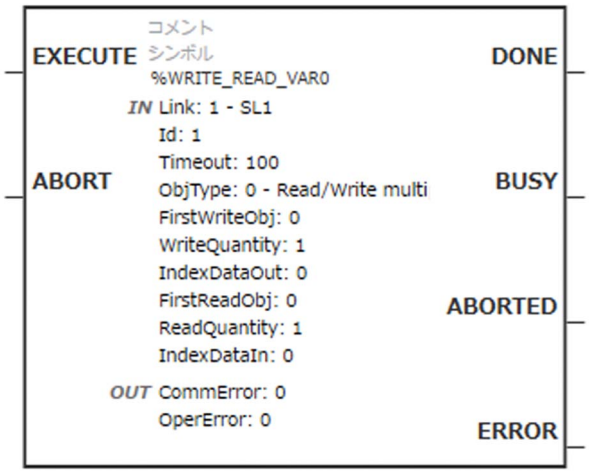
概要

%WRITE_READ_VAR ファンクションブロックは、内部ワードメモリーのデータの読み込みや、Modbus SL、または Modbus TCP プロトコル を使用した外部デバイスへの書き込みに使われます。

このファンクションブロックは同じ処理過程で 1 つの書き込みリクエストとそれに続く 1 つの読み取りリクエストを実行します。

図

次の図は %WRITE_READ_VAR ファンクションブロックを示しています。



入力

%WRITE_READ_VAR ファンクションブロックには次の入力があります。

ラベル	タイプ	値
Execute	BOOL	立上がりが検出されたときにファンクションブロックの実行を開始します。ファンクションブロックの実行中に 2 度目の立上がりが検出されても、それは無視されます。また実行中のコマンドは影響を受けません。
Abort	BOOL	立上がりが検出されたときにファンクションブロックの実行を停止します。**Aborted** 出力は 1 に設定され、%WRITE_READ_VARi.CommError オブジェクトはコード 16 進数 02 (ユーザーリクエストにより通信停止) を含みます。

注記 : RUN の最初のタスクサイクルで **Execute** または **Abort** 入力を TRUE に設定しても、立上がりとして検出されません。ファンクションブロックが連続した立上がりを検出するためには、最初に入力を FALSE にしてください。

出力

%WRITE_READ_VAR ファンクションブロックには次の出力があります。

ラベル	タイプ	値
Done	BOOL	TRUE の場合、ファンクションブロックの実行がエラーなしに完了したことを示します。
Busy	BOOL	TRUE の場合、ファンクションブロックの実行が進行中であることを示します。
Aborted	BOOL	TRUE の場合、ファンクションブロックの実行が **Abort** 入力でキャンセルされたことを示します。
Error	BOOL	TRUE の場合、エラーが検出されたことを示します。ファンクションブロックの実行は停止されます。 CommError および OperError の詳細については、通信エラーコード (*200* ページ参照) および 動作エラーコード (*200* ページ参照) の表を参照してください。

ファンクションブロックの出力オブジェクトを次の表に示します。

出力オブジェクト	タイプ	説明
CommError	BYTE	%READ_VARi.CommError の詳細は、通信エラーコード (*200* ページ参照) を参照してください。
OperError	DWORD	%READ_VARi.OperError の詳細は、動作エラーコード (*200* ページ参照) を参照してください。

通信エラーコード

通信エラーコード (*200* ページ参照) を参照してください。

動作エラーコード

動作エラーコード (*200* ページ参照) を参照してください。

ファンクション設定

プロパティ

ファンクションブロックをダブルクリックして、ファンクションプロパティテーブルを開きます。
このファンクションブロックのプロパティは、オンラインモードでは変更できません。
%WRITE_READ_VAR ファンクションブロックのプロパティを次に示します。

プロパティ	値	説明
使用	有効または無効のチェックボックス	アドレスが使用中かどうかを示します。
アドレス	%WRITE_READ_VARi の i は 0 からロジックコントローラーで使用可能なオブジェクトの数までです。	i はインスタンス識別子です。インスタンスの最大数については オブジェクトの最大数の表 *(Modicon M221, ロジックコントローラープログラミングガイド)* を参照してください。
シンボル	ユーザー定義テキスト	シンボルはオブジェクト固有の識別子です。詳細については、SoMachine Basic オペレーティングガイド (シンボルの定義と使用) *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)*
Link	• **SL1**: シリアル 1 • **SL2**: シリアル 2 • **ETH1**:Ethernet	ポートの選択 **注記**: 標準通信ポート **SL2** および **ETH1** は、特定のコントローラーでのみ使用できます。
Id	このパラメーターはリンク設定によって異なります。 • シリアルラインスレーブアドレスの場合は 1...247 • Ethernet インデックスの場合は 1...16	デバイス識別子 Ethernet インデックスの詳細については、リモートサーバーの追加 *(Modicon M221, ロジックコントローラープログラミングガイド)* を参照してください。
タイムアウト	100 ms 単位で指定され、初期値は 100 (10 s) です。 値が 0 の場合はタイムアウトが強制されません。	タイムアウトは応答を受信するまで待機する最大時間の設定です。 タイムアウトになると、通信はエラーとなって終了しエラーコードを生成します (CommError = 16 進数 01)。タイムアウト後、システムが応答を受信してもこの応答は無視されます。 **注記**: ファンクションブロックに設定されたタイムアウトは、SoMachine Basic の設定画面に設定された値を上書きします (Modbus TCP 設定 *(Modicon M221, ロジックコントローラープログラミングガイド)* および シリアルライン設定 *(Modicon M221, ロジックコントローラープログラミングガイド)*)。
ObjType	**%MW (Mbs Fct 17)**: ワードメモリー	Modbus 読み込み / 書き込みファンクションコードの型は、Modbus ファンクションコード 17 に相当する **Mbs Fct 17** です。
FirstWriteObj	0...65535	値が書き込まれるリモートデバイス上の最初のオブジェクトのアドレス。
WriteQuantity	1...121	リモートデバイスに書き込むオブジェクトの数。
IndexDataOut	0...7999	リモートデバイスに書き込む値を格納するローカルワードテーブルのアドレス (%MW)。
FirstReadObj	0...65535	値を読み取るリモートデバイス上の最初のオブジェクトのアドレス。
ReadQuantity	1...125	リモートデバイスから読み込むオブジェクトの数。
IndexDataIn	0...7999	読み込んだ値を格納するローカルワードテーブル (%MW) のアドレス。
コメント	ユーザー定義テキスト	オブジェクトに対応するコメント。

オブジェクト

%WRITE_READ_VAR ファンクションブロックには次のオブジェクトがあります。

オブジェクト	説明	値
%WRITE_READ_VARi.LINK	ポートの選択	プロパティ (*216* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%WRITE_READ_VARi.ID	リモートデバイスの識別子	プロパティ (*216* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%WRITE_READ_VARi.TIMEOUT	ファンクションブロックタイムアウト	プロパティ (*216* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%WRITE_READ_VARi.OBJTYPE	読み込むオブジェクトのタイプ	プロパティ (*216* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%WRITE_READ_VARi.FIRSTWRITEOBJ	値が書き込まれるリモートデバイス上の最初のオブジェクトのアドレス。	プロパティ (*216* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%WRITE_READ_VARi.WRITEQUANTITY	リモートデバイスに書き込むオブジェクトの数。	プロパティ (*216* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%WRITE_READ_VARi.INDEXDATAOUT	リモートデバイスに書き込む値を格納するローカルワードテーブルのアドレス (%MW)。	プロパティ (*216* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%WRITE_READ_VARi.FIRSTREADOBJ	値を読み取るリモートデバイス上の最初のオブジェクトのアドレス。	プロパティ (*216* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%WRITE_READ_VARi.READQUANTITY	リモートデバイスから読み込むオブジェクトの数。	プロパティ (*216* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%WRITE_READ_VARi.INDEXDATAIN	読み込んだ値を格納するローカルワードテーブル (%MW) のアドレス。	プロパティ (*216* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%WRITE_READ_VARi.COMMERROR	通信エラーコード	通信エラーコード (*215* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。
%WRITE_READ_VARi.OPERERROR	動作エラーコード	動作エラーコード (*215* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。
%WRITE_READ_VARi.DONE	実行は正常に完了しました。	出力 (*214* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。
%WRITE_READ_VARi.BUSY	実行中です	出力 (*214* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。

オブジェクト	説明	値
%WRITE_READ_VARi.ABORTED	実行がキャンセルされました	出力 (*214* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。
%WRITE_READ_VARi.ERROR	エラーが検出されました	出力 (*214* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。

プログラミング例

概要

%WRITE_READ_VAR ファンクションブロックはこのプログラミング例に示すように設定することができます。

プログラミング

次の例は %WRITE_READ_VAR ファンクションブロックです。

ラング	命令
0	BLK %WRITE_READ_VAR0 LD %I0.0 EXECUTE LD %I0.1 ABORT OUT_BLK LD DONE ST %Q0.0 LD BUSY ST %Q0.1 LD ABORTED ST %M1 LD ERROR ST %Q0.2 END_BLK

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

タイミングチャート

通信ファンクションブロックのタイミングチャート (*236* ページ参照)

9.4
ASCII リンク通信 (%SEND_RECV_MSG)

%SEND_RECV_MSG ファンクションブロックの使用

このセクションでは %SEND_RECV_MSG ファンクションブロックのプログラミングガイドラインと説明をします。

このセクションについて

このセクションには次の項目が含まれています。

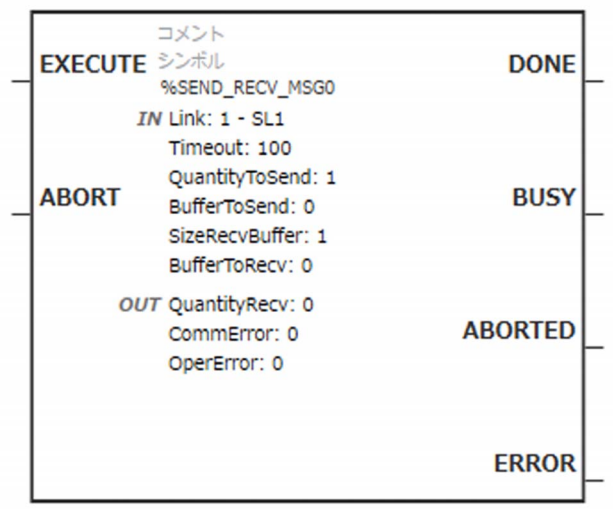
説明

概要

%SEND_RECV_MSG ファンクションブロックは、ASCII プロトコルに設定されたシリアルラインのデータ送受信に使われます。

図

次の図は %SEND_RECV_MSG ファンクションブロックを示しています。



入力

%SEND_RECV_MSG ファンクションブロックには次の入力があります。

ラベル	タイプ	値
Execute	BOOL	立上がりが検出されたときにファンクションブロックの実行を開始します。ファンクションブロックの実行中に 2 度目の立上がりが検出されても、それは無視されます。また実行中のコマンドは影響を受けません。
Abort	BOOL	立上がりが検出されたときにファンクションブロックの実行を停止します。**Aborted** 出力は 1 に設定され、%SEND_RECV_MSGi.CommError オブジェクトはコード 16 進数 02 (ユーザーリクエストにより通信停止) を含みます。

注記: RUN の最初のタスクサイクルで **Execute** または **Abort** 入力を TRUE に設定しても、立上がりとして検出されません。ファンクションブロックが連続した立上がりを検出するためには、最初に入力を FALSE にしてください。

出力

%SEND_RECV_MSG ファンクションブロックには次の出力があります。

ラベル	タイプ	値
Done	BOOL	TRUE の場合、ファンクションブロックの実行がエラーなしに完了したことを示します。
Busy	BOOL	TRUE の場合、ファンクションブロックの実行が進行中であることを示します。
Aborted	BOOL	TRUE の場合、ファンクションブロックの実行が **Abort** 入力でキャンセルされたことを示します。
Error	BOOL	TRUE の場合、エラーが検出されたことを示します。ファンクションブロックの実行は停止されます。 CommError および OperError の詳細については、通信エラーコード (*200* ページ参照) および 動作エラーコード (*200* ページ参照) の表を参照してください。

通信エラーコード

通信エラーコード (*200* ページ参照) を参照してください。

動作エラーコード

動作エラーコード (*200* ページ参照) を参照してください。

終了条件

送信のみ動作の場合すべてのデータ (開始 / 停止文字を含む) が送信されると **Done** 出力は、TRUE になります。

受信のみ動作の場合、システムは終了条件が満たされるまで文字を受信します。終了条件に達すると、**Done** 出力は TRUE に設定されます。受け取った文字列は **sizeRecvBuffer** 文字まで **BufferToRecv** にコピーされます。**sizeRecvBuffer** は終了条件ではありません。

シリアル回線設定画面 *(Modicon M221, ロジックコントローラープログラミングガイド)* で終了条件を設定してください。

終了条件は次のように設定できます。

- 受信バイト数 : **受信フレーム長**
- 無通信フレームの終わり : **フレーム受信タイムアウト (ms)**
- フレーム構造 : **最初の終了文字**

送信 / 受信動作では文字が最初に送られ、その後、終了条件が満たされるまで (受信のみ動作と同じ) 文字が受信されます。

ファンクション設定

プロパティ

ファンクションブロックをダブルクリックして、ファンクションプロパティテーブルを開きます。
このファンクションブロックのプロパティは、オンラインモードでは変更できません。
%SEND_RECV_MSG ファンクションブロックのプロパティを次に示します。

プロパティ	値	説明
使用	有効または無効のチェックボックス	アドレスが使用中かどうかを示します。
アドレス	%SEND_RECV_MSGi の i は 0 からロジックコントローラーで使用可能なオブジェクトの数までです。	i はインスタンス識別子です。使用可能なインスタンスの最大値については、ロジックコントローラーのプログラミングガイドを参照してください。
シンボル	ユーザー定義テキスト	シンボルはオブジェクト固有の識別子です。詳細については、SoMachine Basic オペレーティングガイド (シンボルの定義と使用) *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)*
Link	• **SL1**: シリアル 1 • **SL2**: シリアル 2	ポートの選択 **注記**：標準通信ポート **SL2** は特定のコントローラーでのみ使用できます。
タイムアウト	100 ms 単位で指定され、初期値は 100 (10 s) です。 値が 0 の場合はタイムアウトが強制されません。	タイムアウトは応答を受信するまで待機する最大時間の設定です。 タイムアウトになると、通信はエラーとなって終了しエラーコードを生成します (CommError = 16 進数 01)。タイムアウト後、システムが応答を受信してもこの応答は無視されます。 **注記**：ファンクションブロックに設定されたタイムアウトは SoMachine Basic 設定画面 (*Modbus TCP 設定* および *シリアルライン設定*、ロジックコントローラーのプログラミングガイドを参照してください) に設定されている値よりも優先されます。
QuantityToSend	0...254 0 の値は、ファンクションブロックがデータの受信のみ行うことを意味します。	送信するバイト数
BufferToSend	0...7999	送信する先頭オブジェクトのアドレス
SizeRecvBuffer	0...254 0 の値は、ファンクションブロックがデータの送信のみ行うことを意味します。	使用可能な受信バッファーのバイトサイズ
BufferToRecv	0...7999	読み取った値が格納されるワードテーブルの先頭アドレス (%MW)。
QuantityRecv	0...254	受信したデータのバイト数
コメント	ユーザー定義テキスト	オブジェクトに対応するコメント。

オブジェクト

%SEND_RECV_MSG ファンクションブロックには次のオブジェクトがあります。

オブジェクト	説明	値
%SEND_RECV_MSGi.LINK	ポートの選択	プロパティ (*223* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%SEND_RECV_MSGi.TIMEOUT	ファンクションブロックタイムアウト	プロパティ (*223* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%SEND_RECV_MSGi.QUANTITYTOSEND	送信するバイト数	プロパティ (*223* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%SEND_RECV_MSGi.BUFFERTOSEND	送信する先頭オブジェクトのアドレス	プロパティ (*223* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%SEND_RECV_MSGi.SIZERECVBUFFER	使用可能な受信バッファーのバイトサイズ	プロパティ (*223* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%SEND_RECV_MSGi.BUFFERTORECV	読み取った値を格納するワードテーブルの先頭アドレス	プロパティ (*223* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%SEND_RECV_MSGi.QUANTITYRECV	受信したデータのバイト数	プロパティ (*223* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで編集できます。
%SEND_RECV_MSGi.COMMERROR	通信エラーコード	通信エラーコード (*222* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。
%SEND_RECV_MSGi.OPERERROR	動作エラーコード	動作エラーコード (*222* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。
%SEND_RECV_MSGi.DONE	実行は正常に完了しました。	出力 (*221* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。
%SEND_RECV_MSGi.BUSY	実行中です	出力 (*221* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。
%SEND_RECV_MSGi.ABORTED	実行がキャンセルされました	出力 (*221* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。
%SEND_RECV_MSGi.ERROR	エラーが検出されました	出力 (*221* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで読み込めます。

プログラミング例

概要

%SEND_RECV_MSG ファンクションブロックはこのプログラミング例に示すように設定することができます。

プログラミング

次の例は %SEND_RECV_MSG ファンクションブロックです。

ラング	命令
0	BLK %SEND_RECV_MSG0 LD %I0.0 EXECUTE LD %I0.1 ABORT OUT_BLK LD DONE ST %Q0.0 LD BUSY ST %Q0.1 LD ABORTED ST %M1 LD ERROR ST %Q0.2 END_BLK

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

タイミングチャート

通信ファンクションブロックのタイミングチャート (*236* ページ参照)

9.5
SMS の送受信 (%SEND_RECV_SMS)

%SEND_RECV_SMS ファンクションブロックの使用

このセクションでは %SEND_RECV_SMS ファンクションブロックのプログラミングガイドラインと説明をします。

このセクションについて

このセクションには次の項目が含まれています。

説明

概要

%SEND_RECV_SMS ファンクションブロックは、シリアルラインに接続されたモデムを介したショートメッセージサービス (SMS) メッセージの送受信に使われます。例えば、コントローラーは SMS を送信して、指定された携帯電話にアラームを送信することができます。また SMS を受信してマシンのファンクションを終了することができます。

注記：SMS 機能を使用するには、アプリケーションの ファンクションレベル *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を**レベル 3.2** 以上に設定してください。

%SEND_RECV_SMS ファンクションブロックは次のいずれかに使用されます。

- 1 人の受信者にのみ SMS を送信する、または
- 承認済電話番号のテーブルによってフィルタリングされた SMS を受信する。

1 つの %SEND_RECV_SMS ファクションブロックはプログラムで使用できます。

命令された機械の動作、コントローラーの状態の変化、またはデータメモリーや機械動作パラメーターの変更による意図しない影響を回避するために、遠隔制御装置として SMS 機能を使用する際は注意してください。

⚠ 警告

装置の意図しない動作

- 離れた場所から操作する場合は、現地に資格を保有する適格な監視者がいることを確認してください。
- 遠隔によるコントローラーへの命令の送信に関わらず、コントローラーの開始または停止に対するローカル制御を維持するために、アプリケーションに運転 / 停止入力を設定および設置してください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

SMS 機能は、外部電気通信ネットワークおよびパラメーターにより異なります。機械に送信された SMS コマンドおよびメッセージは、遅延したり、送信または受信されない場合があります。安全上重要な機能またはその他の重要な目的には SMS 機能を使用しないでください。

⚠ 警告

装置の意図しない動作

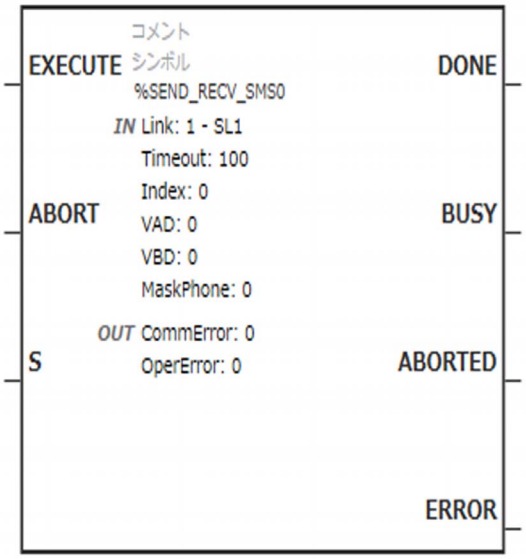
- SMS コマンドは安全上重要な機能には許可しないでください。
- 任務上重要な目的には SMS コマンドまたはメッセージを使用しないでください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

注記：試運転中に SMS 機能および関連する電気通信ネットワークを確認し、定期的にテストしてネットワークの有効範囲を確認してください。

図

次の図は %SEND_RECV_SMS ファンクションブロックを示しています。



入力

ファンクションブロックの入力を次の表に示します。

ラベル	タイプ	値
Execute	BOOL	立上がりが検出されたときにファンクションブロックの実行を開始します。 ファンクションブロックの実行中に 2 度目の立上がりが検出されても、それは無視されます。また実行中のコマンドは影響を受けません。
Abort	BOOL	立上がりが検出されたときにファンクションブロックの実行を停止します。 **Aborted** 出力は 1 に設定され、%SEND_RECV_SMSi.CommError オブジェクトにはコード 16 進数 02 (**Abort** 入力の立上がりで停止した Exchange) が含まれます。
S	BOOL	1 の場合、ファンクションブロックは SMS を送信するように設定されています。 0 の場合、ファンクションブロックは SMS を受信するように設定されています。

注記：RUN の最初のタスクサイクルで **Execute** または **Abort** 入力を 1 に設定しても、立上がりとして検出されません。ファンクションブロックが連続した立上がりを検出するためには、最初に入力を 0 にしてください。

ファンクションブロックの入力オブジェクトを次の表に示します。

入力オブジェクト	タイプ	値の範囲	説明
Link	BYTE	1 - SL1 2 - SL2	モデムを介して通信するために使用されるシリアル回線を示します。
タイムアウト	WORD	0...255	モデムから応答を受信するまで待機する最大時間を設定します。 100 ms 単位で指定され、初期値は 100 (10 s) です。値が 0 の場合はタイムアウトが強制されません。
インデックス	WORD	0...15	以下の場合インデックスの値が使用されます。 ● 送信中に **メッセージ**テーブルから送信するテキストを選択します。 0 はテーブルの最初の文字に対応します。 ● 受信中は受信したテキストと一致する**コマンド**テーブルの文字列に値が対応します。 一致する文字列が見つからない場合は、FFFF (16 進数) に設定します。

入力オブジェクト	タイプ	値の範囲	説明
VAD	DINT	-2147483648...2147483647	● 送信中に、SMS のテキスト内のプレースホルダー $VAD は %SEND_RECV_SMSi.VAD の値に置き換わります。 ● 受信中に、%SEND_RECV_SMSi.VAD の値は SMS の**コマンド**テーブル内に格納されているプレースホルダー $VAD の値を受け取ります。
VBD	DINT	-2147483648...2147483647	● 送信中に、SMS のテキスト内のプレースホルダー $VBD は %SEND_RECV_SMSi.VBD の値に置き換わります。 ● 受信中に、%SEND_RECV_SMSi.VBD の値は SMS の**コマンド**テーブル内に格納されているプレースホルダー $VBD の値を受け取ります。
MASKPHONE	WORD	0...15	● 送信中は、**電話番号**テーブルから SMS の受信者を選択するために、マスクは使用されます。 ● 受信中は、マスクが**電話番号**テーブルに適用され、有効な電話番号のリストが作成されます。

出力

ファンクションブロックの出力を次の表に示します。

ラベル	タイプ	値
Done	BOOL	TRUE の場合、ファンクションブロックの実行がエラーなしに完了したことを示します。
Busy	BOOL	TRUE の場合、ファンクションブロックの実行が進行中であることを示します。
Aborted	BOOL	TRUE の場合、ファンクションブロックの実行が %SEND_RECV_SMSi.Abort 入力でキャンセルされたことを示します。
Error	BOOL	TRUE の場合、エラーが検出されたことを示します。ファンクションブロックの実行は停止されます。 %SEND_RECV_SMSi.CommError および %SEND_RECV_SMSi.OperError の詳細については、通信エラーコード (*229* ページ参照) および 動作エラーコード (*230* ページ参照) の表を参照してください。
注記：Busy 出力が TRUE に設定されると **Done**、**Aborted**、または **Error** 出力のいずれかが TRUE に設定されるまで実行が続けられます。 **注記：Busy** 出力が TRUE に設定されている間は、**Execute** 入力の変更は進行中のファンクションブロックの実行に影響しません。ただし %SEND_RECV_SMS ファンクションブロックが呼び出されると、この SMS は拒否されます (CommError = 255 (16 進数 FF) および OperError = 11 (16 進数 0000000B))。		

ファンクションブロックの出力オブジェクトを次の表に示します。

調	タイプ	説明
CommError	BYTE	%SEND_RECV_SMSi.CommError の詳細は、通信エラーコード (*229* ページ参照) を参照してください。
OperError	DWORD	%SEND_RECV_SMSi.OperError の詳細は、動作エラーコード (*230* ページ参照) を参照してください。

通信エラーコード

この表は %SEND_RECV_SMSi.CommError 出力オブジェクトのエラーコードを示します。

10 進数 (16 進数) 検出エラーコード	名前	説明
0 (00 hex)	CommunicationOK	通信は正常です。 **注記：** この場合 %SEND_RECV_SMSi.OperError 出力オブジェクトは、エラーコードとは対照的にモデム信号レベルを含みます。
1 (01 hex)	TimedOut	タイムアウトにより通信は停止しました。
2 (02 hex)	Abort	Exchange は %SEND_RECV_SMSi.Abort 入力の立上がりで停止します。
3 (03 hex)	BadLink	リンクが不正です。
4 (04 hex)	BadCommand	コマンドが不正です。

10 進数 (16 進数) 検出エラーコード	名前	説明
5 (05 hex)	BadMgtTable	管理テーブルの書式が不正です。
6 (06 hex)	BadParameters	特定のパラメーターが不正です。
7 (07 hex)	ProblemSendingSms	SMS はコマンドを送信できませんでした。
9 (09 hex)	RecvCmdError	無効なコマンドです。
10 (0A hex)	SendValueError	無効な値です。
11 (0B hex)	SystemResourceMissing	システムリソースがありません。
14 (0E hex)	BadLength	長さが不正です。
254 (FE hex)	ProtocolSpecificError	オペレーティングシステムエラーが検出されたことを示します。 **注記：** この場合、%SEND_RECV_SMSi.OperError 出力オブジェクトには詳細が含まれています。動作エラーコード (*230* ページ参照) を参照してください。
255 (FF hex)	Refused	SMS は拒否されました。 **注記：** この場合、%SEND_RECV_SMSi.OperError 出力オブジェクトには詳細が含まれています。動作エラーコード (*230* ページ参照) を参照してください。

動作エラーコード

通信エラーコード (%SEND_RECV_SMSi.CommError 出力オブジェクト) が次の値をもつ場合、このリターンコードは重要です。

- 0 (16 進数 00) (正しいプロトコル)
- 254 (16 進数 FE) (不正なプロトコル)
- 255 (16 進数 FF) (SMS 拒否)

%SEND_RECV_SMSi.CommError が 0 (16 進数 00) (正しいプロトコル) の場合、
%SEND_RECV_SMSi.OperError 出力オブジェクトは RSSI (受信信号強度インジケーション /Received Signal Strength Indication) を表示します。

%SEND_RECV_SMSi.OperError オブジェクトの 10 進数値	RSSI モデム信号レベル
9 未満	限界値 (ワイヤレスネットワークを維持するのに必要な限界を超える減衰)
10 ～ 14	可
15 ～ 19	良
20 以上	優

%SEND_RECV_SMSi.CommError が 254 (16 進数 FE) (不正なプロトコル) の場合、
%SEND_RECV_SMSi.OperError 出力オブジェクトは詳細を返します。

%SEND_RECV_SMSi.OperError オブジェクトの 10 進数 (16 進数) の値	名前	説明
256 (00000100 hex)	ModemConfSLAsciiFailed	シリアル回線の ASCII 設定が不正です。
512 (00000200 hex)	ModemReconfSLFailed	ユーザー設定に戻るシリアル回線の設定が不正です。
768 (00000300 hex)	ModemBusy	モデムはダイヤルコマンドに対し BUSY を返します。
1024 (00000400 hex)	ModemNoDialtone	モデムはダイヤルコマンドに対し NODIALTONE を返します。
1280 (00000500 hex)	ModemNoCarrier	モデムキャリア信号が消失または切断されました。モデムはダイヤルコマンドに対し NO CARRIER を返します。
1536 (00000600 hex)	ModemBadAnswer	モデムからの応答が不正です

%SEND_RECV_SMSi.OperError オブジェクトの 10 進数 (16 進数) の値	名前	説明
SIM カードの使用特有のエラーがあります。		
4096 (00001000 hex)	SimConfigurationFailed	SIM カードの設定が不正です。例えば、PUK コードが必要です。
8192 (00002000 hex)	SimPinCodeInvalid	PIN コードが不正です
16384 (00004000 hex)	SimSmsCenterInvalid	SMS センターの電話番号が間違っています。

%SEND_RECV_SMSi.CommError が 255 (16 進数 FF) (SMS 拒否) の場合、%SEND_RECV_SMSi.OperError 出力オブジェクトは詳細を返します。

%SEND_RECV_SMSi.OperError オブジェクトの 10 進数 (16 進数) の値	名前	説明
1 (00000001 hex)	TargetResourceMissing	ターゲットシステムリソースがありません。
5 (00000005 hex)	BadLength	長さが不正です。
6 (00000006 hex)	CommChannelErr	通信チャンネルでエラーが検出されました。
11 (0000000B hex)	SystemResourceMissing	システムリソースがありません。
12 (0000000C hex)	TargetCommInactive	ターゲット通信機能が無効です。
13 (0000000D hex)	TargetMissing	ターゲットは使用できません。
15 (0000000F hex)	ChannelNotConfigured	通信チャンネルが設定されていません。
16 (00000010 hex)	PhoneNumberNotMatching	受信したメッセージの電話番号が承認済み番号のリスト (ホワイトリスト) と一致しません。
17 (00000011 hex)	MessageNotMatching	受信したメッセージがコマンドリスト内のどのメッセージとも一致しません。送信者の電話番号が承認済み番号のリスト (ホワイトリスト) に一致する場合にのみ発行されます。

ファンクション設定

メインステップ

モデムをシリアル回線に接続した後で、%SEND_RECV_SMS ファンクションブロックを設定する主な手順を次に示します。

手順	アクション
1	SoMachine Basic の **設定** タブで、モデム、**Init コマンド**、および ASCII プロトコルを使用してシリアル回線を設定します。詳細については、ロジックコントローラーのプログラミングガイドを参照してください。
2	モデムがコントローラーのシリアル回線に接続されていることを確認します。 ● SIM カードのロックは解除されており、PIN コードで保護されていません。 ● SMS センターの電話番号が SIM カードに正しく設定されている。
3	**プログラミング**タブで ● %SEND_RECV_SMS ファンクションブロックをダブルクリックし、ファンクションブロックをダブルクリックしてファンクションプロパティテーブルを表示します。 ● **SMS 設定**ボタンをクリックして **SMS アシスタント**ウィンドウを開きます。 ● **メッセージ、コマンド、電話番号**のテーブルを編集します。 詳細は **SMS アシスタント** を参照してください。 ● **適用**をクリックして **SMS アシスタント**を閉じます。 **注記：**ファンクションブロックアドレス (例えば、%SEND_RECV_SMS0) が無効でダブルクリックが無効になっている場合は、アプリケーションのファンクションレベル (**プログラミング** タブ > **タスク** > **動作**) が少なくとも**レベル 3.2** であることを確認してください。
4	**プログラミング**タブで、ファンクションプロパティテーブルのフィールドを編集します。 これらのフィールドの詳細についてはプロパティ (*233* ページ参照) を参照してください。

⚠ 警告

装置の意図しない動作

ファンクションブロックを使用する前に、ファンクションブロックで使用されているメッセージ、コマンド、および電話番号のインデックスが有効であることを確認してください (使用する予定のもの)。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

SR2MOD03 モデムの設定およびインストールに関する詳細は *SR2MOD02 and SR2MOD03 Wireless Modem User Guide* (EIO00000001575) を参照してください

SMS アシスタント

%SEND_RECV_SMS ファンクションブロック、コマンド、メッセージ、および電話番号テーブルの設定を使用するには次を実行します。

送受信 SMS プロパティ領域の **SMS 構成**ボタンをクリックして **SMS アシスタント**ウィンドウを表示します。

SMS アシスタントウィンドウには設定するテーブルを含む 3 つのタブがあります。

- **メッセージ**
  コントローラーが SMS を送信するときに使用する文字列を入力します。変数、日付および時刻の入力にはプレースホルダーを使用します。文字数とフォーマットの制限を考慮してください。
- **コマンド**
  コントローラーが SMS を受信するときに使用する文字列を入力します。変数にはプレースホルダーを使用します。文字数とフォーマットの制限を考慮してください。
- **電話番号**
  ファンクションブロックをプログラミングして SMS メッセージを送信するときは、このテーブルから受信者を選択してください。
  ファンクションブロックをプログラミングして SMS メッセージを受信するときは、このリストから許可された発信電話番号を選択します。選択した電話番号リストは、アプリケーションのセキュリティを強化します。通話と後続の SMS がモデム経由でアプリケーションに送信されると、着信 SMS を処理する前に発信元の電話番号が検証されます。
  詳細は プロパティ (*233* ページ参照) の **MASKPHONE** 行を参照してください。

注記：国際ダイヤルコード形式については、モデムのマニュアルを参照してください。

各テーブルには、最大 16 のエントリが含まれ、0 ～ 15 の各行にインデックスがあります。

アシスタントテーブルに含まれる文字列は、次の書式とメッセージサイズの制限を参照してください。

文字書式	**メッセージ および コマンドテーブル**
GSM 7-bit	最大 105 文字
UNICODE	最大 45 文字
注記：文字の書式はテキストフィールドの文字によって自動的に決まります。	

変数として解釈される**メッセージ**または**コマンド**のテキストには次に示すプレースホルダーを追加できます。

プレースホルダー	**実行時に次のものと置き換えられます。**	**GSM 7-bit フォーマット使用時の文字数**	**UNICODE フォーマット使用時の文字数**
$DATE(1)	YY/MM/DD (現在の日付)	8 + 1	16 + 2
$TIME(1)	HH:MM:SS (現在の時間)	8 + 1	16 + 2
$VAD	DWORD 値 のパラメーター %SEND_RECV_SMSi.VAD が変換されたテキスト	最大 12	最大 24
$VBD	DWORD 値 のパラメーター %SEND_RECV_SMSi.VBD が変換されたテキスト	最大 12	最大 24
$$	シンボル $	1	2
注記：入力されたテキストが有効な場合 (文字の制限を超えない場合、有効なプレースホルダー)、**適用** ボタンがアクティブになります。 **(1) コマンド**で無視されます			

例

メッセージでプレースホルダーを使用する方法を次の例に示します。

メッセージ	
設定されたメッセージ	$DATE : $TIME - Value A = $VAD and Value B = $VBD !
プレースホルダー値	VAD = 10、VBD = 2000
最終的に送信された SMS	15/04/27 : 11:15:43 - Value A = 10 and Value B = 2000 !

コマンドでプレースホルダーを使用する方法を次の例に示します。

コマンド	
設定されたコマンド	Value A = $VAD and Value B = $VBD !
受信した SMS	Value A = 300 and Value B = 2 !
取得された値	VAD = 300、VBD = 2

プロパティ

ファンクションブロックをダブルクリックして、ファンクションプロパティテーブルを開きます。

このファンクションブロックのプロパティは、オンラインモードでは変更できません。

%SEND_RECV_SMS ファンクションブロックのプロパティを次に示します。

プロパティ	**値**	**説明**
使用	有効または無効のチェックボックス	アドレスが使用中かどうかを示します。
アドレス	%SEND_RECV_SMSi の i は 0 からロジックコントローラーで使用可能なオブジェクトの数までです。	i はインスタンス識別子です。使用可能なインスタンスの最大値については、ロジックコントローラーのプログラミングガイドを参照してください。

プロパティ	値	説明
シンボル	ユーザー定義テキスト	シンボルはオブジェクト固有の識別子です。詳細については、シンボルの定義と使用 *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を参照してください。
Link	**1 - SL1** **2 - SL2**	モデムが設定されているシリアル回線 (**設定**タブ)。
タイムアウト	0...255 100 ms 単位で指定され、初期値は 100 (10 s) です。 値が 0 の場合はタイムアウトが強制されません。	タイムアウトはモデムから応答を受信するまで待機する最大時間の設定です。 タイムアウトになると、通信はエラーとなって終了しエラーコードを生成します (%SEND_RECV_SMSi.CommError = 16 進数 01)。タイムアウト後、システムが応答を受信してもこの応答は無視されます。 **注記**：ファンクションブロックで設定されたタイムアウトは、SoMachine Basic 設定画面で設定された値を上書きします。詳細については、ロジックコントローラーのプログラミングガイドを参照してください。
インデックス	0...15 **注記**：0 はリストの最初の文字に対応します。	● 送信中はインデックスの値を使用して**メッセージ**テーブルから送信するテキストを選択します。 ● 受信中は受信したテキストと一致する**コマンド**テーブルのインデックスに値が対応します。
VAD	-214748364...2147483647	● 送信中に、SMS のテキスト内のプレースホルダー $VAD は %SEND_RECV_SMSi.VAD の値に置き換わります。 ● 受信中に、%SEND_RECV_SMSi.VAD の値は SMS の**コマンド**テーブル内に格納されているプレースホルダー $VAD の値を受け取ります。
VBD	-214748364...2147483647	● 送信中に、SMS のテキスト内のプレースホルダー $VBD は %SEND_RECV_SMSi.VBD の値に置き換わります。 ● 受信中に、%SEND_RECV_SMSi.VBD の値は SMS の**コマンド**テーブル内に格納されているプレースホルダー $VBD の値を受け取ります。
MASKPHONE	0000000000000000 bin から 1000000000000000 bin	マスクの初期値。 ● 送信中、このマスクは**電話番号**テーブルから SMS の受信者を選択するために使用されます。 例 :0000000000000010 bin = SMS は、**電話番号**テーブルに 2 番目にリストされているの電話番号 (インデックス 1) に送信されます。 ● 受信中は、マスクが**電話番号**テーブルに適用され、有効な発信者電話番号のリストが作成されます。マスクのビットは SMS をロジックコントローラーに送信するために使用した電話番号を示します。 例 :0000000000000100 bin **電話番号**リスト (インデックス 2) の 3 番目の電話番号が SMS を送信したことを意味します。
コメント	ユーザー定義テキスト	オブジェクトに対応するコメント。

オブジェクト

%SEND_RECV_SMS ファンクションブロックには次のオブジェクトがあります。

オブジェクト	説明	値
%SEND_RECV_SMSi.LINK	ポートの選択	プロパティ (*233* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%SEND_RECV_SMSi.TIMEOUT	ファンクションブロックタイムアウト	プロパティ (*233* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%SEND_RECV_SMSi.INDEX	メッセージまたはコマンドテーブルのインデックス	プロパティ (*233* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。

オブジェクト	説明	値
%SEND_RECV_SMSi.VAD	VAD - プレースホルダー A	プロパティ (*233* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%SEND_RECV_SMSi.VBD	VBD - プレースホルダー B	プロパティ (*233* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%SEND_RECV_SMSi.MASKPHONE	電話番号テーブルのエントリーを選択するためのマスクです。	プロパティ (*233* ページ参照) を参照してください。読み書きが可能です。アニメーションテーブルで編集できます。
%SEND_RECV_SMSi.COMMERROR	通信エラーコード	通信エラーコード (*229* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで編集できます。
%SEND_RECV_SMSi.OPERERROR	動作エラーコード	動作エラーコード (*230* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで編集できます。
%SEND_RECV_SMSi.DONE	実行は正常に完了しました。	出力 (*229* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで編集できます。
%SEND_RECV_SMSi.BUSY	実行中です	出力 (*229* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで編集できます。
%SEND_RECV_SMSi.ABORTED	実行がキャンセルされました	出力 (*229* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで編集できます。
%SEND_RECV_SMSi.ERROR	エラーが検出されました	出力 (*229* ページ参照) を参照してください。読み込み専用。アニメーションテーブルで編集できます。

タイミングチャート

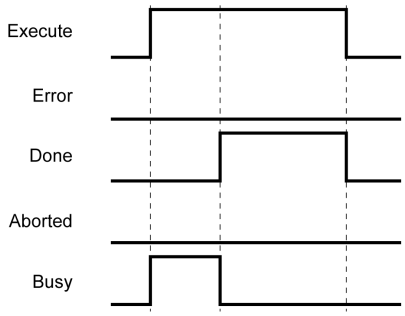
Execute 入力を持つファンクションブロックの信号動作 (*236* ページ参照) を参照してください。

9.6
通信オブジェクトファンクションブロックのタイミングチャート

タイミングチャートの例

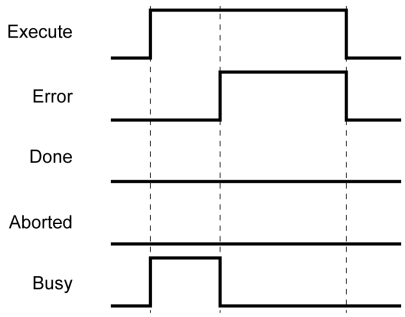
例 1

実行はエラーなしで完了しました。



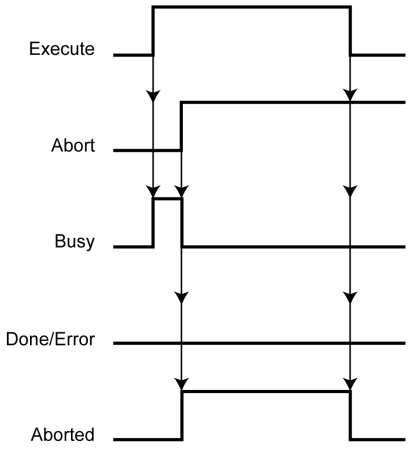
例 2

実行はエラーにより完了しました。

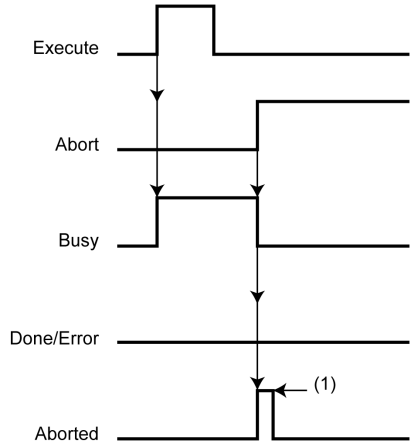


例 3

アプリケーションによって中断されたファンクションブロック



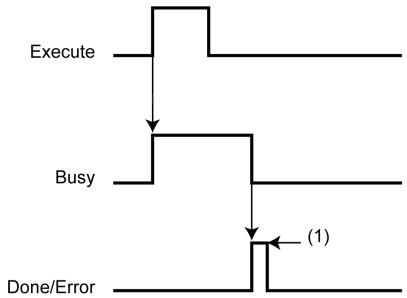
アボートリクエストが発生したときに Execute がすでに FALSE にリセットされている場合、Abort は 1 サイクルだけ TRUE に設定されます。



(1) Execute はすでに FALSE に設定されているので、1 サイクルだけ TRUE に設定します。

例 4

Execute が FALSE に設定された後、実行はエラーなしで完了しました。



(1) Execute はすでに FALSE に設定されているので、1 サイクルだけ TRUE に設定します。

第 10 章
ユーザー定義ファンクション

概略

概要

ユーザー定義ファンクションにより、1 つ以上の入力パラメーター、ローカル変数および戻り値を使用した新しいファンクションを作成できます。ユーザー定義ファンクションは、SoMachine Basic プロジェクトの一部として保存されます。

ユーザー定義ファンクションは、次の場所で呼び出すことができます。

- マスタータスク
- 周期タスク
- Free POU

ユーザー定義ファンクションを作成するにはアプリケーションのファンクションレベル *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を **レベル 6.0** 以上に設定してください。

ユーザー定義ファンクションの使用については、SoMachine Basic オペレーティングガイド - ユーザー定義ファンクション *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を参照してください。

第 11 章
ユーザー定義ファンクションブロック

概略

概要

ユーザー定義ファンクションブロックにより、1 つ以上の入力および出力パラメーター、ローカル変数および戻り値を使用した新しいファンクションブロックを作成できます。ユーザー定義のファンクションブロックは SoMachine Basic プロジェクトの一部として保存されます。

ユーザー定義ファンクションブロックは、次の場所で呼び出すことができます。

- マスタータスク
- 周期タスク
- イベント
- Free POU

ユーザー定義ファンクションブロックを作成するにはアプリケーションのファンクションレベル *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を **レベル 6.0** 以上に設定してください。

ユーザー定義ファンクションブロックの使用については、SoMachine Basic オペレーティングガイド - ユーザー定義ファンクションブロック *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を参照してください。

第 12 章
クロックファンクション

概要

この章ではコントローラーの時間管理機能について説明します。

この章について

この章には次の項目が含まれています。

クロックファンクション

概要

リアルタイムクロック (RTC) 機能を備えたロジックコントローラーでは、SoMachine Basic がロジックコントローラーに接続されてい場合は次に示す時刻クロック機能を使用できます。

- **RTC** ファンクションブロック (*183* ページ参照) は RTC から日時を読み込むか、ロジックコントローラー内の RTC をユーザー定義の日時で更新するために使用されます。
- **スケジュール**ファンクションブロック (*179* ページ参照) は、あらかじめ定義された時間または計算された時間に、アクションを制御するために使用されます。
- **時間 / 日付スタンプ** (*245* ページ参照) は時間と日付をイベントに割り当て、イベントの継続時間を測るるために使用されます。

時刻クロックは プログラムで設定 (*245* ページ参照) できます。コントローラーバッテリーを使用すると、コントローラーの電源がオフになってから最大 1 年間時間設定を維持することが簡単にできます。コントローラーには充電式バッテリーがありません。バッテリーの平均寿命は 4 年です。寿命がくる前に交換してください。バッテリーの交換中にデータを失わないようにするには、バッテリーをコントローラーから取り外した後、120 秒以内にバッテリーを交換してください。

時刻クロックは 24 時間形式で、うるう年を考慮します。

日付スタンプ

概要

システムワード %SW49 から %SW53 には、現在の日付と時刻が BCD 形式で格納されています。この日付と時刻は周辺機器への表示や周辺機器への送信に役立ちます。これらのシステムワードはイベントの日時を格納するために使用できます。

BTI 命令は、日付と時刻を BCD 形式からバイナリ形式に変換するために使用されます。詳細については BCD / バイナリ変換命令 (*62* ページ参照) を参照してください。

イベントの日付決定

日付をイベントに関連付けるにはシステムワードの内容をワードメモリーに送信し、次にこれらのワードメモリーを処理するために割り当て操作を使用すれば十分です (例えば、EXCH 命令)。

プログラミング例

この例は %I0.1 入力の立上がりをどのように日付にするかを示しています

ラング	命令
0	LDR %I0.1 [%MW11:5:=%SW49:5]

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

イベントが検出されると、ワードテーブルには以下が含まれます。

エンコーディング	最上位バイト	最下位バイト
%MW11	-	曜日 (1)
%MW12	00	秒
%MW13	時間	分
%MW14	月	日
%MW15	世紀	年
(1) 1 = 月曜日、2 = 火曜日、3 = 水曜日、4 = 木曜日、5 = 金曜日、6 = 土曜日、7 = 日曜日		

ワードテーブルの例

2013 年 6 月 3 日 (月) 13:40:30 のデータ例

ワード	値 (16 進数)	意味
%MW11	0001	月曜日
%MW12	0030	30 秒
%MW13	1340	13 時間 40 分
%MW14	0603	06 = 6 月、3 日
%MW15	2013	2013

前回の停止日時

システム・ワード %SW54 ～ %SW57 には前回の停止日時が入り、%SW58 には停止の原因を示すコードが BCD 形式で格納されます。

日時の設定

概要

次のいずれかの方法を使用して、ロジックコントローラーの日時設定を更新できます。

- SoMachine Basic の **通信** タブの **RTC の管理** タブを使用します。この方法は オンラインモード *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* でのみ利用可能です。
  2 つの方法の中からどちらかを選ぶことができます。
  - 手動 : 時刻 / 日付ピッカーが表示され、ロジックコントローラーの時刻設定を手動で行えます。
  - 自動 : SoMachine Basic を動かしている PC の時刻設定を利用します。

  詳細は RTC の管理 *(EcoStruxure Machine Expert - Basic, オペレーティングガイド)* を参照してください。
- プログラムでは RTC ファンクションブロック (*183* ページ参照) を使用します。
- オンラインモードでは、演算ブロック (%SW49 から %SW53 またはシステムワード %SW59) を使用して直接またはプログラムによりシステムワードを更新します。

注記: 日時はロジックコントローラーで RTC 機能が使用可能な場合にのみ設定できます (ロジックコントローラーの *プログラミングガイド* を参照してください)。

%SW49 から %SW53 の使用

システムワード %SW49 から %SW53 を日時の設定に使用するには、ビット %S50 を 1 に設定します。%S50 が 1 の間は、システムワード %SW49 ～ %SW53 はコントローラーによって更新されません。%S50 の立下がり (%S50 を 0 に設定) で、%SW49 ～ %SW53 の値でコントローラーの内部 RTC は更新されます。コントローラーは RTC を使用して %SW49 から %SW53 への更新を再開します。

リアルタイムクロック (RTC) ファンクションの日時の値 (BCD 表記) を含むシステムワードを次の表に示します。

システムワード	説明
%SW49	xN 曜日 (N=1 は月曜日)
%SW50	00SS: 秒
%SW51	HHMM: 時間 と 分
%SW52	MMDD: 月 と 日
%SW53	CCYY: 世紀 と 年

システムビットとワードの全リストについては、ロジックコントローラーの *プログラミングガイド*を参照してください。

プログラミング例

ラング	命令	コメント
0	LD %S50 R %S50	–
1	LD %I0.1 [%SW50:=%MW11] [%SW51:=%MW12] [%SW52:=%MW13] [%SW53:=%MW14] S %S50	BCD / バイナリ変換命令 (*62* ページ参照) を参照してください。

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

ワード %MW11 ～ %MW14 には新しい日時 (BCD コードの復習 (*62* ページ参照) を参照) が含まれており、これは %SW50 ～ %SW53 のコードに対応しています。

注記：%SW49 (曜日) は指定の日付に基に自動的に計算されます。

ワードテーブルには新しい日時を含めてください。

エンコーディング	最上位バイト	最下位バイト
%MW11	–	秒
%MW12	時間	分

エンコーディング	最上位バイト	最下位バイト
%MW13	月	日
%MW14	世紀	年

2013 年 6 月 3 日のデータ例

ワード	値 (16 進数)	意味
%MW11	0030	30 秒
%MW12	1340	13 時間 40 分
%MW13	0603	06 = 6 月、3 日
%MW14	2013	2013

%SW59 を使用する

日時を更新するもう 1 つの方法は、システムビット %S59 および日付調整用システムワード %SW59 を使用する方法です。

ビット %S59 を 1 に設定すると、現在の日時をワード %SW59 で調整することができます。%SW59 は立上がりで日時の各要素を増減します。

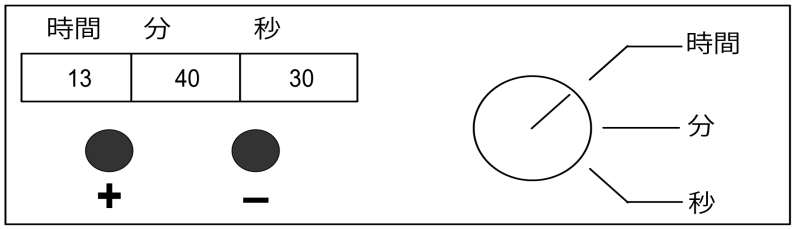
日付パラメーター調整用のシステムワード %SW59 の各ビットを次の表に示します。

インクリメント	デクリメント	パラメーター
ビット 0 :	ビット 8 :	曜日 (1)
ビット 1 :	ビット 9 :	秒
ビット 2 :	ビット 10 :	分
ビット 3 :	ビット 11 :	時間
ビット 4 :	ビット 12 :	日
ビット 5 :	ビット 13 :	月
ビット 6 :	ビット 14 :	年
ビット 7 :	ビット 15 :	世紀 (1)
(1) 曜日および世紀は、ユーザーによって変更 (増減) することはできません。		

システムビットとワードの全リストについては、ロジックコントローラーの *プログラミングガイド* を参照してください。

使用例

内部時計の時、分、秒を変更するために、次のフロントパネルを作成します。



コマンドの説明

- 時 / 分 / 秒スイッチは、それぞれ入力 %I0.2、%I0.3、および %I0.4 を使用して変更するタイムディスプレイを選択します。
- プッシュボタン "+" は、%I0.0 入力を使用して選択されたタイムディスプレイをインクリメントします。
- プッシュボタン "-" は、%I0.1 入力を使用して選択されたタイムディスプレイをデクリメントします。

このプログラムはパネルからの入力を読み取り、内部クロックを設定します。

ラング	命令	コメント
0	LD %M0 ST %S59	–
1	LD %I0.2 ANDR %I0.0 ST %SW59:X3	時間
2	LD %I0.2 ANDR %I0.1 ST %SW59:X11	–
3	LD %I0.3 ANDR %I0.0 ST %SW59:X2	分
4	LD %I0.3 ANDR %I0.1 ST %SW59:X10	–
5	LD %I0.4 ANDR %I0.0 ST %SW59:X1	秒
6	LD %I0.4 ANDR %I0.1 ST %SW59:X9	–

注記：ラダー図を入手するには、可逆性の手順 (*14* ページ参照) を参照してください。

用語集

%

IEC 規格により、% はプログラム変数、定数、I/O などを格納するロジックコントローラーの内部メモリーアドレスを識別する接頭辞です。

%Q

IEC 規格により、%Q は出力ビット (例えば、デジタル OUT タイプのオブジェクト) を表します。

アナログ入力

受け取った電圧または電流を数値に変換します。ロジックコントローラー内にこれらの値を格納し処理可能。

アナログ出力

ロジックコントローラー内の数値を変換し、比例する電圧または電流を出力します。

アプリケーション

設定データ、シンボル、ドキュメントを含むプログラム。

インストラクションリスト言語

コントローラーにより順に実行される一連のテキストベースの命令で書かれたプログラム。各命令は、ライン番号、命令コードおよびオペランドを含みます。(IEC 61131-3 を参照してください。)

コントローラー

産業プロセスを自動化します。(プログラマブルロジックコントローラーまたはプログラマブルコントローラーとして知られる。)

データロギング

オブジェクトや文字列のデータを永久的に保存します。

ファンクションブロック

入力が 1 点または複数点あり、直ちに 1 つまたは複数の結果を返すプログラミング単位。FBs は、インスタンス (専用の名前と変数を持つファンクションブロックのコピー) を介して呼び出され、各インスタンスは呼び出しから呼び出しまで持続的な状態 (出力および内部変数) を保ちます。

例 : タイマー、カウンター

プログラム

アプリケーションのコンポーネント。コンパイルされたソースコードで構成され、ソースコードはロジックコントローラーのメモリーにインストール可能。

ラダー図言語

コントローラープログラムの命令を表す図。コントローラーで順次実行される一連のラングにある接点、コイル、およびブロックのシンボルを含む。(IEC 61131-3 を参照してください。)

ループ要素

オフラインモードでプログラムに命令のシーケンスを実装できます。

拡張バス

拡張 I/O 拡張モジュールとコントローラー間の電子通信用バス。

条件付き要素

オフラインモードでプログラムに条件を実装できます。

設定

システム内のハードウェアコンポーネントの配置と接続、およびシステムの動作特性を決めるハードウェアおよびソフトウェアパラメーターの設定。

ASCII

(*American standard code for Information Interchange*、*情報交換用アメリカ標準コード*) 英数字を表すプロトコル (文字、数字、特定のグラフィックおよび制御文字。)

I/O

(*入力 / 出力*)

RTC

(*real-time clock*、*リアルタイムクロック*) 電池の寿命の間、コントローラーに給電されていない時でも継続して動作する電池バックアップ式日時およびカレンダークロック。

索引